Proceedings of the Seminar in Computer Science (CS-E4000), Spring 2023

Antti Ylä-Jääski and Wencan Mao

Tutors for seminar topics

Suoranta Sanna, Jose Luis Martin Navarro, Aura Tuomas, Ylä-Jääski Antti, Gunn Lachlan, Welsch Robin, Di Francesco Mario, Truong Linh, Sehad Nassim, Shushu Liu, Peltonen Aleksi, Lindqvist Blerta, Chren Stanislav, Zhu Shibei, Bufalino Jacopo, Mauranen Henry, Rannisto Antti, Saif Abduljalil, Gonzalez Torres Ana, Bharti Ayush, Lai Russell W. F., Korpi-Lagg Maarit, Siekkinen Matti, Harjuhahto Jaakko, Zhanabatyrova Aziza, Paler Alexandru, Jäntti Riku, Ollila Esa, and Siekkinen Matti

Preface

The *Seminar on Network Security*, *Seminar on Internetworking* and *Seminar on Software Technology and Systems Research* were previously separate Master's level courses in computer science at Aalto University. These seminar courses have now merged into one seminar course. These seminar series have been running continuously since 1995. From the beginning, the principle has been that the students take one semester to perform individual research on an advanced technical or scientific topic, write an article on it, and present it on the seminar day at the end of the semester. The articles are printed as a technical report. The topics are provided by researchers, doctoral students, and experienced IT professionals, usually alumni of the university. The tutors take the main responsibility of guiding each student individually through the research and writing process.

The seminar course gives the students an opportunity to learn deeply about one specific topic. Most of the articles are overviews of the latest research or technology. The students can make their own contributions in the form of a synthesis, analysis, experiments, implementation, or even novel research results. The course gives the participants personal contacts in the research groups at the university. Another goal is that the students will form a habit of looking up the latest literature in any area of technology that they may be working on. Every year, some of the seminar articles lead to Master's thesis projects or joint research publications with the tutors.

Starting from the Fall 2015 semester, we have merged the three courses into one seminar that runs on both semesters. Therefore, the theme of the seminar is broader than before. All the articles address timely issues in security and privacy, networking technologies and software technology.

These seminar courses have been a key part of the Master's studies in several computer-science major subjects at Aalto, and a formative experience for many students. We will try to do our best for this to continue. Above all, we hope that you enjoy this semester's seminar and find the proceedings interesting.

Seminar papers

Debugging, Logging and Monitoring ML Systems: Techniques and Tools

Ali Ghazal
ali.ghazal@aalto.fi

Tutor: Hong-Linh Truong

Abstract

KEYWORDS: Observability, Debugging, Logging, Monitoring, Software Quality, Machine Learning

1 Introduction

In recent years, Machine learning (ML) models have been a central part of most modern applications. Such models are at the heart of e-commerce applications, streaming services, and search engines. These models manage the critical infrastructure of the banking, healthcare, and transportation industries. As those systems mature, it has become increasingly important to develop observability for debugging, logging, and monitoring the operation of the ML component in those systems. Traditionally, ML models were treated as black boxes. That is, in cases of performance degradation, it is difficult to know which combination of data points and hyperparameters had this influence on the model and why. The ad-hoc way of handling the failures of ML systems was through repeated experimentation and hyper-parameters tunning.

The ML community has proposed a diverse set of tools and techniques to address this problem. For logging, industry tools such as MLflow [24]

and Weights & Biases [1] are widely adopted to track experiments during the training phase of those models. Data Unit Tests [20] and Data Debuggers [19] were introduced to ensure the quality of data coming through the system. Such tools are tailored for individual components in the ML pipeline without providing end-to-end visibility into the platform's performance. A handful of tools were proposed to fill this gap, such as Apple's Overton [18] and IBM's Maro [4]. However, the majority of these tools are not optimal. Overton requires exclusive use of their platform for the entire pipeline [18], and Maro only supports Scikit-learn and requires access to the history of previous training rounds.

This paper explores the different techniques and tools that could be used to monitor, debug, and log machine-learning pipelines.

2 Background: Machine Learning Pipelines

Machine learning models are an integral part of a variety of applications spanning a multitude of domains. Regardless of the downstream task, the majority of those models undergo the same development and deployment lifecycle. As shown in figure 1, ML development lifecycle consists of three main stages: data management, model development, and model serving [10].

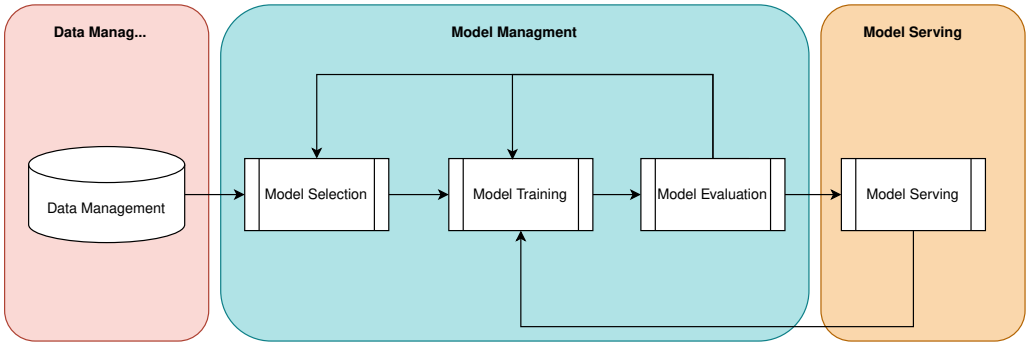


Figure 1. ML lifecycle. It includes data management, model training and evaluation, and model serving.

The data management stage encapsulates the techniques and tools used to store and transform the data used in ML pipelines. A multitude of tools are published every year to optimize the storage and retrieval of different formats of data over various file systems. Upon retrieval, data points typically undergo multiple stages of transformations in order to be usable by the model. For example, most ML frameworks provide tools to impute missing values, scale, reshape, or even drop data points. Further,

to compensate for the lack of real data, many researchers started adopting various data augmentation techniques to their pipelines. To mention a few, researchers augment their data by applying linear transformations or adding white noise to the data [23]. In addition, they could depend on entirely synthetic data generated from auto-encodes and generative adversarial models [21].

The model development stage represents the activities involved with selecting the hyper-parameters of the systems and fitting the parameters of the selected model. The process of selecting the optimal hyper-parameters is iterative in nature. That is, researchers run different experiments with different combinations of hyperparameters before selecting the optimal ones. Those experiments could be performed manually or automatically. Multiple tools, such as MlFlow [24], were introduced to facilitate keeping records of the different experiments and their results. Further, tools, such as AutoML [7], perform an optimized search for the optimal hyperparameters in more constrained ranges, thus leading to convergence faster.

Lastly, the model serving stage describes the process of exposing the ML model for users to query. This could be achieved by manually packaging the model and serving it through a RESTful API. Alternatively, this could be realized through using out-of-the-box tools such as TensorFlow Serving, AWS Sagemaker, or Cortex.

Models typically are subject to iterations of retraining to incorporate the newly collected data. Most out-of-the-box model serving tools facilitate the retraining process [15]. Thus, periodically, data gets transformed, and models get re-trained and served. However, this process is error-prone since there are no guarantees that the newly added data points are semantically correct. Thus, the need for rigorous observability mechanisms has become more pressing.

3 Data: monitoring and debugging techniques

Machine learning applications are typically described as data-driven programming [2]. The performance of a model is bounded by the quality of the data used during training. Similarly, if the performance of a model is degrading, this would be directly related to the degraded quality of the training data. Thus, the task of debugging ML models revolves around debugging their training data. To mention a few, data could be corrupted because of skewed data distribution, misconfiguration errors in the pre-

processing pipeline, or inaccurate labeling.

Numerous techniques are proposed by the research community to address those problems. Dagger was introduced to be a "data debugger." Dagger provides a framework-agnostic interface that provides high-level abstractions that enable researchers to monitor and debug the lineage of the data through their ML pipeline [19]. In Dagger's terminology, each pre-processing step is thought of as a "block." Those blocks are used to construct a directed acyclic graph (DAG) to describe the execution of the pipeline. As data points undergo different transformations through each block, Dagger keeps track of the changes, *Deltas*, happening to each row of data. Dagger also could be used to insert data breakpoints between different pre-processing blocks. Further, Dagger provides an interface to split the pipeline resulting in forked pipelines running using different hyperparameters [19]. Thus, it enables comparing the performance of those different pipeline splits. Lastly, Dagger comes with a DQL (Dagger Query Language) that could be used to query the logging manager for different versions of the data at different stages of the pipeline [19]. In addition to DQL, a preliminary version of Dagger is integrated into data civilizer 2.0, enabling researchers to debug their pipeline through an intuitive GUI.

Additionally, MLinspect targets the data distribution problem [6]. It is framework and language-independent. Similar to Dagger, It transforms the pipeline into a graph presentation. However, in MLinspect, nodes are snapshots of the data and edges are the transformations applied over the data. At each node, it keeps track of the data distribution of the different variables in the dataset. MLinspect comes with a GUI where users could hover over the node and explore the statistics associated with each snapshot of the data.

As we could have noticed in the aforementioned tools, debugging is performed at the data level independently of the performance of the downstream prediction task. Flokas et al. introduced a complaint-based debugging mechanism to solve this problem [5]. Basically, domain experts could define rules describing how the model should be performing. Then, the data is transformed to enable the model to follow the pre-specified rules. Further, to avoid the cost of retraining the model with the newly refined data, Flokas et al. utilized Influence Functions [9] to manipulate the behavior of the model until it reaches the desired state.

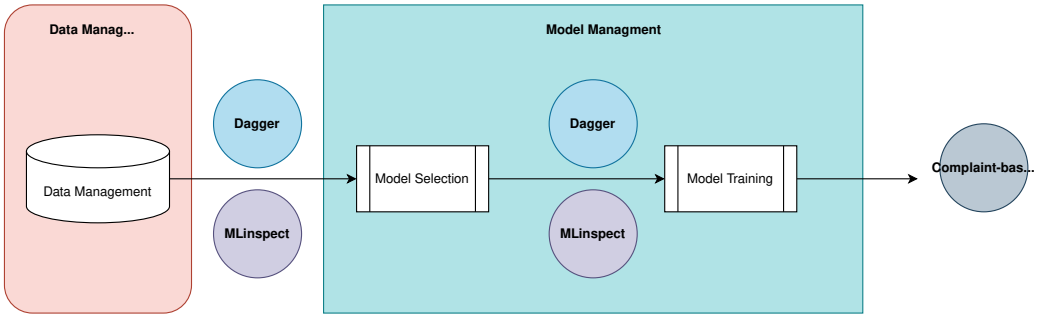


Figure 2. A summary of the different tools that could be used to debug data related problems

4 Models: monitoring and debugging techniques

The second main component of the ML pipelines is the model. Assuming a model is malfunctioning, the second step after debugging data would be debugging the model itself. The research community proposes an array of tools that could be used for this purpose. In [3], Ariadne, an analysis tool for machine learning programs, utilized static analysis techniques to localize errors during the development process of the model. BugDoc [13] and Maro [4] incorporated iterative approaches to identify errors' root causes. Further, OMG [8] used model performance assertion to make sure that the model is behaving as expected. Lastly, others [22], [14] proposed using model management and versioning tools to compare the performance of different variations of models.

BugDoc and Maro are iterative debugging tools. On one hand, BugDoc could assist in localizing errors in any computational pipelines. On the other hand, Maro could only be used to debug AutoML pipelines. In essence, both tools are capable of determining the root cause of errors and providing a natural language explanation for what went wrong. However, Maro is additionally capable of automatically remediating the pipeline [4]. Bugdoc takes as input 1) a pipeline description, 2) a history of the runs of the pipeline, and 3) an evaluation function to determine if the pipeline succeeds or fails [13]. Then, it produces a report about the possible root causes of the failure of the pipeline if any.

BugDoc uses two main algorithms to determine the root cause of the error. The first algorithm is refered to as Shortcut. It assists in determining single hyperparameter misconfigurations in the system. To detect more complex misconfiguration errors, a second algorithm, Debugging Decision Tree, is used. The output of both algorithms is fed into the explainer module whose main function is to describe the root cause of the error in

natural language description. On the other hand, Maro models bug localization as a search problem over the set of all possible hyperparameters used in the pipeline. It uses Satisfiability Modulo Theories (SMT) Solvers to localize the error [4], as well as derives a set of constraints over the original model to describe the error. Then, using those constraints, it identifies the best configuration for the pipeline that would eliminate the detected errors. Lastly, similar to BugDoc, the identified constraints are fed to the explainer module which provides a description of the error in natural language.

Realizing the importance of Boolen Assetions in software production, Kang et al. in [8] introduced the idea of model assertions which could be used to enforce expectations over the behavior of the model. Assertions could either be soft (probabilistic) or hard (deterministic). The paper proposed OMG Model Guardian which consists of an API to specify assertions, a runtime system, and a logging mechanism. OMG (OMG Model Guardian) could be used to enforce assertions during both development and deployment. Further, it could be used to encode corrective actions when the model fails any of the pre-defined assertions. Lastly, since the logging manager keeps track of the instances when our model fails, those records could be utilized during the process of active learning. As reported by Kang et al, OMG increased the accuracy of a Single Shot Detection (SSD) [12] model from 40.4% to 82.6% [8].

Lastly, model versioning mechanisms are essential for the effective debugging and monitoring of ML pipelines. The need for model versioning arises from the nature of the ML lifecycle. As developing ML models pertains to extensive experimentation, manually keeping track of all the versions, their related artifacts, and their corresponding performances becomes impractical. Further, without proper versioning and archiving mechanisms, storing snapshots of production models required hundreds of MBs or even GBs. Multiple tools were proposed to solve those problems. ModelDB was one of the earliest systems for versioning ML systems [22]. It had client libraries that could be integrated with ML environments, such as Spark and Scikit-learn. As researchers conduct their experimentations, ModelDB collects metadata about the model parameters in the background and sends it to a hosted backend. Using this system, researchers have access to an interface where users could browse through the different medals and their associated metadata [22]. Furthermore, Modelhub, a lifecycle management tool for deep learning, presented a sys-

tem that enabled effective debugging and exploration for model versions [14]. Modelhub introduced a model versioning system (DLV) along with a Domain Querying Language (DQL) that enables researchers to explore the models in a repository based on the network architecture or any of the selected hyper-parameters. Distinctive features of ModelHub include its efficient storage mechanism along with its optimized query execution planner. Additionally, ModelHub could be deployed to enable model sharing among teams.

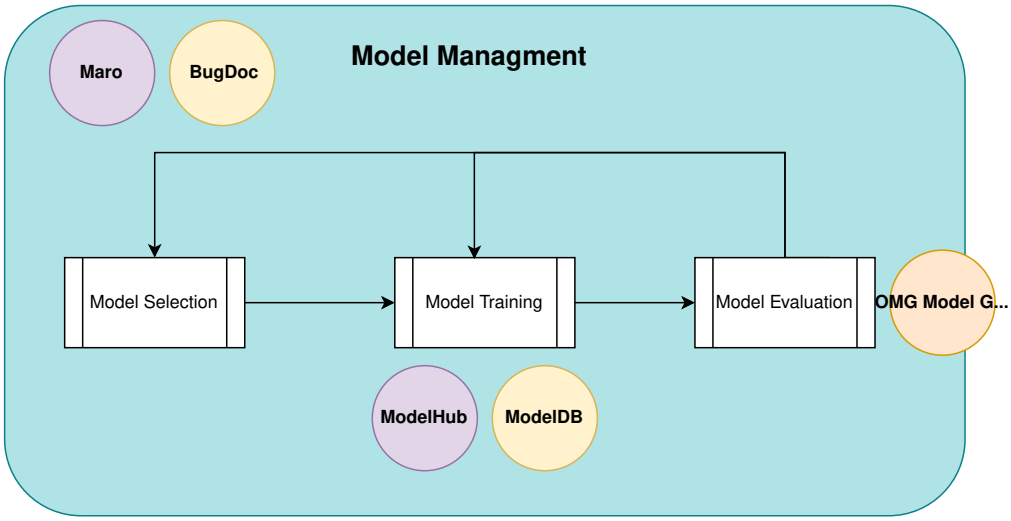


Figure 3. A summary of the different tools that could be used to debug model-related problems

5 Inference: monitoring and logging techniques

The next step in the ML lifecycle is model deployment. Models in production commonly experience performance degradation. This happens because the training data doesn't always have the same distribution as the real-world data. As time progress, changes in the user-generated data might render models ineffective. Additionally, in cases of Continuous Training and Continuous Deployment (CT/CD), the parameters of models are prone to poisoning and drifting. For example, under a brute force attack, a fraud detection model, if left unmonitored, would be trained on an uneven distribution of fraud transactions [17]. Thus, the model might make more false-positive predictions, i.e. predicting that more transactions are fraudulent, leading to a degradation of the overall performance of the system. To address this problem, multiple open-source and proprietary tools were proposed. Most of those tools provided capabilities to monitor and detect drifts. Further, most of them are equipped with log-

ging mechanisms that would enable debugging the performance degradation of models. Since performance monitoring is essential for reliable AI, many enterprise solutions were proposed, such as Amazon SageMaker Model Monitor [16], SAMM by Feedzai [17], and Fiddler.ai. Further, many open-source solutions exist, such as Great Expectations, Angur, and Whylogs. Most of those tools share similar functionalities. In this section, we are presenting the overall architecture of a representable subset of those tools.

Amazon SageMaker Model Monitor is one of the most used model monitoring tools. It is designed for scalability and usability, and it easily integrates with Amazon Web Services (AWS) tools. SageMaker Model Monitor consists mainly of three components: a model monitor, data collection endpoints, and a scheduler. The concept of Data Sketches is essential to understand how SageMaker Model Monitor works [16]. Basically, through the data collection endpoint, a summary of the data is accumulated in a Data Sketch Object. Then, the model monitor component queries those Data Sketch Objects to get different measurements about the model, the input data, and the predictions made. The execution frequency of this process is managed by the scheduler module. SageMaker Model Monitor is equipped with numerous functions that could assist in detecting drifts in model fairness and in model feature attributions. When the Model Monitor detects any of those drifts beyond a preconfigured threshold, it triggers alarms and produces a report with the preliminary root cause analysis [16].

The majority of the monitoring tools assume the existence of immediate labels, a ground truth, which could be used to assess the performance of the model. This is only possible in limited scenarios. For example, consider a case where a model is predicting whether, for example, a user is going to click on a certain link. Then, in a few seconds, the ground truth about the user behavior is reported. However, in most cases, human input is needed for labeling, and this process is slow and unreliable. This motivated the work done on the Streaming System for Automatic Model Monitoring (SAMM) by Feedzai [17]. This system operates on data streams of the input data feed into the model under observation. And it computes a signal indicating the shift in the distribution of the data compared to historical records of data. In [17], the authors proposed a novel, constant-time streaming algorithm SPEAR for percentile estimation using a fixed number of bins. Users could select a certain threshold before alarms could

Enterprise	Drift Detection for unlabled data	Drift Induction and Analysis
AWS SageMaker Model Monitor	SAMM	Augur
Fiddler.ai	-	-
Arize AI	-	-
Evidently	-	-

Table 1. A summary of the different tools that could be used to debug model performance problems

be triggered. Lastly, SAMM includes a configurable response module that could trigger retraining for the model.

In [11], Augur was proposed to address the explainability problem with the monitoring of the performance of the ML models. That is, although different tools such as Amazon SageMaker Model Monitor [16] and SAMM [17] could detect the occurrence of data drifts, they don't provide a systematic analysis of the underlying cause and the type of drifts. Lewis et al. listed the different types of drift that could occur. Concept drifts happen when the underlying relationship between the features and the output, $p(y \mid x)$, changes. Feature drift happens when the distribution of the input to the model, $p(x)$, changes. Label drift happens when the distribution of the output of the model, $p(y)$, changes. Further, those types of drifts could be categorized into continuous, abrupt, and re-occurring [11]. Angur addressed this problem by creating a simulation test environment where different types of data drifts could be induced into the model. Then, it generates a report of the best metrics and thresholds that could be used to detect the different types and categories of drifts.

6 Conclusion

This paper presents an overview of the different technologies and techniques that could be used to debug and monitor Machine Learning applications. As shown in figure 4, The paper was organized according to the ML lifecycle, and it aims to asset systems designers select the right tools for each stage of the ML pipeline. For the data management stage, this paper discussed data flow debuggers, complaint-based data assertions, and data distribution debuggers. For the model fitting stage, an overview of

iterative debugging tools such as Maro and BugDoc was presented. Further, we discussed how ideas from the software industry such as Boolen Assertion and Software Versioning could be used to debug and monitor ML applications. For the model serving stage, we discussed the reasons behind the performance degradation of models. Further, we presented a sample of different enterprise performance monitoring tools such as Amazon SageMaker Model Monitor, SAMM, and Angur.

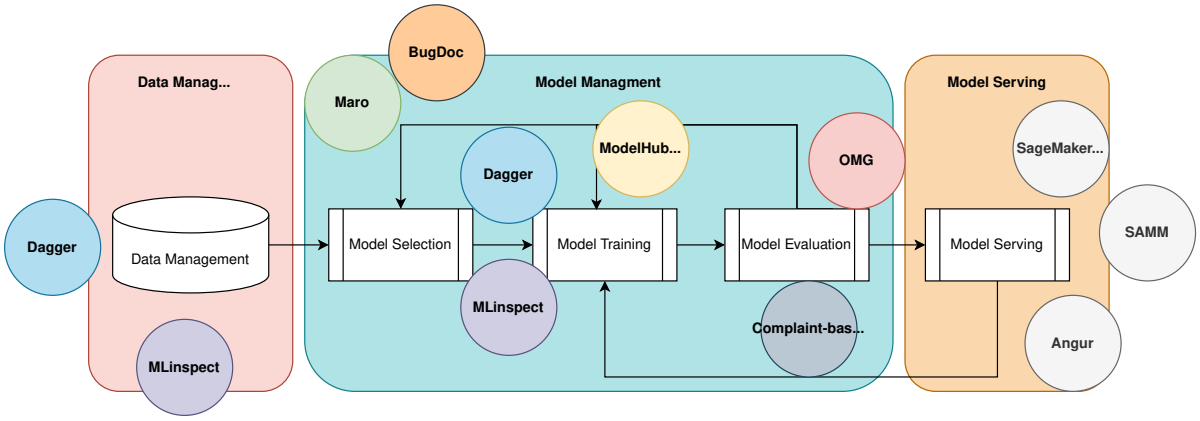


Figure 4. A summary of the different tools that could be used to debug and monitor ML applications.

References

[1] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[2] Malinda Dilhara, Ameya Ketkar, and Danny Dig. Understanding software-2.0: A study of machine learning library usage and evolution. *ACM Trans. Softw. Eng. Methodol.*, 30(4), jul 2021.

[3] Julian Dolby, Avraham Shinnar, Allison Allain, and Jenna Reinen. Ariadne: analysis for machine learning programs. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. ACM, jun 2018.

[4] Julian Dolby, Jason Tsay, and Martin Hirzel. Automatically debugging automl pipelines using maro: ML automated remediation oracle. In Swarat Chaudhuri and Charles Sutton, editors, *MAPS@PLDI 2022: 6th ACM SIGPLAN International Symposium on Machine Programming, San Diego, CA, USA, 13 June 2022*, pages 60–69. ACM, 2022.

[5] Lampros Flokas, Weiyuan Wu, Jiannan Wang, Nakul Verma, and Eugene Wu. How i stopped worrying about training data bugs and started complaining. In *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, DEEM '22, New York, NY, USA, 2022. Association for Computing Machinery.

[6] Stefan Grafberger, Shubha Guha, Julia Stoyanovich, and Sebastian Schelter. Mlinspect: A data distribution debugger for machine learning pipelines. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2736–2739, New York, NY, USA, 2021. Association for Computing Machinery.

[7] Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, jan 2021.

[8] Daniel Kang, Deepti Raghavan, Peter D. Bailis, and Matei A. Zaharia. Model assertions for debugging machine learning. 2018.

[9] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2017.

[10] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature Cell Biology*, 521(7553):436–444, May 2015. Funding Information: Acknowledgements The authors would like to thank the Natural Sciences and Engineering Research Council of Canada, the Canadian Institute For Advanced Research (CIFAR), the National Science Foundation and Office of Naval Research for support. Y.L. and Y.B. are CIFAR fellows. Publisher Copyright: © 2015 Macmillan Publishers Limited. All rights reserved.

[11] Grace A. Lewis, Sebastián Echeverría, Lena Pons, and Jeffrey Chrabaszcz. Augur: A step towards realistic drift detection in production ml systems. In *2022 IEEE/ACM 1st International Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI)*, pages 37–44, 2022.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[13] Raoni Lourenço, Juliana Freire, Eric Simon, Gabriel Weber, and Dennis Shasha. Bugdoc: Iterative debugging and explanation of pipeline. *The VLDB Journal*, 32(1):75–101, feb 2022.

[14] Hui Miao, Ang Li, Larry S. Davis, and Amol Deshpande. Modelhub: Towards unified data and lifecycle management for deep learning, 2016.

[15] Hui Miao, Ang Li, Larry S. Davis, and Amol Deshpande. Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 571–582, 2017.

[16] David Nigenda, Zohar Karnin, Muhammad Bilal Zafar, Raghu Ramesha, Alan Tan, Michele Donini, and Krishnaram Kenthapadi. Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models, 2022.

[17] Fábio Pinto, Marco O. P. Sampaio, and Pedro Bizarro. Automatic model monitoring for data streams, 2019.

[18] Christopher Ré, Feng Niu, Pallavi Gudipati, and Charles Srisuwananukorn. Overton: A data system for monitoring and improving machine-learned products. *CoRR*, abs/1909.05372, 2019.

[19] El Kindi Rezig, Lei Cao, Giovanni Simonini, Maxime Schoemans, Samuel Madden, Nan Tang, Mourad Ouzzani, and Michael Stonebraker. Dagger: A data (not code) debugger. In *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org, 2020.

[20] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proc. VLDB Endow.*, 11(12):1781–1794, aug 2018.

[21] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data augmentation using gans, 2019.

[22] Manasi Vartak, Harihar Subramanyam, Wei-En Lee, Srinidhi Viswanathan, Saadiyah Husnoo, Samuel Madden, and Matei Zaharia. Modeldb: A system for machine learning model management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA '16, New York, NY, USA, 2016. Association for Computing Machinery.

[23] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc., 2020.

[24] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41(4):39–45, 2018.

A Review of Existing Tools for Automated Formal Verification of Security Protocols

Alina Kostetska
alina.kostetska@aalto.fi

Tutor: Lachlan Gunn

Abstract

Formal verification is a powerful technique for ensuring the security, correctness, and efficiency of security protocols, but it can be complex and time-consuming to conduct manually. Fortunately, many tools have been developed to automate formal verification, and this paper provides an analysis and classification of these tools to help developers and researchers understand their variety and differences and choose the most appropriate one for their security protocol. The paper discusses the applications of formal verification, the challenges associated with it, and various tools available for automated formal verification. The analysis and classification of these tools are presented in a diagram to facilitate their comparison and selection. This paper contributes to the development and improvement of formal verification techniques and tools, ultimately leading to stronger security and correct and efficient security protocols.

KEYWORDS: formal verification, security protocols

1 Introduction

Security protocols are rules designed to protect data and ensure secure communication. They define how information is exchanged between par-

ties and how it is protected from unauthorized access. Typically, such protocols leverage cryptography for data protection. Security protocols are used in many areas, including internet communication, mobile devices, financial transactions, and healthcare. Given their widespread use and critical importance in protecting sensitive information, it is essential to ensure their security and correct implementation.

Formal verification is one technique to verify the correctness and security of security protocols, by checking whether they comply with the specification using formal mathematical proofs. Formal verification provides exhaustive coverage of all possible inputs and paths, and a mathematical proof that predefined properties hold (or do not).

Formal verification can be complex and time-consuming, so many tools have been developed for its automation, including [6, 16, 7]. Their main purpose is to help create formal models of security protocols, and automatically verify the desired properties. They have proved their efficiency before [13, 19, 22].

Different tools usually focus on specific properties they prove. Since most of the tools have their own descriptive language for models, it is difficult to use several of them, as it would require a new model specification. With a huge number of existing tools, it may be difficult to understand their differences and advantages to choose a suitable one. Also, there exists an implementation gap [22]. Most of the formal verification tools only verify models described in a specific formal language. It is still a challenge to provide a source code (or a machine code) that would be just as secure and correct.

To address these problems, this paper discusses how formal verification can be used to reach different goals, including the security of a protocol, correctness, efficiency, and security of implementations. In addition, this paper provides analysis and classification of existing formal verification tools in form of Fig. 1. This can help developers and researchers understand the variety of formal verification tools, choose a tool suitable for their security protocol, and understand how (and if) it can be translated to a high-level language or machine code.

This paper is organized as follows. Section 2 discusses applications of formal verification. Section 3 reviews existing tools and presents them in a diagram showing possible connections and translations. Section 4 discusses existing challenges formal verification faces when dealing with security protocols. Finally, Section 5 presents some concluding remarks.

2 Applications of formal verification

Formal verification of cryptography can be applied to three main areas: protocol-level security, functional correctness and efficiency, and security of implementations [20]. Different tools exist addressing each of these areas, but the real challenge is to combine them and prove that a secure design results in a secure, correct, and efficient implementation. This section describes reasons and approaches to formal verification in each of the areas.

2.1 Protocol-level security

Protocol-level security can ensure that a security protocol is resistant to many attacks that can be caused by bad specifications. It is usually achieved by using symbolic and computational mathematical security models [5].

Symbolic models
Symbolic models, also known as the Dolev-Yao models, define cryptographic primitives as black-boxes, with atomic terms as their arguments that cannot be further divided into smaller components [27]. An adversary in such models is an abstract machine that can only use predefined primitives. It ensures that any possible actions and their combinations are not leading to a successful attack.

The majority of automated formal verification tools use a symbolic model because it is relatively simple to use and automate. Meanwhile, this model may be less realistic and therefore accurate than a computational model.

Computational models
Computational models present arguments as bitstrings, and cryptographic primitives as probabilistic algorithms [27]. An adversary of such model is a probabilistic Turing machine. It means that security holds if an adversary cannot efficiently solve a hard computational problem.

A computational model is more realistic, usually used by cryptographers [5]. It implies a stronger security notion than a symbolic model but is more difficult to prove.

2.2 Functional correctness and efficiency

Functional correctness means that a design provides correct results on all valid inputs. It can be achieved by proving equivalence to a reference implementation or meeting a functional specification that usually includes pre- and post-conditions that describe requirements for a program's inputs and outputs.

For practical reasons, programs should also be efficient. It is often achieved by heavy optimizations, or using low-level languages. While functional correctness may be checked on different levels (including protocol model, and source code), efficiency is measured while running machine code. It can be difficult to combine both and ensure machine code guarantees functional correctness.

2.3 Security of implementations

Protocol-level security can protect from many attacks, and implementations that follow the protocol specification will be secure with respect to the design. Meanwhile, in the real world, attackers are not bounded by what exists in the design. They can observe other useful information, including timing patterns, and power consumption, which can lead to side-channel attacks. Side-channel attacks rely on such information and may lead to key recovery or leakage of other secrets. Proving the security of implementations means proving that a program can withstand these types of attacks, usually by ensuring it runs in constant time for all the inputs. Also, implementation security means avoiding any kind of possible weaknesses, including memory vulnerabilities.

3 Automated formal verification

Formal verification can be applied to check the security of protocols and their implementations, and their correctness and efficiency. Fig. 1 aims to categorize different tools and languages that aid in formal verification.

The items in the Fig. 1 can be divided into five groups:

- formal verification tools that use symbolic model (located in the top left corner);

- formal verification tools that use computational model (located in the top right corner);

- assistive tools that allow translating a model from one syntax to another;

- formal description languages (in the center);

- other programming languages, including JavaScript, C (found at the bottom).

This section presents a more detailed description and analysis of the tools for automated formal verification. Tools that work in symbolic and computational models usually verify security properties only. Correctness and efficiency may be in the scope of assistive tools and formal description languages. Security of implementations is partially addressed in languages, such as hacspec and Jasmin [22].

Tools that work in a symbolic model

ProVerif is one of the most popular tools for security protocol verification that works in a symbolic model [6]. It is fully automatic, and works for an unbounded number of sessions and messages, but still has some defects, partially addressed by existing extensions [18]. They deal with stateful protocols or help verify protocols with algebraic properties.

Tamarin prover is another popular tool for security protocols analysis [23]. It also works in a symbolic model and has both automated and interactive modes. DeepSec is a verification tool that works for a bounded number of sessions [9]. ProVerif, Tamarin, and DeepSec have been recently combined into the SAPIC+ tool, which allows to conveniently write one model and use it with any of the three tools [24].

AVISPA is a popular suite of tools for formal verification [34]. It includes four tools OFMC, SATMC, CL-AtSe, and TA4SP. OFMC has been proven to run faster if parallelized [25]. Other tools that work in a symbolic model include AVANTSSAR (a successor of AVISPA) [12], AKISS [7], Casper/FDR [30], CryptoSolve [14], CPSA [28], APTE [8], Maude-NPA [11], Scyther [10], and scyther-proof [31]. A relatively new tool called Verifpal aims to be intuitive and easy to use [26]. It has a list of predefined primitives and can be translated to the ProVerif syntax.

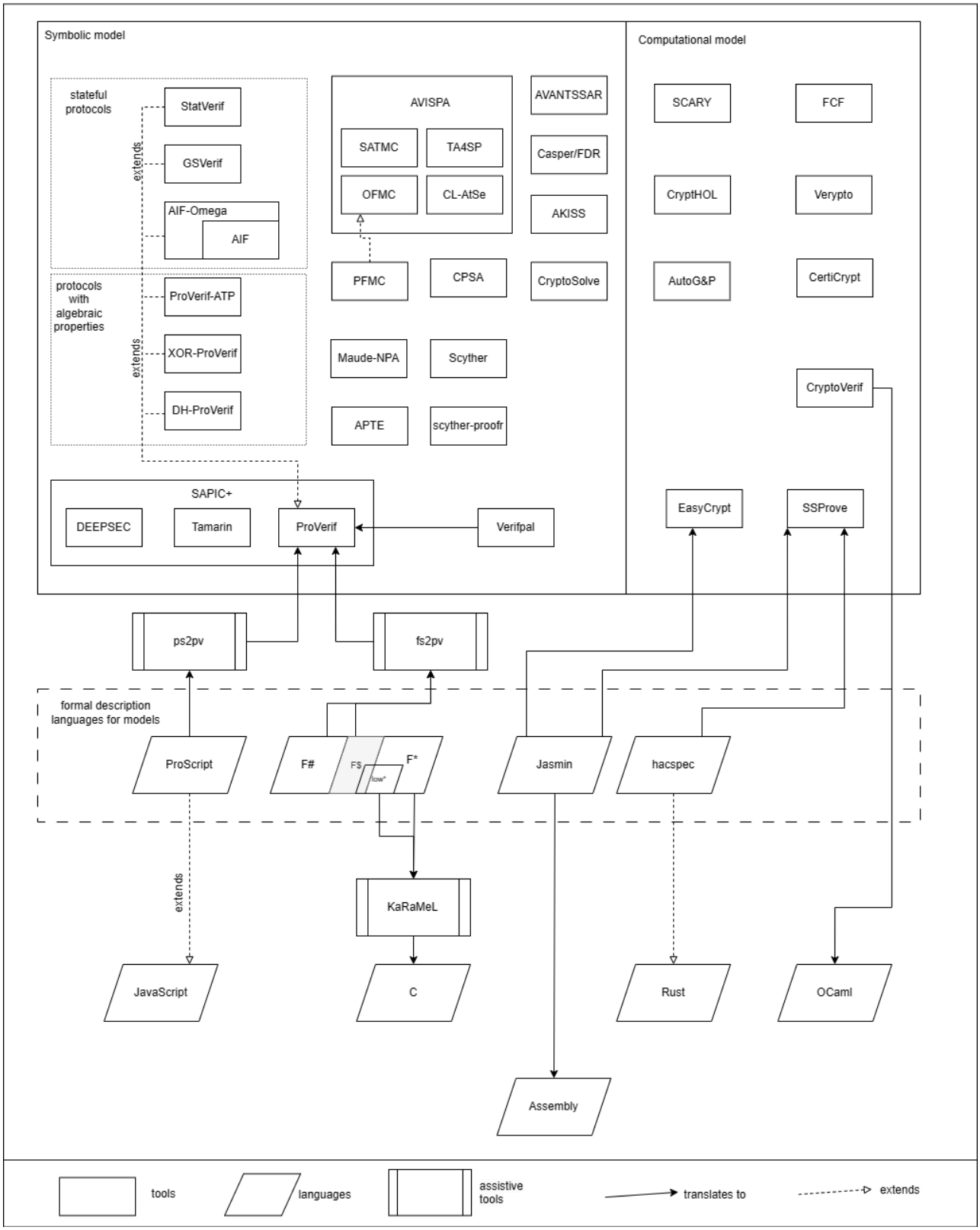


Figure 1. Tools that automate formal verification of security protocols

Tools that work in a computational model

CryptoVerif and EasyCrypt are amongst the most popular tools that work in a computational model. CryptoVerif is semi-automatic and uses games to generate proofs, but does not reconstruct attacks [4]. Compared to CryptoVerif, EasyCrypt is a less automated tool, but it is capable of performing more subtle proofs [16]. Scary is a fully automatic prover that works in a computational model, but combines it with a symbolic model [33]. Others include CryptHOL [29], AutoG&P [17], CertiCrypt [15], FCF [32], SSProve [21].

Formal description languages for models

Verification tools usually have their own descriptive languages for models. In some cases, it can be convenient to use other languages to write a model, and then translate it to a verification tool syntax. ProScript can be used as an example of such language [2].

Assistive tools

Different tools exist to translate a model from one syntax to another. It can be useful when building connections (proving equivalences) between a model and its implementation. fs2pv can translate models written in F# to ProVerif [3]. ps2pv is an experimental tool to translate ProScript to ProVerif [2]. KaRaMeL (formerly KReMLin) is a tool that extracts an F* program to readable C code [1].

Solutions for model-to-implementation mapping

Several papers have proposed solutions for translating formal models into protocol implementations. These approaches can be useful in addressing this challenge:

- the BIFROST approach allows to connect C code (implementation) with its verified model in ProVerif by utilizing F* [13];

- a model written in a subset of F* can be translated to ProVerif via fs2pv, and to C via KaRaMeL [19];

- a model written in Jasmin can be translated to and verified with EasyCrypt, and translated to Assembly [19];

- a model written in hacspec (a subset of Rust language) can be translated

to and verified with SSProve [22];

- a model written in Jasmin can be translated to and verified with SSProve [22].

4 Discussion

Formal verification is a highly active research domain that has given rise to several tools that utilize different approaches to formal verification. However, there is still a lack of standards and common practices in the field, posing a challenge for users seeking to utilize formal verification. Attempts have been made to address this challenge through papers that aim to combine existing tools into one [24] or develop new tools that are compatible with older ones [6]. Despite such efforts, the adoption of formal verification remains limited, and more work is needed to establish widely accepted standards and practices.

Formal verification faces another challenge with respect to the translation of formally verified models into secure, correct, and efficient implementations. Many tools that facilitate formal verification operate independently and do not consider the eventual translation of the model into a high-level language (or Assembly). Several papers have attempted to address this gap by developing tools for model translation [13, 19, 22], but limitations remain.

5 Conclusion

In conclusion, formal verification tools can play an important role in ensuring the correctness, security, and efficiency of software and protocols. However, with the vast number of existing tools and the implementation gap between formal models and actual code, it can be challenging to choose and use the right tool for a particular task. This paper addresses these challenges by providing an overview of the tools of formal verification and their classification. It can help developers and researchers better understand and utilize formal verification tools.

References

[1] Karamel. https://github.com/FStarLang/karamel. Accessed: 2023-04-12.

[2] ps2pv. https://github.com/Inria-Prosecco/proscript-messaging. Accessed: 2023-04-12.

[3] K. Bhargavan, C. Fournet, A.D. Gordon, and S. Tse. Verified interoperable implementations of security protocols. In *19th IEEE CSFW*, pages 14 pp.–152, 2006.

[4] Bruno Blanchet. CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols. 2007.

[5] Bruno Blanchet. Security protocol verification: Symbolic and computational models. In *Principles of Security and Trust*, pages 3–29, Berlin, Heidelberg, 2012.

[6] Bruno Blanchet, Vincent Cheval, and Véronique Cortier. ProVerif with lemmas, induction, fast subsumption, and much more. In *IEEE S&P'22*, pages 205–222, San Francisco, May 2022.

[7] Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. pages 108–127, 03 2012.

[8] Vincent Cheval. APTE: An Algorithm for Proving Trace Equivalence. In *TACAS*, pages 587–592, Berlin, Heidelberg, 2014.

[9] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. The deepsec prover. In *Computer Aided Verification*, pages 28–36, Cham, 2018.

[10] Cas J. F. Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer Aided Verification*, pages 414–418, Berlin, Heidelberg, 2008.

[11] Santiago Escobar, Catherine Meadows, and José Meseguer. *Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties*, pages 1–50. Berlin, Heidelberg, 2009.

[12] Alessandro Armando et al. The AVANTSSAR Platform for the Automated Validation of Trust and Security of Service-Oriented Architectures. pages 267–282, 03 2012.

[13] Camille Sivelle et al. Automatic implementations synthesis of secure protocols and attacks from abstract models. In *Secure IT Systems*, pages 234–252, Cham, 2022.

[14] Dalton Chichester et al. CryptoSolve: Towards a Tool for the Symbolic Analysis of Cryptographic Algorithms. *EPTCS*, 370:147–161, sep 2022.

[15] Gilles Barthe et al. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on POPL*, POPL '09, page 90–101, New York, NY, USA, 2009.

[16] Gilles Barthe et al. *EasyCrypt: A Tutorial*, pages 146–166. Cham, 2014.

[17] Gilles Barthe et al. Automated Proofs of Pairing-Based Cryptography. In *Proceedings of the 22nd ACM SIGSAC CCS*, CCS '15, page 1156–1168, New York, NY, USA, 2015.

[18] Jiangyuan Yao et al. Formal Verification of Security Protocols: ProVerif and Extensions. In *Artificial Intelligence and Security*, pages 500–512, Cham, 2022.

[19] José Almeida et al. The Last Mile: High-Assurance and High-Speed Cryptographic Implementations. 04 2019.

[20] Manuel Barbosa et al. SoK: Computer-Aided Cryptography. Cryptology ePrint Archive, Paper 2019/1393, 2019. https://eprint.iacr.org/2019/1393.

[21] Philipp G. Haselwarter et al. SSProve: A Foundational Framework for Modular Cryptographic Proofs in Coq. Cryptology ePrint Archive, Paper 2021/397, 2021. https://eprint.iacr.org/2021/397.

[22] Philipp G. Haselwarter et al. The Last Yard: Foundational End-to-End Verification of High-Speed Cryptography. Cryptology ePrint Archive, Paper 2023/185, 2023. https://eprint.iacr.org/2023/185.

[23] Simon Meier et al. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*, pages 696–701, Berlin, Heidelberg, 2013.

[24] Vincent Cheval et al. Sapic+: protocol verifiers of the world, unite! Cryptology ePrint Archive, Paper 2022/741, 2022. https://eprint.iacr.org/2022/741.

[25] Alex James, Alwen Tiu, and Nisansala Yatapanage. PFMC: a parallel symbolic model checker for security protocol verification, 2022.

[26] Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. Verifpal: Cryptographic Protocol Analysis for the Real World. Cryptology ePrint Archive, Paper 2019/971, 2019. https://eprint.iacr.org/2019/971.

[27] Baiyu Li. *Computational and Symbolic Models for Secure Computation*. PhD thesis, University of California, San Diego, USA, 2020.

[28] Ramsdell J.D. Guttman J.D. Rowe P.D. Liskov, M.D. The Cryptographic Protocol Shapes Analyzer: A Manual. 2016.

[29] Andreas Lochbihler and S. Reza Sefidgar. A tutorial introduction to CryptHOL. Cryptology ePrint Archive, Paper 2018/941, 2018. https://eprint.iacr.org/2018/941.

[30] G. Lowe. Casper: a compiler for the analysis of security protocols. In *Proceedings 10th CSFW*, pages 18–30, 1997.

[31] Simon Meier, Cas Cremers, and David Basin. Efficient construction of machine-checked symbolic protocol security proofs. *Journal of Computer Security*, 21:41–87, 02 2013.

[32] Adam Petcher and Greg Morrisett. The Foundational Cryptography Framework. abs/1410.3735, 2014.

[33] Guillaume Scerri. Proof of security protocols revisited. Jan 2015.

[34] Luca Viganò. Automated Security Protocol Analysis With the AVISPA Tool. *ENTCS*, 155:61–86, 2006.

Energy saving capabilities of Kubernetes

Andrea Amadei
andrea.amadei@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

The paper explores the challenge of energy efficiency in the cloud computing industry, which has been growing rapidly due to the increasing demand for cloud services. Data centers consume a significant amount of electrical power, and traditional hardware-based solutions have been insufficient in addressing energy inefficiencies. However, software-only solutions such as containers and container orchestration systems have emerged as promising alternatives. The paper focuses on scheduling algorithms as means to improve energy efficiency in the cloud, both for homogeneous and heterogeneous clusters, while providing a meaningful comparison between them.

KEYWORDS: cloud computing, energy efficiency, virtualization, containers, container orchestration, Kubernetes, scheduling algorithms

1 Introduction

With the ever-growing demand for different information and communication technology services delivered through the cloud, the number of energy-intensive data centers is continuously growing [1]. The data center industry is now estimated to consume 2-3% of worldwide electrical power, with traffic more than doubling every four years [2]. Furthermore,

the typical workload of a data center ranges between 10% and 30% [3, 2].

However, as energy has become more expensive and the market for cloud services has become more competitive, cloud providers are researching and developing energy-saving alternatives for data centers [3]. This significant issue prompts a reconsideration of the methods and research approaches to reduce energy consumption as a major factor in the cloud environment [1]. Although researchers have developed a number of solutions to overcome this challenge, energy efficiency still remains a difficulty.

The data center business has historically mostly addressed energy inefficiencies through hardware innovation, such as more effective processors, better cooling systems, and better power distribution systems. However, there is room to improve efficiency with software solutions as well [2].

The goal of this paper is to explore those software-only solutions that could further improve energy efficiency in the cloud, in particular, through the utilization of containers, their orchestration systems, such as Kubernetes, and scheduling algorithms.

In Section 2, this paper will introduce the reader to relevant technologies, such as containers, orchestration systems, and scheduling algorithms. Section 3 will provide various solutions found in literature to the problems presented in Section 2 about energy efficiency of orchestration systems. Finally, Section 4 will provide an analysis and meaningful discussions on the topic, and Section 5 will present the conclusions.

2 Background

Before discussing the energy saving capabilities of various Kubernetes configurations, it is important to examine the energy efficiency provided by containers and orchestration technologies.

2.1 The advent of Containers

Because of the demand for computing power, agility, and manageability over the past decade, Virtual Machines (VMs) have been widely used in the cloud environment [4]. However, there are various other approaches for running workloads efficiently, such as containers.

Containers are a form of operating system virtualization that allows packaging and isolating applications into a single unit that can run con-

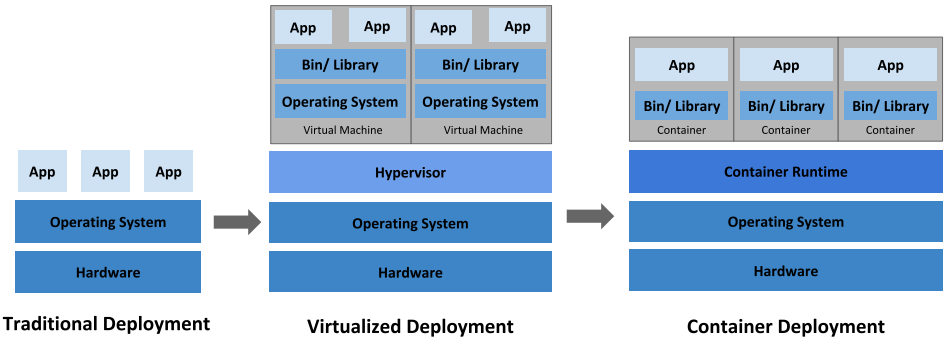


Figure 1. Evolution of deployments, from VMs to containers [6]

sistently on any infrastructure. Unlike traditional virtualization methods, containers offer many important advantages, such as being more portable, quick to boot and efficient – both in terms of resources and costs [3]. Most importantly, containers introduce no form of overhead on their underlying system since they can be executed directly on kernel level, in contrast to VMs which require a hypervisor. In practical terms, when containers are used instead of virtual machines, either more tasks can be executed on the same physical hardware, or the same tasks can be executed with a lower energy consumption. In both cases, energy efficiency can increase up to 21% compared to a traditional VM setup [3].

At the time of writing of this paper, Docker is one of the most well-known and utilized container management system [5].

2.2 Container orchestration with Kubernetes

Most high-level container management systems (also known as container engines) have the capability of managing the full lifecycle of a container, including being able to pull container images from repositories, managing them and running them through lower-level runtimes [7]. However, when scaling a system to cluster levels, managing the lifecycle of containers becomes extremely difficult, especially if the system must consider elastic demand. The solution to this problem is container orchestration, which consists of a smart container management system capable of deploying, scheduling, and networking all containers belonging to the cluster [1, 7]. Currently, the most used container orchestration system is Kubernetes [8].

Kubernetes, also known as k8s, is an open-source system for automating deployment, scaling, and management of containerized applications [9]. In recent years, Kubernetes has become the de-facto standard for con-

tainer orchestration. Therefore, most of the conclusions drawn from this paper can also be applied to other container orchestration systems as well.

2.3 The scheduling problem

The main goal of Kubernetes is managing tasks on containers over a cluster. Clusters are a set of nodes, either physical devices or abstract entities, that can communicate and act together as a single object. Nodes can be vastly different from each other, since Kubernetes is specifically designed to interact with real-life environments, comprised of both homogeneous and heterogeneous systems.

The component of the orchestration systems that assigns tasks to nodes is the scheduler [7]. The role of the scheduler is to handle system resources on the cluster level. It must assign the most suitable task to the most suitable node in each instant of execution, also considering factors such as cluster structure, node affinity (not all nodes can handle all tasks) and other constraints. The role of the scheduler is extremely important since it can influence both the efficiency of the entire cluster and the amount of system resources needed to perform any given tasks. Different approaches to scheduling can help prioritize different targets, such as resource utilization, time requirements, and energy efficiency. So far, no scheduler has proven itself superior to the others. Therefore the scheduling problem still remains unsolved [7].

3 Energy efficiency of schedulers

By default, the Kubernetes scheduler attempts to distribute the workload evenly across all nodes [2, 7]. This is achieved by design, since the default scheduler is meant to be a general-purpose system that accommodates all kinds of different workloads. This behavior, however, is not ideal when accounting for the energy efficiency of the system.

The default scheduler also allows a more fine-grained customization, due to a set of properties that can be assigned to tasks and nodes. One of the most useful properties are *affinity* and *taint* [10], which can be used to respectively increase or decrease the likeness of a task being assigned to a certain node. Due to a third parameter named *tolerance* it is also possible to completely prevent a task from being performed on a node with a taint level superior to the one tolerated by the task, effectively preventing some

tasks from being executed on certain nodes. The goal of these measures is to maximize affinity on all kinds of systems, by assigning tasks to the nodes best suited to perform them correctly.

Although these functionalities alone are often enough to operate most real-life scenarios, the standard Kubernetes scheduler still cannot achieve more complicated situations, such as energy efficiency. For this reason, Kubernetes also supports several ways to extend its scheduling features [7]. To achieve this, the *Kubernetes Scheduling Framework* [11] is the officially recommended solution to extend the standard functionalities of the scheduler with user-created plugins.

With both the default scheduler customizability features and the extension capabilities provided by the scheduling framework, new scheduling algorithms can be implemented. Since the scheduling problem is a hard-NP problem [7, 12], indicating there is no optimal solution for a generic scenario, multiple solutions and approaches are applicable.

3.1 Improving task distribution across nodes

One of the most common strategies to improve the energy consumption of the cluster and, consequently, the overall energy efficiency is to improve the method used by the scheduler to distribute tasks across nodes.

Menouer [13] proposes a method that improves the standard scheduling rules by introducing a multi-criteria decision algorithm. Instead of distributing tasks evenly across the whole cluster, the revised algorithm takes into account multiple factors, such as processor usage, memory usage, storage disk usage, power consumption, and running containers. The system also takes into account if the requested container image is already cached by the node. Every time a new task has to be assigned to a node, the scheduler computes a score for each of the aforementioned parameters for every node, multiplies them by their respective weight, and finally ranks them, assigning the task to the node which ranked the highest. By default, the weight is set equal for all parameters.

The proposed algorithm ranks higher those nodes whose processor, memory, and disk usage is higher than the competing nodes, contrary to how the default scheduler works. This is done in order to keep as many tasks as possible running on the same node, with a technique defined as “compacting”. As shown in the paper, compacting helps mitigating energy consumption and decreases waiting times by fully utilizing fewer nodes, and

therefore processors, instead of partially utilizing all processors, as performed by the standard scheduler.

As shown in by Menouer [13], the proposed scheduling algorithm is successful since it provides both a lower energy consumption and lower waiting times, thus proving itself as computationally and energy efficient. Menouer also claims that the inclusion of the power consumption parameter in the scheduling process alone is enough to improve energy efficiency, since it is never considered by default.

However, one of the major shortcomings of this approach is the complete lack of support for heterogeneous clusters, since all nodes are considered virtually identical to each other.

Kaur et al. [14] propose a similar but different approach. First and foremost, the proposed method takes into account the energy consumption of nodes, just like the previously mentioned one. However, while the first paper designs a specific scheduling algorithm to perform a task distribution across nodes, this paper formulates the scheduling problem as a multi-objective optimization problem, whose goal is to minimize energy consumption while also trying to reduce interference between tasks.

Further differentiating the two approaches, the proposed solution supports both heterogeneous nodes and heterogeneous workloads. To achieve this, the authors proposed a system capable of being configured with different types of tasks (either processor-intensive or network-intensive) as well as various constraints, such as minimum processor and memory resources, job deadlines, and number of replicas. The result is a robust system which is able to operate an enormous number of workloads and cluster configurations, while also providing efficiency. The scheduling system also provides a feature which allows those nodes powered by green energy sources to be prioritized over the others in order to provide a more environmentally friendly approach.

Just like the previous solution, in order to optimize power consumption and efficiency, tasks are compacted onto fewer nodes. However, because the system also support heterogeneous workloads, a more advanced scheduling solution is used, which will try to group similar tasks onto similar nodes.

According to the authors of the paper, the proposed scheduling solution can help increasing energy efficiency by up to 14%, while also reducing interference by 47% and the carbon footprint of the cluster by 31%.

3.2 Limiting performance to increase efficiency

While the previously presented algorithms try to achieve energy efficiency by manipulating the way tasks are assigned to nodes, Rocha et al. [15] take a fundamentally different approach.

The proposed scheduling system offers the client the capability to specify the maximum acceptable energy-performance ratio, also known as trade-off. When a new task will be queued for execution in the cluster, the proposed scheduler will first check which node is capable of executing the task, also excluding those nodes which are incompatible or too busy with other tasks. The scheduler will then try to estimate the energy efficiency of the new task before execution starts. This estimation is performed thanks to pre-computed references as well as similar tasks already running in the system. Finally, the scheduler will compute an efficiency score for every capable node and will assign the task to the node whose score is closest to the requested trade-off previously set by the user.

With this method, the scheduler effectively excludes some nodes from the active ones because they were not considered energy efficient enough to perform the required task, therefore drastically improving energy efficiency. However, this method also drastically reduces overall performances, since high-performance nodes are more likely to draw more power and therefore are also more likely to be excluded. This performance drop, however, is often acceptable, especially for deadline-free and low-priority tasks. Those workloads do not require a fast execution or a real-time response form the system, therefore achieving energy savings is the preferable alternative.

Rocha et al. [15] claim an average energy saving of 1.5% and consequently a total efficiency improvement of 7.1% while running the proposed scheduler against the default one on a realistic test environment. The drawback, however, is the penalty in performance, which can only be considered acceptable when running deadline-free tasks.

3.3 Reducing energy consumption by predicting performance

The scheduler proposed in [15] is peculiar because it can achieve energy savings by predicting workloads and assigning tasks.

Another similar paper on the topic is [2], which presents an improved scheduler that can allocate tasks more efficiently by using experimental data gathered previously. The authors collected all required data through

experiments conducted on the selected hardware in a wind tunnel, and then designed a custom scheduler to accommodate their needs.

The scheduler showed vast improvements in both energy consumption and energy efficiency, while also providing good overall performances. However, the proposed approach requires advanced research and can only be applied to a well-defined hardware. Furthermore, those systems that improve energy efficiency by predicting workloads attributes are not guaranteed to work as expected for real-time applications or I/O bound applications (those applications dependent on external factors). For these reasons, the solution presented cannot be easily achievable unless for very specific hardware configurations and workloads.

4 Analysis and considerations

Overall, all presented methods seem to provide clever solutions that manage to improve energy efficiency by only changing the behavior of the scheduler. In particular, the methods proposed by Menouer [13] for homogeneous clusters and Kaur et al. [14] for heterogeneous clusters seem particularly promising because of their simplicity of implementation and use, while also remaining effective. The method proposed by Rocha et al. [15] also seems effective and efficient, although it requires a more advanced setup and a deeper knowledge of the underlying heterogeneous system. The scheduler proposed by Townend et al. [2], however, requires a more advanced physical setup as well as advanced equipment to be performed correctly, therefore it might be the least widely applicable method of the analyzed ones, although still effective.

5 Conclusion

In conclusion, the demand for cloud services has led to an increase in energy-intensive data centers, which account for a significant amount of worldwide electrical power consumption. While the data center industry has historically focused on hardware innovation to address energy inefficiencies, cloud providers are now exploring energy-saving software solutions. Containers and container orchestration systems, such as Kubernetes, have emerged as efficient alternatives to traditional virtualization methods. This paper has explored software-only solutions to improve en-

ergy efficiency in the cloud, specifically through scheduling algorithms. Although some effective methods to improve energy efficiency have already been developed, the scheduling problem remains unsolved, and no scheduler has proven itself superior to others. Overall, energy efficiency in the cloud remains an ongoing challenge, and continued research is needed to further reduce energy consumption.

References

[1] N. Gholipour, E. Arianyan, and R. Buyya, "Recent advances in energy-efficient resource management techniques in cloud computing environments," *New Frontiers in Cloud Computing and Internet of Things*, pp. 31–68, 2012.

[2] P. Townend, S. Clement, D. Burdett, R. Yang, J. Shaw, B. Slater, and J. Xu, "Improving data center efficiency through holistic scheduling in kubernetes," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 156–15610, IEEE, 2019.

[3] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: a survey on software technologies," *Cluster Computing*, pp. 1–31, 2022.

[4] V. Dakic, M. Kovac, and J. Redzepagic, "Optimizing kubernetes performance, efficiency and energy footprint in heterogenous hpc environments.," *Annals of DAAAM & Proceedings*, vol. 10, no. 2, 2021.

[5] "Docker website." `https://www.docker.com/`.

[6] "Kubernetes documentation." `https://kubernetes.io/docs/concepts/overview/`.

[7] C. Carrión, "Kubernetes scheduling: Taxonomy, ongoing issues and challenges," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–37, 2022.

[8] M. Coté, "Why large organizations trust kubernetes." `https://tanzu.vmware.com/content/blog/why-large-organizations-trust-kubernetes`.

[9] "Kubernetes website." `https://kubernetes.io/`.

[10] "Kubernetes documentation - taints and tolerations." `https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/`.

[11] "Github scheduling framework repository." `https://github.com/kubernetes/enhancements/tree/master/keps/sig-scheduling/624-scheduling-framework`.

[12] Z. Rejiba and J. Chamanara, "Custom scheduling in kubernetes: A survey on common problems and solution approaches," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–37, 2022.

[13] T. Menouer, "Kcss: Kubernetes container scheduling strategy," *The Journal of Supercomputing*, vol. 77, no. 5, pp. 4267–4293, 2021.

[14] K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, and M. Atiquzzaman, "Keids: Kubernetes-based energy and interference driven scheduler for industrial

iot in edge-cloud ecosystem," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4228–4237, 2019.

[15] I. Rocha, C. Göttel, P. Felber, M. Pasin, R. Rouvoy, and V. Schiavoni, "Heats: Heterogeneity-and energy-aware task-based scheduling," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 400–405, IEEE, 2019.

The Password Conundrum: Rethinking Authentication for the Digital Age

Anoosha Sajid
anoosha.sajid@aalto.fi

Advisor: Sanna Suoranta

Aalto University

Abstract

This paper examines the usability of passwords as a primary means of authentication and explores alternative forms of authentication that can improve usability. It highlights the difficulties faced by users in remembering complex and unique passwords, the impracticality of complying with rigid standards for robust passwords, and the complexity of creating and managing passwords. The paper also discusses other types of authentication as possible methods to increase the security of authentication systems without negatively impacting their usability.

1 Introduction

Passwords have been the primary means of authentication for decades, but their inefficiency has become increasingly apparent in the present world. The widespread use of the Internet has led to a surge in online security threats, and passwords are often the weakest link in the security chain. With the increasing number of data breaches, it is clear that traditional passwords are no longer sufficient to protect sensitive information, as

cybercriminals have developed sophisticated methods to steal passwords and render them ineffective. To mitigate these risks, there is a pressing need to enhance password authentication systems, to ensure that they are secure, yet still usable for the end-user.

The usability of passwords constitutes a vital facet of their overall security, as it influences the ease with which users are able to retain and input their passwords without compromising security. According to Whitten and Tygar [19], password usability is "the degree to which a password system is easy to learn, efficient to use, easy to remember, and satisfying to the user." However, the utilization of passwords as the predominant mode of authentication has been hampered by the difficulties that users experience in generating and retaining complex and unique passwords. This leads to the widespread usage of readily guessable or repeated passwords, as seen by the findings of a SplashData survey done in 2019, which revealed that the most often used passwords in the year were "123456" and "password." [16]. Additionally, the recurrent imposition of password changes can also generate confusion and frustration, leading to forgotten passwords. Despite these issues, passwords remain a widely used form of authentication due to their ease of use, low cost, and widespread availability. [13]

The purpose of this paper is to examine the interdependence of password usability and security, and to offer suggestions for enhancing both. Section 2 discusses the implications of password usability on security and the importance of preserving both usability and security. Section 3 examines various methods of improving password usability while maintaining security, including the use of password managers, alternative authentication techniques, and the impact of user education and industry standards. The paper concludes with a summary of the findings and suggestions for future advancements in password usability.

2 Challenges to Password Usability

Passwords, despite being a widely used form of authentication, have been increasingly criticized for their poor usability. Bergstrom et al. [5] found that the linguistic and phonological difficulty of passwords greatly impacts the usability of these authentication methods. This issue is exacerbated by the increasing need for users to enter complex passwords on mobile

devices, as highlighted by Greene et al. [7]. This can lead to difficulties in typing and, ultimately, a decrease in the overall usability of passwords.

Greene et al. [8] also highlighted the issue of password security being compromised by poor usability in their study on the usability and security of peruted passwords on mobile platforms. They found that users often choose easily guessable or reused passwords, which can have a significant impact on security, as easily guessable or reused passwords can be easily cracked by attackers, potentially leading to unauthorized access to sensitive information.

2.1 The trade-offs between security and usability in password design

The design of passwords is a delicate balance between ensuring security and promoting usability. Research in the field of information security has long highlighted the trade-offs between these two goals. As stated by Sasse [14], "usability and trust are often seen as conflicting goals in information systems." This is because security measures, such as requiring complex and unique passwords, can make it difficult for users to effectively recall and apply them, ultimately resulting in poor password management practices, such as writing them down or reusing the same password across multiple accounts. On the other hand, longer passphrases that are easier to remember can be more secure, but users may still make errors that compromise their effectiveness. Therefore, when designing password systems, it is crucial to balance both security and usability considerations to ensure that they are efficient and effective in promoting strong security practices among users.

3 Improving usability of passwords

To address the security and usability issues associated with passwords, researchers and developers have proposed various solutions. These solutions include the use of password managers and alternative authentication methods, such as token-based authentication, as well as both physiological and behavioral biometrics. Password managers provide a convenient solution for managing complex and unique passwords for multiple online accounts. They generate strong passwords, store them securely, and aut-

ofill them when needed, thus mitigating the risk of password reuse and theft. Alternative authentication methods can provide additional layers of security when used in conjunction with passwords, such as two-factor authentication. While they are not inherently more secure than passwords, they can address certain security risks associated with passwords and may offer a more seamless user experience. However, it is important to note that these methods also have their own limitations and potential vulnerabilities that must be considered when implementing them in a security system.

This section will examine the potential of password managers and alternative authentication methods to enhance the usability of passwords. The advantages and limitations of these approaches will be explored, along with recommendations for their effective implementation. Furthermore, the importance of user-centered design and human factors in the implementation of these solutions will be discussed to ensure their usability and effectiveness.

3.1 The use of password managers

Password managers have become increasingly popular due to their many advantages. By reducing the risk of password guessing and brute-force attacks, password managers provide a higher level of security for online accounts. This is accomplished by using complex algorithms, such as cryptographic hash functions, random number generators, and symmetric-key encryption, to generate strong and unique passwords that are difficult to crack. By doing so, password managers provide a higher level of security for online accounts.

Florencio and Herley [6] highlight the significance of password managers in enhancing the security of online accounts by generating and storing strong passwords in an encrypted format. This encryption method ensures that even if an attacker gains access to the password database, they cannot read the passwords without the encryption key, which is typically stored locally on the user's device or in a secure cloud-based vault. Although different password managers store passwords in various locations, such as a user's device, the cloud, or across multiple devices, they all aim to securely store passwords to minimize the risk of data breaches. Ultimately, password managers offer a convenient and secure way to generate and store strong passwords, providing an extra layer of protection to online

accounts and reducing the risk of unauthorized access.

Furthermore, as highlighted by Oorschot et al. [18], password managers simplify the login process by filling in the username and password fields automatically, alleviating the burden of memorizing or typing in passwords. By doing so, password managers save users time and effort while reducing the likelihood of errors. Additionally, password managers can be used across multiple devices, which is particularly advantageous for those who utilize several devices.

Additionally, password managers can prevent man-in-the-middle attacks by providing login credentials only for recognized websites based on their URL or domain, which reduces the risk of credential theft. Password managers also enhance password usability by generating strong passwords and storing them securely. This allows users to avoid the burden of creating and recalling passwords for multiple accounts, which in turn lowers the likelihood of insecure password practices such as using easily guessable passwords or reusing the same password across multiple accounts.

Despite their many advantages, password managers are not without drawbacks. One of the primary concerns is the risk of a single point of failure, which arises if the database of the password manager is compromised, thereby putting all user accounts at risk. Furthermore, the security measures implemented by password managers, such as encryption and protocols, can also be vulnerable to attack. Additionally, factors such as human error, the use of weak master passwords, or concerns about privacy can impede users from adopting password managers.

Hence, as with any security measure, password managers are not infallible, and their effectiveness ultimately depends on how they are used. Users should be cautious about selecting a reliable and trustworthy password manager, keeping the master password secure, and regularly updating and backing up their password database. By using password managers carefully, users can enjoy the many benefits they offer while minimizing the risks associated with password-based authentication.

3.2 The use of alternative authentication methods

The use of alternative authentication methods has gained significant attention in recent years as a way to improve password usability. Token-based authentication, as well as both physiological and behavioral biometrics, are among the most commonly used alternative authentication methods.

Token-based authentication is a passwordless authentication method that utilizes a token or code, generated by a third-party service, to authenticate a user's identity. This technique is gaining popularity as an alternative authentication method for improving password usability because it eliminates the need for users to remember and manage passwords. In their study, Banerjee and Hasan [4] evaluated the effectiveness of token-based authentication techniques on open-source cloud platforms. The study found that token-based authentication techniques are more secure than traditional password-based authentication, as the token or code is only valid for a short period, reducing the risk of unauthorized access. Moreover, the study found that token-based authentication is easier to implement and manage than traditional authentication methods, which can reduce the burden on the user and the system administrator. The results suggest that token-based authentication can be an effective alternative authentication method to enhance password usability and security, particularly in cloud-based environments. However, it should be noted that the distribution of tokens can present security challenges, as the tokens must be distributed in a secure manner to prevent interception or theft, which could undermine the security of the authentication process. Therefore, it is imperative to implement appropriate security measures to ensure the safe distribution and management of tokens.

Physiological biometrics, which uses physical or behavioral characteristics of a person to authenticate their identity, is a promising alternative authentication method for improving password usability. One such biometric authentication method is FingerPIN, which utilizes fingerprints to authenticate a user's identity. The usability of FingerPIN was evaluated in a study conducted by Marasco et al. [10], which found that FingerPIN was perceived as easy to use and efficient by users, compared to traditional password-based authentication. The study also found that FingerPIN was effective in reducing the risk of unauthorized access, as it is difficult

for an attacker to replicate a person's fingerprints. The results suggest that biometric multi-factor authentication, such as FingerPIN, can be an effective solution to improve password usability and security, particularly in situations where the user is required to access sensitive information or perform high-value transactions.

Behavioral biometrics is a relatively new authentication method that utilizes the unique behavioral patterns of the user, such as keystroke dynamics and touch screen interactions, to authenticate the user. This technique provides several advantages over traditional authentication methods, including the elimination of the need for the user to remember and manage passwords, which can be challenging for many users. Moreover, behavioral biometrics is more secure because it is difficult for attackers to replicate or steal a person's unique behavioral patterns. According to a survey by Stylios et al. [17], continuous user authentication based on behavioral biometrics, such as touch dynamics and accelerometer data, on mobile devices shows promise for improving password usability and security while providing a better user experience compared to traditional password-based authentication.

Consequently, authentication methods such as token-based authentication, as well as both physiological and behavioral biometrics are promising ways to improve password usability and security. However, they may not provide sufficient security on their own. Multi-factor authentication (MFA) is necessary to combine two or more authentication factors to ensure the highest level of security. MFA can provide an additional layer of security to protect against various types of attacks and unauthorized access attempts. Therefore, it is crucial to consider implementing MFA in combination with alternative authentication methods to ensure the best possible security while maintaining easy usability for users.

3.3 The role of user education and standards

Enhancing and improving user education as well as the rigorous adherence of industry standards are vital for establishing appropriate password habits among users. According to the study by Adams and Sasse [1], inadequate policies, rather than user carelessness, are the main cause of insecure password practices. Therefore, it is essential to educate users on password management policies to improve password usability.

One way to improve password usability is to adopt standardized password policies that are user-friendly and easy to understand. A study by AlFayyadh et al. [2] found that standardized password policies can significantly improve the usability of password management systems by providing clear guidelines and reducing the burden of remembering complex passwords. These policies can include guidelines for creating strong passwords, using unique passwords for each account, and performing regular password updates. By providing users with clear and easy-to-follow guidelines, organizations can promote good password practices and improve password usability.

Government and industry standards can also play a role in improving password usability. For example, the General Data Protection Regulation (GDPR) of the European Union mandates the use of strong passwords and password protection measures to ensure the confidentiality and integrity of personal data. In the United States, the National Institute of Standards and Technology (NIST) has published guidelines on password security that include recommendations on password complexity and length, as well as restrictions on password reuse and expiration [11]. By adhering to these guidelines, organizations can ensure that their password policies are user-friendly and easy to follow.

4 The future of password usability and security

With the ongoing advancement of technology, the emergence of novel password technologies is expected to transform both the usability and security of passwords in the foreseeable future. One such technology is passwordless authentication, which eliminates the need for users to remember and enter complex passwords by using alternative factors, such as biometric authentication or hardware tokens, to authenticate users. Parmar et al. [12] conducted a comprehensive study on passwordless authentication, which evaluated the effectiveness and usability of various passwordless authentication methods. Their study contributes to the understanding and development of passwordless authentication, which is expected to become more widely adopted in the future.

Another emerging technology is the use of artificial intelligence (AI)

to enhance password security. AI can analyze user behavior and detect anomalies that may indicate a security breach. For example, a study by Azanguezet Quimatio et al. [3] proposed an ensemble learning approach for user authentication through keystroke dynamics or mouse movements. This approach is different from keylogger attacks, which record the keys pressed by a user to steal their password. While keylogger attacks can capture passwords regardless of the complexity of the password, analyzing keystroke patterns adds an extra layer of security and usability, as it is less intrusive and does not require users to change their behavior. Additionally, AI can also be used to generate strong passwords that are difficult for attackers to guess or crack.

Blockchain technology is also being explored as a potential solution for password security. Blockchain technology provides a decentralized and secure platform for storing and sharing user credentials, eliminating the need for passwords. In a blockchain-based authentication system, user credentials are stored on a distributed ledger, and access is granted through cryptographic protocols. This technology has the potential to provide a more secure and transparent authentication system. To get access, users would need to use their private key to verify their identity and grant permission to access their credentials on the blockchain. [15].

In addition to these emerging technologies, there are also innovative password mechanisms being developed that could revolutionize password security in the future. According to Gui et al. [9], researchers are exploring the use of electroencephalogram (EEG) signals to authenticate users based on their brainwaves, providing a secure and easy-to-use alternative to traditional passwords. This technology has the potential to revolutionize password security in the future, making it more convenient and reliable for users to access their accounts while also reducing the risk of security breaches.

Overall, the future of password usability and security is likely to involve a combination of emerging technologies and innovative password mechanisms. Passwordless authentication, AI-based security, blockchain-based authentication, and brainwave authentication are just a few examples of the potential solutions that may become widely adopted in the future. As technology continues to evolve, it is essential to evaluate the security and

usability of these solutions carefully to ensure that they provide users with the highest level of protection.

5 Conclusion

In conclusion, passwords remain the most widely used form of authentication, but their usability and security have increasingly become a challenge in the digital age. This paper has explored the difficulties faced by users in generating and retaining complex and unique passwords, and the negative impact on security when users choose weak and easily guessable passwords. We have also discussed the trade-offs between security and usability when designing password systems and highlighted the need to strike a balance between the two.

To improve the usability of passwords while maintaining security, we have explored the use of password managers and alternative authentication methods such as biometrics and token-based authentication. We have also emphasized the importance of user-centered design and human factors in the implementation of these solutions to ensure their usability and effectiveness.

Furthermore, the role of user education and industry standards in improving password usability and security cannot be overlooked. Educating users on the importance of strong passwords, the risks of weak passwords, and best practices in password management can go a long way in mitigating the risks associated with passwords. Industry standards can also help promote consistent and effective password policies across different platforms and organizations.

In summary, the password conundrum requires a rethinking of authentication for the digital age. While passwords remain relevant, there is a need to explore and implement alternative authentication methods and user-centered solutions that can enhance password usability and security. The way forward requires collaboration between users, developers, industry leaders, and policymakers to ensure a secure and user-friendly authentication system for the digital age.

References

[1] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] Bander AlFayyadh, Per Thorsheim, Audun Jøsang, and Henning Klevjer. Improving usability of password management with standardized password policies. *arXiv preprint arXiv:1910.10941*, 2019.

[3] Benoît Martin Azanguezet Quimatio, Olive Flore Yatio Njike, and Marcellin Nkenlifack. User authentication through keystroke dynamics based on ensemble learning approach. *Journal of Computer and Communications*, 8(12):43–53, 2020.

[4] Amit Banerjee and Mahamudul Hasan. Token-based authentication techniques on open source cloud platforms. *International Journal of Computer Applications*, 181(30):11–14, 2018.

[5] J. R. Bergstrom, S. A. Frisch, D. C. Hawkings, J. Hackenbracht, K. K. Greene, M. Thefanos, and B. Griepentrog. Development of a scale to assess the linguistic and phonological difficulty of passwords. In *Cross-Cultural Design, 6th International Conference, CCD 2014, Held as Part of HCI International 2014*, 2014.

[6] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, 2014.

[7] K. K. Greene, M. A. Gallagher, B. C. Staton, and P. Y. Lee. I can't type that! p@$$w0rd entry on mobile devices. In *Human Aspects of Information Security, Privacy, and Trust, Second International Conference, HAS 2014*, 2014.

[8] K. K. Greene, J. Kelsey, and J. M. Franklin. Measuring the usability and security of perturbed passwords on mobile platforms. Technical Report 8040, NIST, 2016.

[9] Qiong Gui, Zhanpeng Jin, and Wenyao Xu. Exploring eeg-based biometrics for user identification and authentication. *IEEE Access*, 6:44605–44618, 2018.

[10] Emanuela Marasco, Massimiliano Albanese, Venkata Vamsi Ram Patibandla, Anudeep Vurity, and Sumanth Sai Sriram. Biometric multi-factor authentication: On the usability of the fingerpin scheme. *Pervasive and Mobile Computing*, 58:1–11, 2019.

[11] National Institute of Standards and Technology. Digital identity guidelines: Authentication and lifecycle management. Technical Report Special Publication 800-63B, National Institute of Standards and Technology, 2017.

[12] Viral Parmar, Harshal A. Sanghvi, Riki H. Patel, and Abhijit S. Pandya. A comprehensive study on passwordless authentication. In *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, pages 1266–1275, Erode, India, 2022. IEEE.

[13] Robert W Proctor. Improving computer security for authentication of users: Influence of proactive password restrictions. *Memory & Cognition*, 34(3):573–582, 2006.

[14] M Angela Sasse. Usability and trust in information systems. *University College London*, 1999.

[15] Yashvardhan Singh, Sakshi Jain, and Shubham Rawal. Blockchain based password free authentication. *International Journal of Scientific and Research Publications (IJSRP)*, 11:418–419, Apr 2021.

[16] SplashData, Inc. Splashdata's "worst passwords list" for 2019. https://www.splashdata.com/press/worst-passwords-of-2019-report, 2019.

[17] Ioannis Stylios, Spyros Kokolakis, Olga Thanou, and Sotirios Chatzis. Behavioral biometrics & continuous user authentication on mobile devices: A survey. *Journal of Ambient Intelligence and Humanized Computing*, 11(10):4471–4495, 2020.

[18] Paul C. van Oorschot. Computer security and the internet: Tools and jewels from malware to bitcoin. In *Computer Security and the Internet*, chapter 3.6. John Wiley & Sons, 2 edition, 2022.

[19] Alan Whitten and J D Tygar. Guerrilla hci: using off-the-shelf user-centered design. *Human-computer interaction*, 14(1-2):63–97, 1999.

A Security Overview of OAuth 2.0

Apramey Bhat
apramey.bhat@aalto.fi

Tutor: Aleksi Peltonen

Abstract

OAuth 2.0 is an open-standard protocol widely adopted by organizations. It enables secure access to resources stored on one website from another, without exposing the user credentials. While initially intended for authorization, OAuth 2.0 is also increasingly used for authentication. In fact, a study found that 77% of users view social login as a good registration solution. This highlights the growing trend towards the use of OAuth 2.0 and social login to facilitate user authentication and improving user experience on the web. However, the widespread use of the framework also makes it a popular target for attackers. This paper reviews the security features, common security vulnerabilities, and best practices for securing OAuth 2.0-based applications.

KEYWORDS: OAuth 2.0, Authorization, Authentication, Session integrity, Token

1 Introduction

OAuth 2.0 gained immense popularity as an authorization and authentication protocol in web and mobile applications. The growing adoption of OAuth 2.0 made it integral to many online services, including social

media, email, and e-commerce platforms [1]. Although OAuth 2.0 has become a widespread solution for secure access, it has faced criticism and concerns regarding its security and privacy vulnerabilities. Various researchers have analyzed the protocol and several security weaknesses have been identified [2, 3, 4, 5]. These shortcomings have the potential to put sensitive information at risk and leave organizations and individuals exposed to malicious attacks.

Despite established guidelines and best practices aimed at securing OAuth 2.0 applications, current implementations remain vulnerable to attack. These vulnerabilities are exemplified by the attack [6] on Booking.com, an online travel agency with over 100 million users. In addition, a study conducted by Arshad et al. on the "Alexa top 50k websites" [7] revealed that nearly 36% of the sites using OAuth 2.0 are vulnerable to cross-site request forgery (CSRF) attacks [4, 5]. These findings emphasize the urgency for continued research and development efforts to improve the security of OAuth 2.0 implementations. This paper provides an overview of OAuth 2.0 protocol and summarizes its security aspects.

The rest of the paper is structured as follows: Section 2 summarizes the technical details of OAuth 2.0 and explains how it enhances application security. Section 3 discusses the fundamental security concepts that underlie OAuth 2.0. Section 4 examines various types of attacks that can exploit vulnerabilities in OAuth 2.0 implementations and provides guidance on how to mitigate these risks. Section 5 presents a brief discussion about the paper. Finally, Section 6 summarizes the key findings and highlights the importance of following the strict guidelines and best practices provided by the OAuth 2.0 community to ensure the secure implementation of the protocol.

2 Background

2.1 OAuth 2.0 Protocol Overview

OAuth 2.0 is a widely used authorization framework for securing access to Application Programming Interfaces (API) [8]. It enables third-party applications to access the user's resources, such as their data, without requiring the user to share their login credentials. The OAuth 2.0 protocol was developed by the Internet Engineering Task Force (IETF) in 2012 as

an upgrade to the earlier OAuth 1.0 protocol.

The OAuth 2.0 framework involves four roles [9]: (1) The *Resource Owner* owns the resource and can grant access to it through the Authorization Server (AS). In the context of OAuth 2.0, this is typically the end-user who possesses the resources that a third-party service desires to access. (2) The *Client* is an application that communicates with the AS to receive an access token, that can be used to access the user's protected resources on the resource server. (3) The *Resource Server (RS)* stores the user's resources. It is responsible for validating the client provided access token and granting access to the requested resources if the token is valid. (4) The *Authorization Server (AS)* grants access tokens to clients after the resource owner grants authorization. The AS is responsible for verifying the identity of the client and the resource owner, and to ensure if the client has appropriate permissions to access the requested resources.

OAuth 2.0 protocol provides four types of grant, each with unique characteristics and recommended use cases [9]. (1) The *Authorization Code Grant* is a server-to-server exchange where an authorization code is exchanged for an access token, providing secure access to resources without sharing the user's credentials with the client application. (2) The *Implicit Grant* involves a client receiving an access token directly from the AS. The OAuth 2.0 best security practices [3] recommends against using Implicit grant type because it involves transmitting the access token directly to User Agent (UA) or browser that can be easily intercepted and misused [10]. (3) The *Resource Owner Password Credentials Grant* requires the user to provide their credentials to the client application to exchange for an access token, and is only recommended for highly trusted clients. (4) The *Client Credentials Grant* involves the client applications providing their credentials to the AS to receive an access token. It is typically used for machine-to-machine communication.

Figure. 1 depicts a typical OAuth 2.0 protocol flow [9, 11] between the above mentioned roles. The numbers below denote the order of the messages exchanged between the roles. (1) The client initiates the process by sending an authorization request to the resource owner. (2) The resource owner then generates an authorization grant as a code and sends it back to the client. (3) After receiving the authorization grant, the client authenticates to the AS and presents the code to initiate an access token request. (4) The AS then validates the authorization grant and issues an access token to the client. (5) The client redeems the access token to

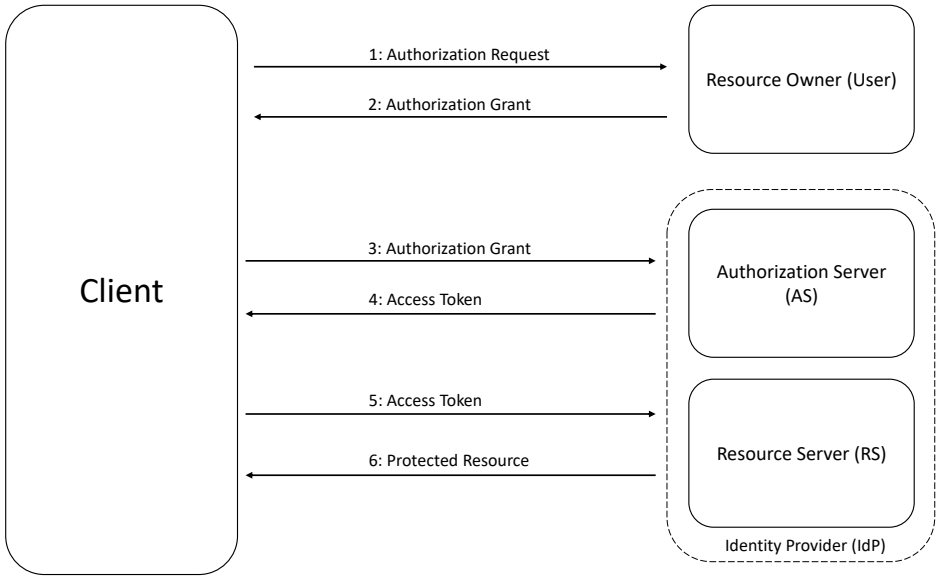


Figure 1. OAuth 2.0 Protocol Flow [11]

the RS to access the protected resource. (6) Finally, the RS validates the access token and sends the requested resource.

2.2 OAuth 2.0 Application Security

One of the most common approaches legacy applications can handle authentication is through a username and password prompt. In this method, the user enters their password, and the application exchanges it for a session cookie [12]. This works well for simple systems. However, it becomes limiting when creating multiple apps that share a user database with a single sign-on.

Without OAuth 2.0, the applications would have to collect the user password and send it to the API. This is not a good idea because it presents a set of risks. For instance, the user cannot trust the application that asks for a password, whether it is a legitimate or third-party app. For third-party apps, this is a severe problem as the app can read any files in the account, change the password, and much more [10].

OAuth 2.0 requires applications to redirect users to the AS for login, thereby avoiding the application's access to the user credentials. This enhances security against untrusted third-party applications, and enables first-party applications to become more flexible. For example, implementing multi-factor authentication at the identity provider (IdP) can improve system security and increase flexibility for updates without requiring changes to individual applications [3, 9, 10]. In summary, OAuth 2.0 improves ap-

plication security by reducing the instances where users enter credentials and enabling centralized control over authorization and authentication mechanisms.

3 OAuth 2.0 Security Concepts

3.1 Front Channel and Back Channel

In OAuth 2.0, there are two communication channels between the different roles involved in the protocol: the front and the back channel [10]. The front channel is an unauthenticated communication channel typically uses the user agent to interact with the AS and the client. It involves user-facing interactions, such as granting permissions and redirection between various web pages [13]. The back channel is a more secure communication channel that is used between the AS and the client. This channel is used for exchanging tokens and other sensitive information. The back channel is secured using Transport Layer Security (TLS) encryption and requires authentication from both the AS and the client [13]. By separating these channels, OAuth 2.0 can ensure that sensitive information, such as access tokens is not leaked to unauthorized parties.

3.2 OAuth 2.0 Security Parameters

OAuth 2.0 protocol uses various security parameters to ensure secure communication between the authorization server, the resource server and the client [3, 10, 13].

The IdP assigns unique *Client ID* and *Client Secret* to a client application, which are primary security parameters in OAuth 2.0. These parameters are essential for authenticating the client application and allowing it to access protected resources on behalf of the user. In certain instances, IdPs may choose to omit the provision of a client secret to clients with a higher likelihood of exposing this sensitive parameter due to their underlying architecture, such as Single Page Applications (SPA) and native mobile apps.

The authorization server redirects the resource owner to *Redirect URI* after the user has granted permission for the client to access their resources. The Redirect URI is used to receive the authorization code or access token from the AS. The Client must provide the precise URI or

URI pattern that AS will utilize to redirect back to the client. This parameter holds significant importance as it is the only reliable means by which the AS can ensure that the code or token is being transmitted to a legitimate source in a client with no client secret [3].

The *State* parameter is a random value generated by the client and included in the authorization request. The AS returns this value along with the authorization code or access token. The client should verify that the state value returned by the AS matches the one that is sent in the authorization request to prevent CSRF attacks [13].

The *Scope* parameter is used by the client to request access to specific resources or actions on behalf of a user. The scope can specify a set of permissions or access rights, such as read-only access to user data or the ability to post on behalf of a user.

3.3 Proof Key for Code Exchange

Proof Key for Code Exchange (PKCE) is an OAuth 2.0 protocol extension that enhances the security of authorization code flows [14]. It ensures that only intended client can obtain an access token, even when an attacker intercepts the authorization code. PKCE introduces a *code verifier* and a *code challenge* in the authorization process.

When a client initiates an authorization request, it generates a random code verifier and computes a hash called code challenge based on the verifier. The code challenge is sent to the AS with the authorization request. The AS then stores the code challenge and generates an authorization code that is sent back to the client. The client redeems the authorization code back to the AS, and the code verifier it initially generated. The AS verifies that the hash of the incoming code verifier matches the code challenge it initially received, and if it does, it issues an access token to the client [3, 15].

4 Attacks and Mitigations

OAuth 2.0 is the most reliable option of other frameworks that provide the same functionality, yet there are several open issues that can make it vulnerable to attacks. To address these issues, the IETF has made efforts to educate the developer community on best practices for implementing OAuth 2.0 securely [3, 13]. The following section discusses some of the

attacks associated with these vulnerabilities and their mitigation strategies.

4.1 Insufficient Redirect URI and State Validation

As discussed in the Section 3.2, particular authorization servers enable clients to provide redirect URI patterns rather than specific redirect URIs. This parameter holds a significant role in client validation at the AS endpoint, especially for public client with no client secret. While this approach allows clients to add a range of URI patterns, it is vulnerable to various types of impersonation and CSRF attacks, making it error-prone [13]. For example, a URI pattern *https://*.someclient.com/* can be replaced by an attacker as *https://evil-server.com/.someclient.com/* in a way to bypass the AS validation, resulting in getting the authorization code or token to attacker's domain. Even if validations are handled correctly, this vulnerability can exist due to subdomain takeover attack [16]. An experiment by Arshad et al. on CSRF attacks due to the improper redirect URI claims that around 36% of the sites are vulnerable to CSRF attacks. The experiment also claims that around 13% of the sites have empty state parameters, 27% of sites have state parameters but with improper validation, and 5% of the sites use same state for every request [5].

Mitigation: The intricacy involved in the correct implementation and management of pattern matching of the URI raises few security concerns. It is recommended to simplify the logic and configuration required by employing exact redirect URI matching [3]. PKCE, as discussed in the Section 3.3 can be used as a countermeasure for CSRF attacks [14].

4.2 Authorization Code Injection

Attackers can gain unauthorized access to protected resources by stealing the authorization code intended for a legitimate client during the authorization process [3]. he attacker can use various means to intercept the authorization code, including network eavesdropping, Cross-Site Scripting (XSS), and Man-in-the-Middle (MitM) attacks. Once the attacker has the stolen authorization code, they can inject it into their own session with the client. This enables the attacker to obtain victim's access token, which can be used to access the victim's protected resources. The attacker can then read, modify, or delete sensitive information, potentially compromis-

ing the victim's data security [13].

Mitigation: Implementing measures, such as HTTPS to encrypt communications, PKCE or nonces to prevent code interception, and client authentication to prevent unauthorized clients from accessing the authorization server [3].

4.3 PKCE Downgrade Attack

An attacker intercepts the initial authorization request and replaces it with a request that does not include the PKCE parameter i.e., code challenge. The attacker then uses the resulting authorization code to obtain an access token and potentially gain unauthorized access to protected resources [3, 13]. The PKCE downgrade attack can be carried out through various means, such as intercepting and modifying the initial authorization request, exploiting vulnerabilities in the client or authorization server, or performing a MitM attack.

Mitigation: Authorization servers should enforce the use of PKCE for all authorization requests and ensure that clients and authorization servers are configured to reject requests that do not include PKCE parameters [15]. Additionally, clients should use secure methods to generate code verifiers and code challenges, such as cryptographic libraries, and validate the PKCE parameters during the token exchange.

5 Discussion

The security of a system depends on three key factors: authorization, authentication, and session integrity. Authorization controls access to protected resources, authentication verifies the user's identity, and session integrity ensures that protected resources are accessed correctly [2]. Although OAuth 2.0 is widely used for secure resource access, there is no centralized body for testing authorization and authentication schemes of the framework. Developers typically conduct their own pen-testing on their implementation of the OAuth 2.0 services [17]. OAuth's popularity stems from its interoperability across platforms and programming languages, unlike rival solutions that are site-specific, such as Yahoo BBAuth and Facebook Auth. The flexibility of OAuth 2.0 has led to misinterpretations and criticisms from the security community [12]. Balancing security and usability is a challenge, especially as demand for user-friendly solu-

tions increases.

6 Conclusion

This paper provides an overview of the OAuth 2.0 security model and a few attacks on OAuth 2.0 implementations. It is important to note that there are many other types of attacks that could occur, such as IdP Mix-Up attacks, Authentication code injection, Credential leakage through browser history, Open redirection, and many more [3, 13]. Ultimately, the security of OAuth 2.0 implementations depends on the developer team behind the client application and their adherence to following the strict guidelines and best practices provided by the community.

The OAuth 2.0 framework offers comprehensive documentation [10], RFCs [9, 3, 13], and security profiles, such as Financial-grade API (FAPI), Pushed Authorization Requests (PAR), Demonstration of Proof of Possession (DPoP), and many more [10] to help developers implement the protocol securely. However, the abundance of documentation may also be overwhelming, leading to confusion and potential mistakes. The OAuth community is addressing this concern by releasing OAuth 2.1 [15] with stricter guidelines for implementing OAuth securely. The developers need to be well-equipped with these guidelines and best practices to ensure the security of their implementations.

References

[1] R. Soni, "Social login and CRO: 9 reasons why you need it." https://cxl.com/blog/social-login/, 2022. last accessed: 12/03/2023.

[2] D. Fett, R. Küsters, and G. Schmitz, "A comprehensive formal security analysis of oauth 2.0," vol. 24-28-October-2016, p. 1204 – 1215, 2016.

[3] T. Lodderstedt, J. Bradley, A. Labunets, D. Fett, "OAuth 2.0 Security Best Current Practice." https://www.ietf.org/archive/id/draft-ietf-oauth-security-topics-21.html, 2022. last accessed: 12/03/2023.

[4] E. Arshad, M. Benolli, and B. Crispo, "Practical attacks on Login CSRF in OAuth," *Computers and Security*, vol. 121, 2022.

[5] M. Benolli, S. A. Mirheidari, E. Arshad, and B. Crispo, "The Full Gamut of an Attack: An Empirical Analysis of OAuth CSRF in the Wild," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12756 LNCS, p. 21 – 41, 2021.

[6] A. Carmel, “Traveling with OAuth - account takeover on Booking.com.” https://salt.security/blog/traveling-with-oauth-account-takeover-on-booking-com, 2023. last accessed: 12/03/2023.

[7] “Alexa Top Websites.” https://www.expireddomains.net/alexa-top-websites/, 2022. last accessed: 12/03/2023.

[8] R. Yang, W. Lau, and T. Liu, “Signing into one billion mobile app accounts effortlessly with OAuth 2.0,” 2016.

[9] D. Hardt, “RFC 6749: The OAuth 2.0 Authorization Framework.” https://www.rfc-editor.org/rfc/rfc6749, 2012. last accessed: 06/04/2023.

[10] A. Parecki, “OAuth 2.0.” https://oauth.net/2/, 2022. last accessed: 07/04/2023.

[11] W. Li, C. J. Mitchell, and T. Chen, “Mitigating CSRF attacks on OAuth 2.0 Systems,” 2018.

[12] M. Papathanasaki, L. Maglaras, and N. Ayres, “Modern authentication methods: A comprehensive survey,” *AI, Computer Science and Robotics Technology*, Jun 2022.

[13] T. Lodderstedt, M. McGloin, P. Hunt, “OAuth 2.0 Threat Model and Security Considerations.” https://www.rfc-editor.org/rfc/rfc6819, 2013. last accessed: 09/04/2023.

[14] N. Sakimura, J. Bradley, N. Agarwal, “Proof Key for Code Exchange by OAuth Public Clients.” https://www.rfc-editor.org/rfc/rfc7636.html, 2015. last accessed: 12/03/2023.

[15] D. Hardt, A. Parecki ,T. Lodderstedt, “The OAuth 2.1 Authorization Framework.” https://www.ietf.org/archive/id/draft-ietf-oauth-v2-1-07.html, 2022. last accessed: 09/04/2023.

[16] D. Liu, S. Hao, and H. Wang, “All your DNS records point to us: Understanding the security threats of dangling DNS records,” vol. 24-28-October-2016, p. 1414 – 1425, 2016.

[17] M. Argyriou, N. Dragoni, and A. Spognardi, “Security flows in oauth 2.0 framework: A case study,” pp. 396–406, 09 2017.

A survey on participant selection for mobile crowdsensing

Ashok Dhungana
ashok.dhungana@aalto.fi

Tutor: Aziza Zhanabatyrova

Abstract

The evolution of the internet and the emergence of various sensor devices connected to it have made it possible to collect vast amounts of data for different purposes. Companies and researchers can utilize this data to create products and analyze patterns. With numerous devices already connected to the internet, optimizing the number of participants and balancing the server's load that collects the data is essential[9]. In mobile crowdsensing (MCS), participant selection strategies must be carefully designed to ensure adequate coverage while avoiding unnecessary energy consumption. In this survey, we will examine various techniques developed to optimize participant selection for MCS.

KEYWORDS: Mobile Crowdsensing, Internet of Things, Participant Selection

1 Introduction

Mobile crowdsensing (MCS) is a new paradigm in sensing and data collection that utilizes the power of mobile devices and crowdsourcing to gather data on a large scale. MCS enables individuals to participate in data collection activities using their mobile devices, such as smartphones, wear-

able sensors, and other portable gadgets, and share data with researchers, companies, or public institutions. The resulting data can provide insights into various domains, such as urban planning, healthcare, and environmental monitoring. In MCS, participants can actively contribute to data collection and analysis, allowing for a more comprehensive understanding of the phenomenon being studied. This approach has the potential to revolutionize data collection, making it more efficient, cost-effective, and scalable, and it has already found numerous applications in both academia and industry. This introduction will explore the concept of MCS and the various techniques used to optimize participant selection in this emerging field.

Mobile crowdsensing is a collaborative process in which individuals use their sensing and computing devices, such as smartphones, wearables, in-vehicle sensors, and other Internet of Things (IoT) devices, to share data and extract information for measuring and mapping phenomena of common interest [7]. Smartphones, which are widely available, are equipped with sensors such as GPS, compass, camera, microphone, proximity sensor, light sensor, and inertial sensor. Additionally, wearable devices like smartwatches and rings can sense heart rates, blood pressure, and perform ECGs, among other things. With the availability of such data, numerous applications can be developed.

MCS has numerous applications in environmental, infrastructural, and social domains [7]. For instance, microphones can be utilized with GPS location data to map noise pollution, and inertial sensors in vehicles can be combined with GPS to track potholes. Similarly, pictures of trash taken near water bodies can be used to map pollution levels in water resources. Infrastructural aspects like road conditions, traffic congestion, parking space availability, and traffic light operations can be studied using positioning data [7]. In the social domain, individual data such as exercise data can be beneficial in medical science to create products focused on health.

Despite the increasing benefits of MCS due to advancements in sensors and internet connectivity, resource limitations in terms of energy, bandwidth, and computation pose significant challenges [16, 7]. Dynamic nature of the devices collecting sensor data means that they may share energy sources and bandwidth for various other activities, further complicating the situation. Additionally, the large number of devices connected to servers, which identify, schedule, and communicate with them, further

adds to the complexity [7].

Although some research has proposed different participant selection strategies, there are relatively few works summarizing the current state of algorithms [7, 5]. In this survey, we will compare various works in the literature related to energy-efficient mobile crowdsensing services, focusing on participant selection strategies. The optimizations can include data sensing scheduling, participant selection, and piggybacking [9]. However, our analysis will be limited to works related to participant selection.

2 Background

MCS can be classified into two primary types based on the involvement of participants. The first type is participatory sensing, where users are aware and voluntarily participate in sending out data. The second type is opportunistic sensing, where data is shared without direct involvement from the user, and data is shared in the background [8]. In particular, with opportunistic sensing, the devices sensing the data must decide when and how user participation should take place based on the user's context [8]. However, selecting participants is an optimization challenge even before allocating the sensing task to these devices. In this survey, our focus is on participant selection and not task allocation.

Let's take an example to illustrate the need for participant selection. Suppose we need GPS data to map road congestion, and there are numerous active vehicles on the road that can provide this data. However, under resource constraints, it is not feasible to select all active sensors to send this data to the MCS server. Instead, it would be sufficient to optimally select a subset of participants from various target locations to generate reliable road congestion information [17]. This is because sensing GPS data and sending it consumes a considerable amount of energy, and devices have limited battery capacity [4].

While road congestion information serves as an example, having a participant selection strategy is essential to maximize the output of information, such as Quality of Information(QoI) [15], coverage [1, 9], load balancing [2], and minimize energy consumption [15, 9], in various scenarios. To achieve this goal, extensive research has been conducted in the literature. We will focus on some noteworthy contributions, compare their strategies, output, and effectiveness, and reach a conclusion that directs toward the use of different algorithmic approaches based on the expected data and

information.

3 Challenges in participant selection

- To select participants who are deserving of incentives while adhering to budget constraints, the algorithms for the mobile crowd-sensing system must be implemented at the operation center. This is necessary to avoid participants providing redundant information. Additionally, selecting participants based solely on their local information poses another challenge[12].

- The majority of participant selection strategies focus on the participants' a priori knowledge and their suitability for the task's sensing region[15, 14, 6]. Alternatively, some strategies address energy efficiency by reducing individual sensing frequency or replacing sensors based on energy cost, which may be limited in their application[15, 10, 3].

- To ensure complete coverage while minimizing energy consumption, it is essential to strategize a plan for participant selection[9].

- Sensing tasks performed on smartphones often result in a poor user experience, as they consume limited resources such as energy and cellular bandwidth[2, 13]. To ensure a better user experience, it is recommended to distribute sensing workloads equally among participants.

4 Literature Survey

The literature offers several approaches to tackle the aforementioned challenges in participant selection. While there is no single implementation that addresses all challenges, various works target specific subsets of the challenges to optimize the effectiveness of selection.

The problem of participant selection in MCS systems without redundant information is investigated by Nguyen and Zeadally. Their study focuses on minimizing the number of users while maximizing the number of events sent by users, which is also known as the Participant Report-

Incident Redundant Avoidance (PRIRA) problem. To solve the PRIRA problem, they introduce a new approximation algorithm called the Maximum-Participant-Report Algorithm (MPRA). The authors conduct theoretical analysis and experimentation and show that the proposed method performs well within reasonable bounds of computational complexity[12].

According to Li et al., the behavior of participants in mobile crowdsensing systems is influenced by the incentives offered. To account for this, they suggest that participant selection methods should be varied based on the incentives, and propose a dynamic algorithm for adjusting incentive mechanisms. The authors utilize game theory to estimate the impact of incentives on participant behavior and predict their mobility patterns. Based on this analysis, they propose a distributed approximate algorithm to address the participant selection problem[11].

In their study, Song et al. present a QoI-aware energy-efficient participant selection strategy that includes four key design elements. Firstly, they measure the extent of satisfaction with the task's QoI requirements by analyzing the data granularity and quantity collected by participants. Secondly, they estimate the impact of energy cost on the participant's device's current energy levels using an energy consumption index. Thirdly, they propose a probabilistic movement model to estimate the amount of data collected from participants. Fourthly, they suggest a multi-objective constrained optimization problem for participant selection, where task QoI requirements and energy consumption index of all participants are considered as optimization objectives. They present a suboptimal, easy-to-implement solution for solving this optimization problem[15].

In their work, Ko et al. propose a strategy for selecting participants that ensures coverage and energy efficiency, called CG-EEPS. This strategy utilizes participants' data usage profile and mobility level, and employs a piggyback approach to enable energy-efficient transmissions of sensory data. To achieve optimal performance, a constraint Markov decision process (CMDP) problem is optimized using linear programming to obtain the optimal policy. To overcome the dimensionality issues associated with CMDP, the authors also propose a greedy heuristic, which is evaluated in their study[9].

Ahmed et al. proposes a concept of (α, T)-coverage of the target field where each point in the field is sensed by at least one node with a probability of at least α during the time period T. It aims to achieve (α, T)-coverage by a minimal set of mobile sensor nodes for a given area of in-

terest, coverage ratio α, and time period T. Two algorithms namely *inter-location* and *inter-meeting-time* are proposed to estimate the expected coverage of the specified area of interest for a set of selected nodes. The *inter-location* algorithm selects a minimal number of mobile sensor nodes from nodes inside the area of interest taking into account the distance between them. The *inter-meeting-time* selects nodes taking into account the expected meeting time between the nodes[1].

The authors of [2] propose an approach to optimize the load balancing of resource-constrained individual mobile users. This approach does not prioritize the minimization of overall sensing cost or user utility. Specifically, they formulate the load-balanced mobile user recruitment (LB-MUR) problem as a mixed integer linear programming and prove its NP-hardness. To address this challenge, they propose an efficient polynomial time suboptimal algorithm based on linear programming relaxation. Moreover, they derive the approximation ratio of the proposed algorithm.

5 Performance study

In this section, we will try to list the outputs of the above-mentioned strategies proposed in different papers and articles for participant selection.

5.1 Nguyen and Zeadally's Maximum-Participant-Report Algorithm[12]

The performance of the MPRA algorithm was evaluated through experiments, the results of which are presented in this section. The simulations were conducted using 100-1000 participants and events uniformly distributed in a 100x100 square area, with a coverage range of 15 for every participant. The performance of the MPRA was compared with that of Greedy-based algorithms (GBA) in terms of the number of participants' reports and the number of paid awards. The simulations were repeated 10 times to obtain average results.

In the first experiment, the number of participants was varied from 100 to 1000, while the number of events was fixed at 500. The results showed that the MPRA algorithm achieved a higher number of participants' reports than the GBA and the GBA (d=10). The total number of reports increased sharply with the increase in the number of participants, as

more participants were able to provide reports. Moreover, the number of paid awards in the MPRA was lower than that in the GBA and the GBA (d=10), indicating that the MPRA is more resource-efficient in mobile crowd-sensing applications. The GBA (d=10) had a very high number of paid awards, as it could not select the best participant.

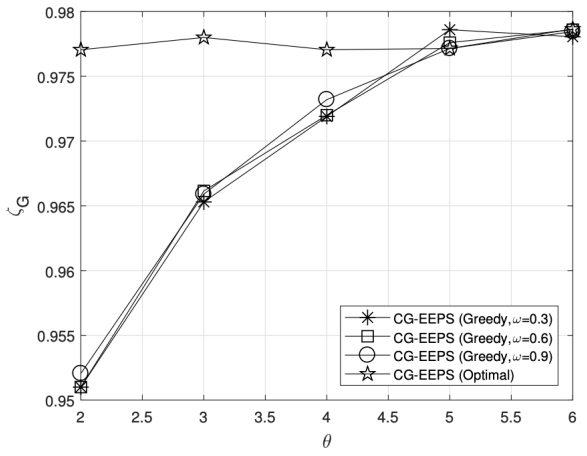
In the second experiment, the number of events was varied from 100 to 1000, while the number of participants was fixed at 100. The MPRA algorithm again outperformed the GBA and the GBA (d=10) in terms of the number of participants' reports and the number of paid awards. The MPRA was able to select the highest number of suitable candidates to provide reports, leading to better performance compared to the GBA and the GBA (d=10). The results showed that the MPRA required a much lower number of paid awards than the GBA and the GBA (d=10), indicating that it is a more efficient algorithm for obtaining more participants' reports and saving resources for good investments. Overall, these experiments demonstrated that the MPRA algorithm is a better solution for mobile crowd-sensing applications than the GBA and the GBA (d=10).

5.2 Ko et al.'s coverage-guaranteed and energyefficient participant selection strategy [9]

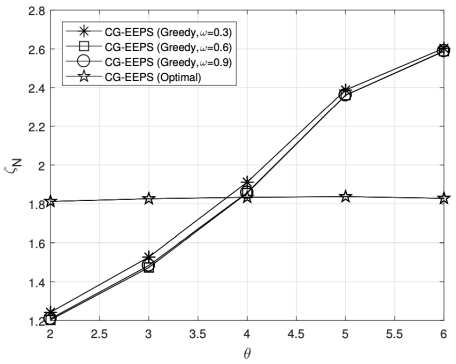
The performance of the greedy CG-EEPS was compared to CG-EEPS with the optimal policy by searching for the optimal solution using brute-force search. The optimal solution of CG-EEPS could only be obtained in a small-scale scenario due to the curse of dimensionality of CMDP and the high complexity of brute-force search. They illustrate the effect of the threshold θ on the average number of MDs in the sensing mode ζ_N and the average coverage rate ζ_G. They showed that ζ_N and ζ_G of the greedy CG-EEPS increased as θ increased. This was due to the fact that as a larger θ was set, the MCS server requested the participation of more MDs, resulting in more MDs being included in the active mobile device by *Algorithm 1*. However, θ did not impact ζ_N and ζ_G of the optimal CG-EEPS because the optimal policy was decided regardless of θ.

They demonstrated that when appropriate ω and θ were set, the greedy CG-EEPS achieved comparable ζ_G and ζ_N to CG-EEPS with the optimal policy. Specifically, based on these results, when $\omega = 0.9$ and $\theta = 4$, the performance difference between CG-EEPS with the optimal policy and the greedy CG-EEPS was less than 5

According to the paper, CG-EEPS selects mobile devices with high mo-



(a)



(b)

Figure 1. Effect of θ & comparison between the optimal solution and the greedy algorithm. (a) Average number of MDs in the sensing mode ζ_N. (b) Average coverage rate ζ_G

bility and data usage frequency as participants to reduce the average number of mobile devices in the sensing mode while achieving a higher coverage rate. Evaluation results demonstrate that CG-EEPS effectively selects only the necessary mobile devices to sense targets. The paper further shows that CG-EEPS reduces the number of mobile devices in the sensing mode by up to 80 percent while still meeting the required coverage guarantee rate. Moreover, the algorithm adapts to changes in the monitoring area and sampling cycle.

5.3 Li et al.'s coverage-guaranteed and energyefficient participant selection strategy [11]

The authors provide a comparison of the TSA and baseline methods based on different values of total incentives and expected values of incentives of subtasks, respectively, under both the performance-centric and energy-efficient incentive mechanisms. The TSA algorithm consistently selects participants with lower total incentives compared to the other three meth-

ods. Although the number of participants selected by the TSA may not always be the lowest among the four methods, it always results in the lowest total incentives. When the total incentives are less than the expected values of incentives of subtasks, the TSA may not select the lowest number of participants. However, when the total incentives exceed the expected values of incentives of subtasks, the TSA can select the fewest participants compared to the other three methods. These findings demonstrate that the TSA can effectively address the original participant selection problem, even when assuming that each participant should be paid equally, by setting the expected value to 0.

The experiments conducted yielded several observations. Firstly, it was found that the TSA algorithm outperformed other baseline methods across two different incentive mechanisms, demonstrating its effectiveness as a participant selection method under various conditions. Secondly, the results showed that the system achieved higher profits and lower energy consumption under performance-centric incentive mechanisms and lower energy consumption with the proposed method, indicating its ability to attain specific system goals under a given incentive mechanism. Lastly, although the TSA method required predicting participants' mobility patterns with the PG game, it took slightly longer to run than the other three greedy algorithms. This demonstrated that the distributed TSA algorithm was practical for real-world MCS systems.

6 Discussion of future prospects

Participant selection is a critical aspect of MCS, as it determines the quality and quantity of data that can be collected. In this context, future prospects in participant selection in MCS are promising and involve various developments that can enhance the efficiency and effectiveness of data collection.

Based on the papers we have discussed, some of the work for the future are stated by different authors. Song et al. have demonstrated the effectiveness and robustness of their approach when compared to existing schemes through extensive experiments based on a real trace in Beijing. They have outlined their plans for future research, which include addressing the multi-task-oriented optimization problem under limited budget constraints for participant selection and motivation, as detailed in [15].

In their upcoming work, the Ko et al. plan to extend the proposed scheme

to identify malicious data and introduce an incentive mechanism to encourage participation in the MCS, as described in [9]. Hassani et al. aims to evaluate their methods further by incorporating real-time sensory data using their task assignment approach. The researchers are interested in expanding the approach to handle periodic sensing tasks that require performance at specific intervals and implementing a mobility model to predict the future location of participants, as proposed in [8].

The results obtained so far have shown the framework's performance in terms of energy/coverage quality and timeliness trade-off. In the future, Bradai et al. plans to adopt regression algorithms to update episodes and progressively enhance the off-line selection's coverage quality, particularly in cases of dissimilar data compared to recorded ones, as explained in [4].

Besides the future efforts hinted at by the surveyed authors, there are other future prospects in participant selection. One of the most significant future prospects in participant selection in MCS is the use of machine learning (ML) algorithms. ML algorithms can analyze large amounts of data and identify patterns that can help predict which participants are most likely to contribute high-quality data. For example, ML algorithms can analyze the GPS data from participants' devices to determine which participants are most likely to be in the right location to provide relevant data.

Another promising development in participant selection in MCS is the use of incentive mechanisms. Incentives can motivate participants to contribute high-quality data by offering them rewards or recognition. For example, participants can receive points or virtual badges for providing accurate and timely data. Incentives can also be customized to the needs of different participants, such as offering monetary rewards to participants with low incomes.

Furthermore, social networks can be utilized to identify and recruit participants in MCS. Social networks can be leveraged to create communities of participants who share common interests or characteristics. These communities can be targeted for specific data collection tasks, such as studying the behavior of people who live in a particular area or have a specific hobby.

In conclusion, the future prospects for participant selection in MCS are promising and involve various developments that can enhance the efficiency and effectiveness of data collection. These developments include

the use of machine learning algorithms, incentive mechanisms, and social networks. With these advancements, MCS can become an even more powerful tool for obtaining valuable data about the environment and human behavior.

7 Conclusion

In conclusion, participant selection in mobile crowdsensing is a critical factor that determines the success or failure of the sensing campaign. The performance of different algorithms and studies has been explored in this field to achieve better participant selection. Machine learning algorithms have shown promising results in predicting the behavior and performance of potential participants. Additionally, studies have highlighted the importance of considering various factors, such as location, task complexity, incentives, and user preferences, when selecting participants. Future research should continue to investigate novel algorithms and strategies to improve participant selection and ensure the quality and reliability of mobile crowdsensing data. By doing so, we can maximize the potential benefits of this innovative sensing paradigm for various applications, ranging from urban planning and transportation to healthcare and environmental monitoring.

References

[1] Asaad Ahmed, Keiichi Yasumoto, Yukiko Yamauchi, and Minoru Ito. Distance and time based node selection for probabilistic coverage in people-centric sensing. In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 134–142, 2011. doi: 10.1109/SAHCN.2011.5984884.

[2] Xin An, Hao Guo, Xiumin Wang, and Xiaoming Chen. Load balanced mobile user recruitment for mobile crowdsensing systems. *IEEE Communications Letters*, 21(11):2420–2423, 2017. doi: 10.1109/LCOMM.2017.2732403.

[3] Patrick Baier, Frank Durr, and Kurt Rothermel. Psense: Reducing energy consumption in public sensing systems. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 136–143. IEEE, 2012.

[4] Salma Bradai, Sofien Khemakhem, and Mohamed Jmaiel. Re-opsec: Real time opportunistic scheduler framework for energy aware mobile crowdsensing. In *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5, 2016. doi: 10.1109/SOFTCOM.2016.7772174.

[5] Andrea Capponi, Claudio Fiandrino, Burak Kantarci, Luca Foschini, Dzmitry Kliazovich, and Pascal Bouvry. A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities. *IEEE communications surveys & tutorials*, 21(3):2419–2465, 2019.

[6] Lingjie Duan, Takeshi Kubo, Kohei Sugiyama, Jianwei Huang, Teruyuki Hasegawa, and Jean Walrand. Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing. In *2012 Proceedings IEEE INFOCOM*, pages 1701–1709. IEEE, 2012.

[7] Raghu K. Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011. doi: 10.1109/MCOM.2011.6069707.

[8] Alireza Hassani, Pari Delir Haghighi, and Prem Prakash Jayaraman. Context-aware recruitment scheme for opportunistic mobile crowdsensing. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, pages 266–273, 2015. doi: 10.1109/ICPADS.2015.41.

[9] Haneul Ko, Sangheon Pack, and Victor C. M. Leung. Coverage-guaranteed and energy-efficient participant selection strategy in mobile crowdsensing. *IEEE Internet of Things Journal*, 6(2):3202–3211, 2019. doi: 10.1109/JIOT.2018.2880463.

[10] Immanuel König, Abdul Qudoos Memon, and Klaus David. Energy consumption of the sensors of smartphones. In *ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems*, pages 1–5. VDE, 2013.

[11] Shu Li, Wei Shen, Muhammad Bilal, Xiaolong Xu, Wanchun Dou, and Nour Moustafa. Fair and size-scalable participant selection framework for large-scale mobile crowdsensing. *Journal of Systems Architecture*, 119:102273, 2021.

[12] Tu N. Nguyen and Sherali Zeadally. Mobile crowd-sensing applications: Data redundancies, challenges, and solutions. *ACM Trans. Internet Technol.*, 22(2), oct 2021. ISSN 1533-5399. doi: 10.1145/3431502. URL https://doi-org.libproxy.aalto.fi/10.1145/3431502.

[13] J Paczkowski. iphone owners would like to replace battery. *All Things Digital*, 21, 2009.

[14] Mehdi Riahi, Thanasis G Papaioannou, Immanuel Trummer, and Karl Aberer. Utility-driven data acquisition in participatory sensing. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 251–262, 2013.

[15] Zheng Song, Bo Zhang, Chi Harold Liu, Athanasios V. Vasilakos, Jian Ma, and Wendong Wang. Qoi-aware energy-efficient participant selection. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 248–256, 2014. doi: 10.1109/SAHCN.2014.6990360.

[16] Leye Wang, Daqing Zhang, Zhixian Yan, Haoyi Xiong, and Bing Xie. effsense: A novel mobile crowd-sensing framework for energy-efficient and cost-effective data uploading. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(12):1549–1563, 2015.

[17] Haoyi Xiong, Daqing Zhang, Leye Wang, and Hakima Chaouchi. Emc3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint. *IEEE Transactions on Mobile Computing*, 14(7):1355–1368, 2015. doi: 10.1109/TMC.2014.2357791.

Kubernetes Cluster Network Model and its Limitations

Atte Rouhe
atte.rouhe@aalto.fi

Tutor: Tuomas Aura

Abstract

Kubernetes is an open-source system for managing a cluster of machines running containerized applications. Kubernetes is widely adopted by large organizations to streamline development of their applications. Networking in Kubernetes is defined by the Kubernetes cluster networking model, which outlines the requirements for how the networking should work in the cluster. Kubernetes itself provides no implementation of the model, and it is up to the developer to choose a solution best suited for their needs. In this paper, we analyze the networking model and the limitations that it imposes.

KEYWORDS: Kubernetes, container network interface, networking, containers, Kubernetes networking model

1 Introduction

Kubernetes is an open-source software project, originating from Google, for the deployment and management of complex cloud applications that consist of containerized microservices [6].

Kubernetes works by managing a cluster for you. A cluster in Kubernetes is a collection of virtual or physical machines called Nodes that

can be used to run container-based services. A Node can host several Pods. A Pod can comprise several containers. The clusters operations are controlled by a special Node or multiple Nodes called the Control Plane, which has several components. [12] Networking inside the cluster is defined by the Kubernetes networking model. The Kubernetes cluster networking model is only a definition of requirements for networking in a cluster. The Container Network Interface (CNI) specification has been chosen for the implementation of the model. The specification is then in turn implemented by third-party developers.

The basic model of networking in Kubernetes is somewhat simple, but the implementations of the model bring in more complexity. In this paper we will first go through the basics of the model, followed by looking into the variations in implementing the model and how they affect the resulting application. We will also be looking at Service Meshes, which provide functionality that extends the networking capabilities by adding an overlaying infrastructure on top of the basic model and its implementation.

2 Microservice architecture

Applications have traditionally been developed with the monolithic approach, in which multiple highly coupled services are bundled together into a single codebase. However, the rise of cloud computing has revealed that monolithic applications are hard to scale in response to increased load, and can lead to coordination problems between development teams.

The microservice approach has emerged as a solution to these challenges, breaking up applications into independent, decoupled services that are responsible for their specific tasks. The microservice architecture has been found to outperform monolithic applications in terms of increasing throughput, decreasing response times, allowing for higher scalability, and reducing the time and effort needed for development and deployment [13].

The microservices naturally need to communicate with each other over a computer network. This network is a standard IP network. However, modern microservices are implemented as virtual computer units such as virtual machines and containers, which do not have direct access to the physical datacenter network. Instead, some kind of virtual network is needed to connect the microservices that belong to the same application. In Kubernetes, the microservices are implemented as Pods. The

cluster network is the virtual network that connects the microservices in the same cluster to each other.

3 Kubernetes Networking Model

The Kubernetes networking model defines the following requirements for networking inside a cluster: containers can communicate with any other containers without Network Address Translation (NAT), A Pod can communicate with any other Pod without NAT, and the IP address of a Pod is the same both inside the Node as well as in the cluster. Additionally, all the agents on a Node, such as system daemons, can communicate with all Pods on that Node. The main advantage of this model is that it allows for Pods to be treated in a similar way that older infrastructures would be utilizing Virtual Machines (VM). Since IP addresses are allocated at the pod level, all containers in a Pod share the same IP and MAC address, it also means that the containers can communicate with each other using the local loopback interface. [3] The Kubernetes networking model only defines the requirements. It is up to the developer to choose how to implement these requirements, since Kubernetes offers no default networking solution.

3.1 Cluster network

Kubernetes cluster networking can be viewed through three distinct scenarios: traffic inside a Pod, traffic between Pods, and traffic entering or leaving the cluster itself.

When setting up a Pod, an infrastructure container is provisioned to it, which sets up the Pod's network namespace and IP address. The infrastructure container uses Docker's bridge networking mode, meaning that it creates a virtual Ethernet bridge attaching to the node's network interface. All other containers in the Pod then join the network and namespace of this container using Dockers container networking mode. In the resulting configuration, all the containers share the same IP address and can communicate with each other on the localhost. It is possible for the containers to have port collisions with each other in this scenario. To avoid tight coupling between microservices, each Pod usually implements only one microservice.

All Pods in the cluster can communicate with each other directly with-

out the use of NAT. Pods can either communicate straight with other Pods by discovering the Pod's IP address or by using a Service. The problem with communicating directly with Pods is that, as characteristic to Kubernetes, Pods come and go, meaning that their IP addresses might need to be rediscovered constantly. A Service in Kubernetes provides a stable, discoverable Virtual IP address that hides behind it a set of Pods, which might be destroyed and created frequently. The Service operates as a sort of load balancer, providing a stable interface for the set of Pods.

Endpoints for traffic coming in to the Kubernetes cluster can be configured using Kubernetes objects such as NodePort, LoadBalancer, ExternalName and Ingress. These objects allow for exposing parts of the cluster for external traffic. All of these objects operate at the TCP or UDP level except for the Ingress service, which operates at the HTTP level. [12]

3.2 Container Network Interfaces

The Container Network Interface (CNI) specification was initially proposed by CoreOS and has since been adopted by Kubernetes as the model for container networking. At its core, the CNI specification defines a set of interfaces that enable network interaction. Third-party vendors can then implement the CNI specification in their own ways, creating CNI plugins. These plugins then handle tasks such as of IP address management, Pod network namespace and host network connections, IP address allocation and route configuration in the cluster. Each CNI implementation differs in their design and scope of implemented features. CNI plugins differ in their way of utilizing the link-layer for communications in addition to using either an underlay or an overlay network. Additionally, some CNI plugins implement Kubernetes network policies while some do not.

CNI plugins mainly utilize layer 3 (the networking layer) of the networking stack for communications between Nodes in the cluster. Some plugins also utilize layer 2 (the link layer) for communications inside a Node. Plugins utilizing only the second layer also exist, such as [7], but their main point is usually to support legacy applications.

Another major design difference in CNI plugins is their packet forwarding configuration. CNIs take two different approaches to implementing packet forwarding: overlay networking and underlay networking. Overlay networking refers to the situation where a virtual network is provisioned on top of the physical or virtual network in which the hosts reside. Another option is the underlay network, which simply means utilizing the

existing layer 3 infrastructure for routing.

As mentioned earlier, all CNIs do not support Network Policies, for example the Flannel CNI does not implement the Network Policy Controller. Other CNIs vary in their implementation of the controller. Some CNIs implement just the standard Kubernetes Network Policy, which operates at layer 3 and 4, while others extend the functionality to configure policies at levels 3 to 7, or to integrate service meshes. [11]

3.3 Namespaces and Network Policies

As required by the Kubernetes cluster network model, all Pods can reach all other Pods in the cluster by default. This is not an ideal situation for a production cluster, since a situation where all Pods need to reach each other is unlikely.

Namespaces [1] in Kubernetes allow you to separate a set of resources in the cluster to a separate group called a Namespace. The main usage of a Namespace is to divide a single cluster for multiple teams working on different microservices. Namespaces can be used to further enhance security in a cluster by providing increased isolation and additional option in controlling access to parts of the cluster.

Network Policies [2] in Kubernetes are used to configure how a Pods can communicate with other Pods and services in the cluster. They can be applied both to ingress and egress traffic at the IP address and port level. When assessing whether a connection is to be allowed, three different aspects must be inspected: what Pod labels, Namespaces and IP address ranges are accepted. Exceptions to these rules are that connections from the underlying Node to the Pod cannot be blocked by IP address, and the containers in the same Pod can always access each other. An important thing to keep in mind is that support for the network policies comes from the CNI plugin, meaning that the plugin must implement necessary features to allow for this functionality. In addition, network policies in Kubernetes are quite limited as they do not support for example any TLS functionality or logging of network security events.

3.4 Service Mesh

While microservice based architectures have streamlined the deployment of applications, they have also introduced new issues. The problem with modern cloud based microservice applications is that they consist of a high

number of services with constantly changing states, possibly spread over multiple domains. This makes the applications hard to debug and the data flow difficult to track. Service mesh aims to answer these issues. A service mesh is an infrastructure layer above the microservices of an application. Its main purpose is to facilitate fault tolerant service to service communications over complex topologies of modern infrastructures. Service meshes provide a lot of the features that are missing from a standard Kubernetes setup, such as service discovery, fault tolerance, traffic monitoring, load balancing, circuit breaking, and authentication and access control. [9]

Previously, a set of components and frameworks was needed to provide the functionality listed above. This introduced a lot of coupling, since certain frameworks only supported some components, and were tied to specific languages. Additionally, the application itself could be coupled to the specific frameworks selected. With a service mesh, all the functionality happens at the network level, meaning that the individual services of the application do not need to be concerned with how the service mesh operates. [8]

A service mesh comprises two main components or layers: the data plane and the control plane. The data plane consists of interconnected proxies that control inter-service communication and provide features such as load balancing, authentication, and authorization. Each service instance has a side-car proxy that enforces security by injecting policies from the control plane, allowing for dynamic policy changes without modifying the microservice's code. The control plane is responsible for implementing all security functions and contains the necessary intelligence, data, and artifacts such as authentication certificates, authentication policies, authorization engine, and monitoring data. It is separate from the orchestrator's control plane, which controls the cluster. The Service Mesh control plane must be integrated with the orchestration platform and have the necessary integration capabilities to be effective. Additionally, it must be highly available and distributed since it is a critical component of the Service Mesh. [8]

The three main challenges with implementing functioning service meshes are that they need to be high performance, adaptable, and highly available to be production ready for modern applications. [9]

4 Limitations of the Cluster Kubernetes Networking Model

Since the Kubernetes cluster networking model is implemented by the CNI plugin, most of the limitations come from the designs of the implementations themselves.

4.1 Performance

The performance of the cluster networking depends on the CNI implementation. The cluster networking model aims to hide the implementation details from the application developer, and thus the developer cannot have much influence on the performance beyond selecting the CNI.

According to [11], the ideal CNI plugin should implement an eBPF based solution for intra-Node communications, as they found that it performs better for package forwarding inside a single Node. Furthermore, native IP routing should be preferred over an overlay network, since it reduces the amount of overhead relating to package encapsulation and decapsulation. However, if a native option is not viable, then the CNI should be able to provide overlay tunneling options to accommodate for a lover overhead. An eBPF based implementation for iptables will also provide a significantly better throughput [4].

4.2 Security

Due to the flat networking model in Kubernetes, where all containers can reach each other, a compromised container can potentially give an attacker access to the entire cluster, highlighting the importance of implementing robust security measures. Another thing to keep in mind is that CNI plugins run as privileged programs on the Node meaning that compromising a CNI plugin results in the attacker having access to the whole cluster network [10].

Network Policies can be used to limit ingress and egress connections between Pods, and they can be a useful tool to limit an attacker that has for example compromised a single container in the cluster. However possible security issues have been identified with Network Policies, for example configuration mistakes are common. Since Kubernetes does not provide warnings on disabled policies, it is up to the administrator to make sure that the policies are valid, enabled and following best practices such as least privilege and zero trust. [5]

The choice of a CNI plugin affects what level of security you can achieve in your application, since they vary in their implementation designs and scope of features. The CNI may for example provide a more advanced interface for the network security configuration, allowing for more sophisticated traffic control and isolation mechanisms than what could be achieved with a basic solution. This is usually achieved by utilizing overlay networking, which refers to when a virtual network is provisioned on top of the underlying physical hardware. This allows Kubernetes administrators to implement a wide range of network policies and controls to ensure the security and reliability of their applications. [11]

Service meshes can be a great solution to further solidifying the security of your application. As mentioned earlier, they provide a wide variety of functionality to further increase the security of an application.

5 Discussions

The Kubernetes networking landscape is currently quite fragmented, with multiple CNI plugins offering varying implementations of the Kubernetes cluster networking model. The choice of CNI plugin is an important one, as it affects the performance, scalability, and security of the application. It remains to be seen whether the industry will converge on a single best approach or whether multiple options will continue to coexist. One potential solution to the challenges of Kubernetes networking is the use of service meshes. Service meshes provide a range of features such as service discovery, fault tolerance, traffic monitoring, load balancing, circuit breaking, and authentication and access control. As modern cloud-based microservice applications become increasingly complex, service meshes may become an essential tool for managing their networking.

6 Conclusion

In this paper, we analyzed the Kubernetes cluster networking model and its implementation by various CNI plugins. We explored the limitations of the model, particularly in terms of performance and security. We also looked at service meshes as a potential solution to these challenges. The choice of CNI plugin is critical to the performance, scalability, and security of a Kubernetes application. Service meshes can provide additional

functionality and capability, making them a potentially valuable tool for managing complex microservice architectures. Overall, it is clear that networking is a crucial aspect of Kubernetes that requires careful consideration and planning. As Kubernetes continues to gain popularity, it will be important for developers and administrators to stay informed about the latest developments and best practices in networking to ensure the success of their applications.

References

[1] The Kubernetes Authors. Namespaces. https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/, 2022. [Online; accessed 01-April-2023].

[2] The Kubernetes Authors. Network policies. https://kubernetes.io/docs/concepts/services-networking/network-policies/, 2022. [Online; accessed 03-April-2023].

[3] The Kubernetes Authors. Services, load balancing, and networking. https://kubernetes.io/docs/concepts/services-networking#the-kubernetes-network-model, 2022. [Online; accessed 07-April-2023].

[4] Matteo Bertrone, Sebastiano Miano, Fulvio Risso, and Massimo Tumolo. Accelerating Linux security with eBPF iptables. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 108–110, 2018.

[5] Gerald Budigiri, Christoph Baumann, Jan Tobias Mühlberg, Eddy Truyen, and Wouter Joosen. Network policies in Kubernetes: Performance evaluation and security analysis. In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 407–412. IEEE, 2021.

[6] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, Omega, and Kubernetes. *Communications of the ACM*, 59(5):50–57, 2016.

[7] Haines Chan. Anchor. https://github.com/hainesc/anchor, 2018. [Online; accessed 02-April-2023].

[8] Ramaswamy Chandramouli, Zack Butcher, et al. Building secure microservices-based applications using service-mesh architecture. *NIST Special Publication*, 800:204A, 2020.

[9] Wubin Li, Yves Lemieux, Jing Gao, Zhuofeng Zhao, and Yanbo Han. Service mesh: Challenges, state of the art, and future research opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 122–1225. IEEE, 2019.

[10] Francesco Minna, Agathe Blaise, Filippo Rebecchi, Balakrishnan Chandrasekaran, and Fabio Massacci. Understanding the security implications of Kubernetes networking. *IEEE Security & Privacy*, 19(05):46–56, 2021.

[11] Shixiong Qi, Sameer G Kulkarni, and KK Ramakrishnan. Assessing container network interface plugins: Functionality, performance, and scalability. *IEEE Transactions on Network and Service Management*, 18(1):656–671, 2020.

[12] Gigi Sayfan. *Mastering Kubernetes*. Packt Publishing Ltd, 2017.

[13] Vindeep Singh and Sateesh K Peddoju. Container-based microservice architecture for cloud applications. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 847–852. IEEE, 2017.

Modern Applications of Software Reliability Growth Models

Başak Amasya
basak.amasya@aalto.fi

Tutor: Stanislav Chren

Abstract

Software reliability is integral for software systems especially in the modern setting and software reliability growth models are useful tools to evaluate software reliability. However, traditional SRGMs, which were developed in the context of legacy software, do not accurately represent the realities of contemporary software projects. Therefore, recent studies propose new SRGMs that utilize revised assumptions for the OSS context. This paper provides a literature review on these recent SRGMs by focusing on their parameters, assumptions and evaluations.

KEYWORDS: Software reliability growth models, open-source software, agile, literature review

1 Introduction

Software reliability has always been a crucial aspect of software systems but even more so now in the modern context, especially with the increasing usage of agile software development and open-source software (OSS). Software reliability is described in [8] as "probability of failure-free operation" in a certain time and certain environment. Software reliability growth models (SRGMs) are among integral tools used to asses software reliability. In Mičko et al. [7], reliability growth process refers to the practice where the recorded faults are removed from the software system. Accordingly, SRGMs are regression models used to estimate the cumula-

tive number of faults detected with respect to the given time [7].

These models can be used by both the customer and the developer as information they present can support the software release planning [6]. SRGMs can be utilized to forecast metrics such as future failure intensity, number of faults left, or the amount of testing necessary to reach a particular reliability level [7]. Moreover, SRGMs can be employed for test resource and cost allocation and management [15].

There is a gap between software reliability models which were developed in the context of legacy software and waterfall mindset, and OSS and agile mindset. It is critical to understand this gap to better evaluate software reliability in agile development and OSS contexts and also to choose an appropriate model. There exists some work on new SRGM suggestions [3], [5], [8], [9], [11], [12], [15]. However, there is still a need for a thorough evaluation of newly proposed models so that software reliability can be assessed more precisely. Figuring out especially the limitations, impacts and shortcomings of these recent models when applied to software projects in modern contexts will guide practitioners to assess reliability of software products more competently.

The goal of this paper is to explore recent SRGMs, focusing on SRGM parameters, assumptions, and evaluations. This review contributes by comparing and discussing different recent SRGMs, examining the advancements in the most recent research. Thus, it can be helpful to see the latest developments in SRGMs. There exists some literature review on SRGMs, but its focus is mostly on parametric vs non-parametric models, hence the focal point of the study is different [6]. Another study evaluated different traditional SRGMs in terms of their fitting and prediction capabilities based on closed and open-source software. The primary focus is to assess specific models' competency rather than investigating the assumptions or constraints of SRGMs when it comes to modern software projects [10].

This paper is organized as follows. Section 2 explains the details of software reliability growth models. Section 3 provides the results of the study based on the research questions provided while Section 4 discusses these results. Section 5 concludes the paper with some final remarks.

2 Software Reliability Growth Models

Software reliability model is defined as a method to assess and predict software reliability based on particular assumptions in a specific environ-

ment. More precisely, software reliability growth models are regression-based models [13]. These models utilize data collected from the software development and testing process to estimate future reliability of the software.

2.1 SRGM Application

There are different variables to take into consideration while using SRGMs such as amount of testing, defect data, grouped data, growth model type, statistical technique etc. [6].

To apply any SRGM, firstly input data needs to be prepared [1]. According to [6], two pieces of data must be provided: defect data and the time at which the defects first appeared in the data. Chren et al. [1] states that the bug reports can be utilized for this purpose. The bug reports should ideally be evaluated to determine the faults that led to the failures but this can be time consuming. Instead, the duplicates can be eliminated from the bug report to speed up the procedure. The reports can also be further reviewed to ensure that they closely match SRGM's assumptions.

It is important to note that, if there is no significant reliability growth trend in the input data, SRGMs may not be able to provide sensible results [7]. Accordingly, a statistical test evaluation should be carried out to see if a trend of decreasing number of failures becomes apparent [1].

After that, since SRGMs are regression models, fitting them to the data requires estimating their parameters. The most common techniques are maximum likelihood technique and least squares technique [14].

A goodness-of-fit (GoF) test needs to be carried out following parameter estimation to determine how well the model fits the data. Most common metrics used for GoF are R-squared (R^2), Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) and Residual Standard Error (RSE). There is no one model that would match all projects [1]. Most suitable model is selected according to the metrics [7].

These application stages are illustrated and summarized in Figure 1.

Figure 1. SRGM application process [1]

2.2 Traditional SRGM Assumptions

Traditional SRGMs, developed in the context of legacy software, involve assumptions of both testing and defect repair processes. Main assumptions of traditional models and why they might not be valid are explained below [14]:

- Although a perfect debugging process is presumed, meaning that any discovered defect will be corrected right away, this is not always the case. Especially in the OSS setting, debugging process is rather non-linear [7]. Furthermore, unfixed defects can make it more difficult to find further defects in the software.
- It is assumed that "defect repair is perfect". However, in practice, defect repair itself might also result in the introduction of new flaws into the system.
- It is assumed that the code being tested does not change during the testing process. However, new code may always be added for both maintenance and adding new functionality.
- Although, traditional models assume that defects can only be reported by the testing team, they can be reported by other teams as well. However, testing time of the quality assurance team is not nearly identical to the testing time of other groups. If the bugs reported by the other groups are not included at all, this crucial information will be lost. This indicates that defects and test time do not always correlate precisely.
- It is assumed that the each unit time such as calendar time or test cases is equivalent during testing process. For instance, tests that are repeated are less likely to discover new defects but each test is still treated as if they have equal impact.
- Testing is thought to indicate an operational profile. This operational profile usually includes types of users, frequency and timing of user interactions, types of input data etc. In practice, it may be impossible to duplicate the operational environment.
- The failures are thought to be independent from one another. This might especially be an erroneous assumption, if there is a part of the code that has not been tested sufficiently which can also lead to cascading failures.

It's vital to keep in mind that, it is challenging to foresee what kind of

an impact will violation of these assumptions have on the models [14].

These assumptions of traditional SRGMs are based on closed-source software and waterfall development approach, and they stand as a challenge in applications of SRGMs to OSS [7].

3 Results

In recent years, studies have been conducted for development of new SRGMs. However, there seems to be not sufficient overview of these recent models. Accordingly, this review was concentrated on studies from last 10 years. First, a keyword based search was done on Google Scholar and IEEE Xplore. The keywords were: software reliability growth models, agile, open-source software. Then, the found papers' references were examined. Altogether, 30 publications were reviewed and 7 of them were chosen for this paper. These papers are chosen because they were suggesting a new model and not focusing on analysing previously proposed models.

Research questions presented in this section are:

1. What are the recent SRGMs?
2. What are the parameters of SRGMs and how are they estimated?
3. What are the assumptions of SRGMs?
4. How are SRGMs applied and evaluated?

These questions were chosen to better understand and compare recent developments of SRGMs in modern context. This review aims to examine the parameters and assumptions used in these models. The findings of this literature review will be discussed in each subsection provided based on these research questions.

3.1 What are the recent SRGMs?

Rawat et al. [8] propose a new model combining Goel and Weibull distributions through the non-homogenous Poisson process (NHPP). They emphasize the frequent incremental release property of agile software development and emphasize that each release has the potential to introduce new bugs to the system, thus affecting the reliability.

Wang [11] questions the old models' "perfect debugging" assumption which does not represent the reality of OSS debugging process. Wang

presents a reliability model based on these non-linear changes of fault detection.

In another study, Wang et al. [12] put more emphasis on "gradual decrease in the number of introduced faults over time". The new model proposed is based on the decline variation of fault introduction.

Zhang et al. [15] also consider imperfect debugging from fault detection rate perspective and propose flexible SRGMs based on imperfect debugging models.

Saraf et al. [9] explore imperfect debugging and multi-release modelling for OSS, proposing a new SRGM for estimating reliability of multi-release OSS.

He [3] proposes a new model based on newly created faults brought about by fault debugging and its incompleteness, revising the presumptions of NHPP. In this model, quantity of faults and the fault removal efficient alters over time.

Authors of [5] propose a two-dimensional multi-release SRGM, modeling the software reliability growth as a function of both testing time and the number of failures instead of only number of failures, focusing on the faults of each release but does not account for imperfect debugging.

3.2 What are the parameters of SRGMs and how are they estimated?

The mean value functions of investigated SRGMs can be seen in Table 1.

Reference	Year	Mean Value Function
Rawat et al. [8]	2017	$m_n(t) = \lambda a F_{n1}(t) + (1-\lambda) a F_{n2}(t)$
Wang [11]	2021	$\mu(t) = \frac{a}{\theta+e^{\omega t}} \left(e^{\left(\beta t^d + \omega t\right)} - \beta d \sum_{i=0}^{n} \frac{\omega^i t^{i+d}}{i!(i+d)} - 1 \right)$
Wang et al. [12]	2022	$\xi(t) = \frac{\eta\left(1 - \frac{\exp(-\mu t)}{1+\mu t}\right) + C(1-\exp(-\psi t))}{1+\gamma\exp(-\psi t)} - \frac{\eta\mu\exp(-\psi t)\sum_{j=0}^{m}\sum_{i=0}^{n}\frac{(-1)^j(j+2)\mu^j(\psi-\mu)^i t^{i+j+1}}{(i+j+1)i!}}{1+\gamma\exp(-\psi t)}$
Zhang et al. [15]	2022	$m_{IDII\_5}(t) = a\int_0^t \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)u}} e^{-\int_0^u p(1-r(\tau))\frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)\tau}}d\tau} du$
He [3]	2013	$m(t) = b\left\{\frac{k(1+\lambda b)}{\lambda-b} + \frac{\alpha}{b+\lambda}\left[\frac{(1+\lambda t)^2}{2\lambda} - \frac{(1+\lambda t)^{1-\frac{b}{\lambda}}-1}{\lambda-b}\right]\right\}$
Saraf et al. [9]	2022	$M_i(t) = \begin{cases} \frac{\eta_i + \left(\eta^*_{(i-1)2} - M(t_{i-1})\right)}{(1-\delta_{i1})}\left[1-\left(1-\left(F_{i1}\otimes G_{i1}\right)(t)\right)^{\rho_{i1}}\left(1-\delta_{i1}\right)\right]; & \text{for } t \le \lambda_i \\ \frac{\eta_i + \left(\eta^*_{(i-1)2} - M(t_{i-1})\right)}{(1-\delta_{i2})}\left[1-\left(1-\left(F_{i1}\otimes G_{i1}\right)(\lambda_i)\right)^{\rho_{i1}(1-\delta_{i1})}\right. & \left.\left(\frac{(1-(F_{i2}\otimes G_{i2})(t))}{1-(F_{i2}\otimes G_{i2})(\lambda_i)}\right)^{\rho_{i2}(1-\delta_{i2})}\right] \\ + \left(\frac{(\delta_{i1}-\delta_{i2})}{(1-\delta_{i2})}\right) M(\lambda_i); & \text{for } t < \lambda_i \end{cases}$
Kapur et al. [5]	2012	$m(s,u) = \frac{a\left(1-\exp\left(-bs^{\alpha}u^{1-\alpha}\right)\right)}{1+\beta\exp\left(-bs^{\alpha}u^{1-\alpha}\right)}$

Table 1. Recent SRGMs with the publishing year and their mean value functions

Expected cumulative number of detected faults are measured in [11], [12] and [3]. Rawat et al. [8] calculates total number of faults, Zhang et al. [15] calculates specific cumulative fault detection function whereas Saraf et al. [9] calculates the expected amount of faults eliminated in an

interval. Lastly, Kapur et al. [5] is interested in the cumulative number of faults removed.

Failure rate of n-th release (λ), probability density function for permanent and transient faults in the n-th release (F_{n1} and F_{n2}) are most notable parameters in [8]. Scale parameter (β in [11], μ in [12]), shape parameter (d in [11] and λ in [12]) and fault detection rate (w in [11], ψ in [12]) are among the noticeable parameters. Initial count of faults (η), Steiltjes convolution ($\otimes$) according to change point (λ) are utilized in [9]. Fault detection rate per remaining fault (b) and Logistic learning factor (β) are worth mentioning for [5].

An important remark about parameters of SRGMs is that, with the increase in number of random factors in software testing, the SRGMs will also become increasingly complicated and difficult to solve [15].

Among the studies which explicitly mention how they estimated the parameters, least square estimation (LSE) is used in all of them [11], [12], [9], [5]. In contrast, He [3], Zhang et al. [15], and Rawat et al. [8] did not explicitly mention the method used to estimate the parameters. He [3] just states that a method can be chosen such as LSE, Maximum Likelihood Estimation (MLE) or Bayesian Estimation (BE) and yet it is not indicated what was chosen for the proposed model.

In both studies [11] and [12], a sensitivity analysis of the parameters is conducted as well to further investigate impact of the parameters.

Saraf et al. [9] and Kapur et al. [5] also specifies the usage non-linear regression module of SPSS (Statistical Package for Social Sciences) as their tool to estimate unknown parameters. To determine the least square estimates, this module utilizes an iterative estimating approach based on sequential quadratic programming.

3.3 What are the assumptions of SRGMs?

While developing new SRGMs, each study proposed their own assumptions explicitly by revising the assumptions of traditional models, especially putting emphasis on fault detection and removal processes of OSS.

Except Kapur et al. [5], all papers assume an "imperfect debugging" process in contrast to assumptions of traditional SRGMs'. They explain this either by stating that there is a non-linear change of number of detected faults during the development and testing of OSS [11] or emphasizing introduction of new faults during removal process [12] or defining an imperfect fault removal operation [15], [3].

All papers also pay attention to changing fault rates over time. Rawat et al. [8] state that failure rates have exponential and Weibull distributions. Saraf et al. [9] acknowledge that there might be variations in the defect detection rate that are not monotonic. He [3] states that fault debugging rate alters over time. Kapur et al. [5] assume a non-decreasing fault detection rate. Wang et al. [12] states a gradual decrease in fault introduction due to the learning phenomenon for debuggers.

Zhang et al. [15] adds that introduction of new faults during repair is proportional to the accumulated fixed faults. Assuming that at most one fault happens during a specified period of time, they construct a proportionate link between detected and undiscovered faults for building two imperfect debugging models.

Almost all papers consider that fault detection procedure follows NHPP in OSS [11], [12], [15], [9], [3], [5]. Wang et al. [12] and Kapur et al. [5] also presume immediate removal of faults.

In contrast, there exists some different assumptions in some papers. Saraf et al. [9] incorporate the latency between the notice of a failure and the correction of the underlying fault. For the mean value function for the next release, only the existing release and the previous one are taking into consideration.

He [3] modified some assumptions of the classical Goel-Okumoto [2] model as all faults are independent and equally identifiable and operational profiles for software and reliability tests are identical.

Lastly, Kapur et al. [5] utilizes the assumptions from their previous work [4] about two dimensional flexible SRGMs. Most notable ones are no introduction of new faults and all remaining faults equally affect the failure rate. They also consider the faults of the current and the remaining faults of the previous release.

3.4 How are SRGMs applied and evaluated?

Comparison metrics chosen for the models and data they have been tested on can be seen in Table 2.

Among the studies, mean squared error (MSE) is the most commonly used metric, followed by R^2. Bias and root mean square error (RMSE) are also chosen frequently. Many metrics are used together to better evaluate the models in these papers.

Wang [11] states that eight models were utilized to compare the performance of the model, these models included perfect debugging, imper-

Reference	Data	Comparison Metrics
Rawat et al. [8]	four consecutive releases	Reliability (probability/quantity of faults)
Wang [11]	three OSS fault data sets	MSE, R^2, RMSE, TS, Bias, Variance, RMSPE and KD
Wang et al. [12]	two OSS fault data sets	MSE, R^2, TS, and Bias (fitting), PSSE, TS, Variance, and Bias (predicting)
Zhang et al. [15]	three failure data sets	MSE, Variance, RMS-PE, BMMRE and R^2
He [3]	not stated	SSE and R^2
Saraf et al. [9]	three releases	MSE and Bias
Kapur et al. [5]	four releases	MSE

Table 2. Recent SRGMs data for evaluation and comparison metrics

fect debugging, closed-source software, OSS models. Wang claims that this new model has superior fitting and predictive performance for OSS, closed-source software, perfect and imperfect debugging models.

In their other study, [12] employed five OSS reliability models to compare two models they proposed, single release OSS reliability model, and a multi-release OSS reliability model. The suggested model outperforms the competition in terms of prediction and fitting, with stable performance in both areas.

In [15], five models were chosen that take imperfect debugging into account. The two different types of imperfect debugging models are integrated with the fault detection rate (FDR) function to create a more accurate model that has better performance than others. Type II imperfect debugging model encapsulates the relationship between fault detection, repair and introduction. It takes into account more random factors in the real software debugging process, thus performs better than type I model.

He [3] claims that the suggested model improved NHPP performance but he also remarks that lots of unknown parameters and insufficient fault data for parameters estimation may lead to inaccurate model parameter estimation and decreasing ability to predict.

In [9] the proposed model was compared with existing models validated on closed source software data sets without change point. It is not explicitly mentioned which models are used for comparison. Improved values of R^2 have been reported for Release 1 and Release 3.

4 Discussion

The reviewed models aim to estimate the reliability of OSS in multi-release scenarios, where each release can introduce new bugs that affect the system's reliability. These models demonstrate that there are more

factors to take into consideration while developing SRGMs for the modern context. These studies mostly start with a widely-accepted function and alter it by using revised assumptions for the OSS context, to propose new models. Almost all count on NHPP for this purpose. There exists common assumptions as imperfect debugging and changing failure rates over time which fit the nature of OSS.

LSE appears to be the most popular methodology for parameter estimation. However, it is noteworthy that some papers did not explicitly mention the parameter estimation technique they utilized [3], [15], [8].

The assumptions in recent models are explicit and considerably more in line with the reality of OSS. The studies generally evaluated the new models based on OSS data with multiple releases. It is evident that current models surpass older models on OSS projects with multiple releases based on metrics used in reviewed studies.

5 Conclusion

In conclusion, the importance of software reliability in modern software projects has increased, especially in the context of agile software development and open-source software. Although SRGMs are essential tools for evaluating software reliability, traditional SRGMs do not provide an accurate depiction of modern software context because they were created in the context of legacy software and a waterfall approach. Thus, their assumptions are made according to their context.

This study looked into the parameters, their estimation techniques, assumptions, applications and evaluations of recent SRGMs in relation to contemporary software projects. Overall, this paper aimed to analyze and explain the current developments in the research. The results of this study may serve as a reference for professionals that evaluate software reliability or develop new SRGMs for modern software projects. In future, more or different papers can be reviewed and interesting results can be obtained. As software development conventions will continue to change, SRGMs also need to keep up with these changes to continue to be useful for estimating reliability.

References

[1] Stanislav Chren, Radoslav Micko, Barbora Buhnova, and Bruno Rossi. Strait: A tool for automated software reliability growth analysis. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 105–110. IEEE, 2019.

[2] Amrit L Goel and Kazu Okumoto. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE transactions on Reliability*, 28(3):206–211, 1979.

[3] Yan He. Nhpp software reliability growth model incorporating fault detection and debugging. In *2013 IEEE 4th International Conference on Software Engineering and Service Science*, pages 225–228. IEEE, 2013.

[4] PK Kapur, RB Garg, Anu G Aggarwal, and Abhishek Tandon. Two dimensional flexible software reliability growth model and related release policy. In *Proceedings of the 4th National Conference, INDIACom-2010, New Delhi, India*, pages 25–26, 2010.

[5] PK Kapur, Hoang Pham, Anu G Aggarwal, and Gurjeet Kaur. Two dimensional multi-release software reliability modeling and optimal release planning. *IEEE Transactions on Reliability*, 61(3):758–768, 2012.

[6] Anurag Kumar. Software reliability growth models, tools and data sets-a review. In *Proceedings of the 9th India Software Engineering Conference*, pages 80–88, 2016.

[7] Radoslav Mičko, Stanislav Chren, and Bruno Rossi. Applicability of software reliability growth models to open source software. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 255–262. IEEE, 2022.

[8] Shubham Rawat, Nupur Goyal, and Mangey Ram. Software reliability growth modeling for agile software development. *Int. J. Appl. Math. Comput. Sci.*, 27(4):777–783, 2017.

[9] Iqra Saraf, Javaid Iqbal, Avinash K Shrivastava, and Shozab Khurshid. Modelling reliability growth for multi-version open source software considering varied testing and debugging factors. *Quality and Reliability Engineering International*, 38(4):1814–1825, 2022.

[10] Najeeb Ullah, Maurizio Morisio, and Antonio Vetro. A comparative analysis of software reliability growth models using defects data of closed and open source software. In *2012 35th Annual IEEE Software Engineering Workshop*, pages 187–192. IEEE, 2012.

[11] Jinyong Wang. Open source software reliability model with nonlinear fault detection and fault introduction. *Journal of Software: Evolution and Process*, 33(12):e2385, 2021.

[12] Jinyong Wang, Ce Zhang, and Jianying Yang. Software reliability model of open source software based on the decreasing trend of fault introduction. *Plos one*, 17(5):e0267171, 2022.

[13] Juhani Warsta and Pekka Abrahamsson. Is open source software development essentially an agile method. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 143–147. Citeseer, 2003.

[14] Alan Wood. Software reliability growth models. *Tandem technical report*, 96(130056):900, 1996.

[15] Ce Zhang, Wei-Gong Lv, Sheng Sheng, Jin-Yong Wang, Jia-Yao Su, and Fan-Chao Meng. Default detection rate-dependent software reliability model with imperfect debugging. *Applied Sciences*, 12(21):10736, 2022.

Can we trust Microsoft and Google Authenticators? Evaluating Security of Widely Used Authenticator Applications for Android

Berk Türetken
berk.turetken@aalto.fi

Tutor: Mario Di Francesco

Abstract

The widespread use of online systems increases the importance of secure authentication methods. Since the traditional single-factor authentication methods have become more vulnerable, the need for multi-factor authentication (MFA) has grown to increase security. Although MFA adds an extra layer of security, it has vulnerabilities while generating the one-time password (OTP) in the time-based OTP algorithm. By using reverse engineering techniques, this seminar paper analyzes the security vulnerabilities of two widely used mobile authenticator applications, Google Authenticator and Microsoft Authenticator. The analysis shows that both mobile authenticator applications have security vulnerabilities in storage or memory. Given these vulnerabilities, more precautions need to be taken in the implementation to enhance security, particularly when these applications are utilized for security-focused services.

KEYWORDS: Authentication, Multi-Factor Authentication, Security, Mobile Authenticator, Microsoft Authenticator, Google Authenticator, One-Time Password, Reverse Engineering

1 Introduction

In recent years, online systems have become an indispensable part of our daily life as they are extensively used in education, shopping, and entertainment among other use cases [13]. Correspondingly, security attacks have also evolved, resulting in numerous security breaches. One main reason for recent security breaches is a faulty authentication system, either in its design or implementation. For a long time, traditional single-factor authentication (SFA) methods – password-based authentication being the most common example – were the main focus in authentication system design [5]. In the face of the growing complexity of newer security attacks, password-based authentication has become more vulnerable than other methods. As a proof, in 2020, Microsoft reported that almost all compromised accounts that they monitor on a monthly basis did not have multi-factor authentication (MFA) enabled [10]. MFA is an authentication method where the user must not only know their login credentials, but also have something in their possession, such as a hardware token, smart card, or one-time password (OTP), which is more secure since it adds one more factor, to access the application [11]. MFA ensures identity verification through multiple independent factors that can be categorized as follows:

- **Knowledge:** Users know their username, password or personal identifier number (PIN).

- **Inherence:** Users use a physical characteristic, which is specific to an individual, such as fingerprint or face recognition.

- **Possession:** Users possess an item, such as a hardware token or a mobile phone with an OTP generator application [14].

Recently, the mobile phone has become an ideal choice for possession, as utilizing a mobile phone for MFA is cost-free, in contrast to a hardware token or smart card [11, 13].

Although MFA provides many advantages in terms of security, it is still not an ideal solution especially since mobile authenticator applications have security vulnerabilities. One of the possible security vulnerabilities is the generation of OTP value in the time-based OTP (TOTP) algorithm,

thus the disclosure of the shared secret [13, 14].

This paper analyzes the existing security vulnerabilities of Google Authenticator and Microsoft Authenticator, which are one of the most widely used mobile authenticator applications, by applying reverse engineering techniques. The working principles of the selected authenticators are also investigated with a special focus on the TOTP algorithm.

The rest of the paper is organized as follows. Section 2 considers the prevalent mobile phone-based MFA solutions including the determination of OTP through the TOTP algorithm. Section 3 describes the experimental setup, simulation of the attacks, forensic analysis and results. Section 4 includes an assessment of the Google and Microsoft Authenticators from a security perspective. Finally, Section 5 provides concluding remarks.

2 Working Principles of Google and Microsoft Authenticators

2.1 Most Popular Mobile Phone-based MFA Solutions

Different approaches and solutions utilize the mobile phone as an MFA. This paper studies three solutions and in all of them, OTP plays a fundamental role in one way or another.

The first and the less secure one among the other options is *OTP sent via Short Message Service (SMS)*, also known as *SMS verification* [11, 13]. In this solution, users try to use the service by sending a request and the service sends an SMS message that includes OTP once it gets the request from the user. After receiving OTP via SMS, users enter the OTP to use the service. Although OTP sent via SMS offers more security than SFA, using SMS as a form of MFA is not completely secure since attackers can intercept SMS messages and alter them [3, 4, 13].

Therefore, the use of mobile applications to generate OTPs, also known as *OTP generated by the authenticator*, has been becoming increasingly popular as an alternative [11]. In this approach, users need to enter the OTP that is generated by the mobile authenticator when they make a request to the service and reach the second-factor page (i.e., login page). Google and Microsoft Authenticators use the TOTP algorithm, which is specified in RFC 6238 [12], to generate OTP. For the generation of OTPs, the user and the service need to agree on a shared secret, also known as a seed value, which can be achieved by the user by determining and enter-

ing the value to the mobile authenticator. After that, the TOTP algorithm starts to generate OTPs on-demand or periodically, usually every 30 seconds [13]. Section 2.2 describes the generation of an OTP by the TOTP algorithm in more detail.

The last and the most secure solution is the *OTP via push notification*. In this solution, the service sends a challenge to the user via a mobile authenticator and accepts the request once the user accomplishes the challenge. To be able to accomplish the challenge, the user first has to log into the mobile authenticator, using a PIN code or inherence factors, such as fingerprint or face recognition. Then, the user needs to accept the notification which is sent from the service [11].

2.2 TOTP Algorithm

Since Google and Microsoft Authenticators use the TOTP algorithm and the shared secret is the key factor of the MFAs, it is worth mentioning how OTP is generated. OTPs are created by combining a shared secret and the current timestamp [12, 13]. The steps for the generation of an OTP can be found below:

1. Calculate T, which is the number of time steps, as follows:

$$T = \lfloor T_{unix}/T_s \rfloor$$

   where T_{unix}: number of seconds elapsed since 1 January 1970 00:00:00, and T_s: time step, 30 seconds is the default.

2. Convert T into hexadecimal format, T_h, ensuring that the resulting value comprises 16 hexadecimal characters. Add leading zeros if necessary.

3. Convert T_h to an array of 8 bytes and let it be A_h. Similarly, convert the shared secret, which the user enters, to an array of 20 bytes and let it be A_s.

4. Compute the HMAC of A_h and A_s by using the HMAC-SHA1 algorithm, let the result be H.

5. Obtain the final byte of the result and convert it to a decimal which will be the offset. Then, retrieve 4 bytes from the result by utilizing the

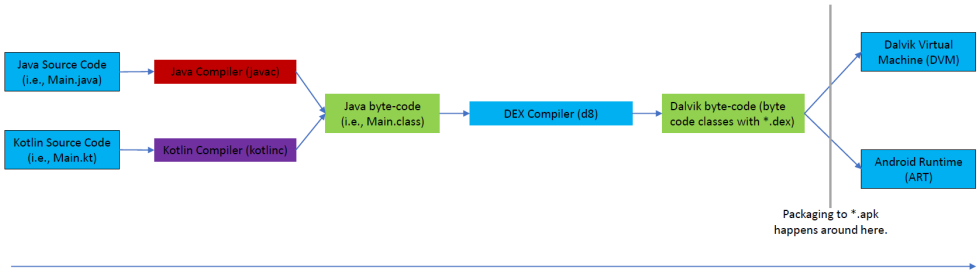


Figure 1. Android Development Cycle and Compilation of Android Applications

offset value, H_r.

6. Perform a bitwise AND operation to H_r with the hexadecimal value 0x7FFFFFFF. After that, convert it to a decimal, D_r.

7. Finally, compute the $OTP = D_r \bmod 10^d$ where d is the number of OTP digits, usually being chosen as 6.

These calculations are carried out both on the mobile authenticator and the service. If the resulting OTPs are identical, the request is approved [13, 14].

3 Forensic Analysis and Results

This section starts by defining the concept of reverse engineering and continues with selected tools, assumptions, simulation setup, attacks and results.

Reverse engineering is, briefly, the process of converting compiled software to source code [15]. The fundamental aim of reverse engineering is to analyze a technology to understand how it works, create a replica or an improved version of it, and obtain more insight information about its design, architecture, or components [7, 13]. Figure 1 shows the development cycle and the compilation of the Android applications. In Section 3.3, the reverse order of Figure 1 is followed while conducting the attacks.

3.1 Selected Tools for Forensic Analysis and Assumptions

The main goal of the simulations is to exploit as many vulnerabilities as possible for Google and Microsoft Authenticators. The targeted main vulnerabilities include retrieving the shared secret from storage as plain

text, copying the database file to another emulator and generating the same OTP values on that emulator, retrieving the shared secret from memory as plain text, and redirecting push notification requests. However, the latter one is only valid for Microsoft Authenticator [16] since Google Authenticator does not have a push notification feature. The following reverse engineering tools are used to conduct the forensic analysis and exploit the above-mentioned vulnerabilities: *Android Debug Bridge (adb)* to debug and communicate to the emulators, *Android Virtual Device (AVD) Manager* to configure the emulators, *apktool* to decompile and recompile Android Package Kit (APK) files after modifications, *DB Browser for SQLite* to view and perform database operations, *Eclipse Memory Analyzer (MAT)* to detect memory leaks, *JADX* to reach the source code, *Profiler in Android Studio* to analyze head dump files, *rootAVD* to root AVDs with Magisk, *Root Explorer* to view and access some files on the emulators, and *mitmproxy* to investigate the network traffic [13, 14].

Before conducting the simulations, two major assumptions have been made. First, the shared secret should be shared between the user and the service to generate OTPs. Once both parties agree on the shared secret, they can start to generate identical OTPs since the shared secret, current timestamp, and execution of the TOTP algorithm are the same for both parties. In this case, everything is known from the perspective of an attacker except the shared secret. If the attackers can obtain the shared secret, they may be able to pass the second factor [13]. The second assumption is that the attackers have root access to the system which can be achieved by exploiting a vulnerability in the system, having physical access to the rooted Android device, or rooting the device [6, 9].

3.2 Simulation Setup and Static Analysis

For the setup, Android Virtual Device (AVD) Manager was used, which is a tool in Android Studio, to create the emulators. Pixel 4, with an API level of 30, and Pixel 3a, with an API level of 29 had been chosen as the target devices. After having two emulators for the forensic analysis, both emulators were first rooted using *rootAVD* and *Magisk*. Then, the APKs of Google Authenticator with version 5.20R4 and Microsoft Authenticator with version 6.2303.2086 were downloaded from APKPure [1].

After having the APKs on the emulators, static analyses, such as analyzing the existing folders, files and classes, were conducted for both mobile authenticator applications by using *JADX* to detect vulnerabilities in

the source code. With these static analyses, the main purpose is to gain a deeper understanding of the functionalities of APK and the process of OTP generation. Then, we tried to determine whether the shared secret is still not encrypted in the storage as the first potential vulnerability because Polleit and Spreitzenbarth's study shows that Google and Microsoft Authenticators did not encrypt the shared secret while storing it in the database in 2018 [14]. Similarly, the same finding has been shown by Ozkan and Bicakci's research in 2020 [13] but Google has made a change for Google Authenticator in the latest update on the 14th of July 2022, and started to encrypt the shared secret in the storage according to their update notes. This update was verified with the static analysis because the shared secret was encrypted by using the Advanced Encryption Standard with Galois/Counter Mode (AES-GCM), which is considered as one of the most secure encryption methods [2, 8], if the Software Development Kit (SDK) version is greater than 22.

For Microsoft Authenticator, three different types of accounts could be added that are *personal account*, *work or school account* and *other (Google, Facebook, etc.) account*. When conducting the static analysis, we realized that Microsoft Authenticator did not encrypt the shared secret for *the other account* type while storing it in the database. In other words, Microsoft has not taken any action for at least three years, unlike Google.

3.3 Conducted Attacks and Results

Once the static analyses were completed for both mobile authenticator applications, they were decompiled by using *apktool* to make modifications to the APKs, such as making the applications debuggable, to validate the findings in dynamic analysis. Next, the modified APKs were installed on the emulators. As the last step before exploring the vulnerabilities, the shared secrets were entered manually into the mobile authenticators, assuming a hypothetical service, such as GitHub or Dropbox, provides that shared secret. After entering the shared secrets, the mobile authenticators were ready to launch attacks against the shared secrets.

The *accounts* table of the *databases* file, which is located within the */data/data/com.google.android.apps.authenticator2/databases* folder and reached by using *Root Explorer* or the combination of *adb* and *DB Browser for SQLite*, contained the encrypted version of the shared secret in Google Authenticator. Since the shared secret was encrypted with AES-GCM, decrypting it would require an infeasible amount of computational power

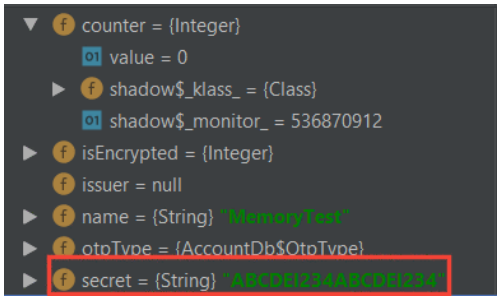


Figure 2. Shared secret in the memory (Google Authenticator)

and time to brute-force the key, thus finding the shared secret [2, 8]. Then, the file was duplicated onto the other emulator to observe if it is possible to generate the same OTPs but Google Authenticator took precautions for this action and did not generate the OTPs. To test the last objective, while the application was running, another account and key were created. After the creation of the account and key, we took a heap dump at any time by using the *Profile* feature of *Android Studio*. Figure 2 shows that the entered shared secret, which is *abcde12345abcde12345*, was stored in the `AutoValue_Account_Builder` class' objects as plain text.

The same objectives were performed for Microsoft Authenticator as well. First, the *accounts* table of the *PhoneFactor* file, which is located within the */data/data/com.azure.authenticator* folder, contained the encrypted shared secret for the *personal accounts* and *work or school accounts*. On the other hand, Microsoft did not encrypt the shared secret key for the other accounts as can be seen in Figure 3, therefore, the shared secret key could be obtained from the storage. Then, the *PhoneFactor* file was copied to the other emulator and only for the *other accounts*, the same OTPs were generated on both devices. For the last objective, we followed the same procedure as in Google Authenticator and the shared secret was obtained as plain text in the object of the *SecretKeyBasedAccount* class.

The detailed findings and corresponding screenshots from the mobile authenticator applications can be found in [17].

4 Security Analysis

After completing the forensic analyses and examining Google Authenticator and Microsoft Authenticator, it has been discovered that both of them are similar in terms of where they store the shared secrets and how they create the OTPs from the shared secrets.

Figure 3. Shared secret in the storage (Microsoft Authenticator)

Both Google and Microsoft Authenticators use the TOTP algorithm to produce the OTPs, and this results in one major disadvantage and advantage. The major disadvantage is that one can produce the same OTPs in another device once the shared secret is obtained since the TOTP algorithm does not need any unique values that are specific to the device, such as the IMEI number or MAC address. On the other hand, the major advantage is to move the accounts to another device in case of losing or changing the mobile device.

Furthermore, advanced obfuscators or ProGuard can make static analysis difficult for reverse engineers by obfuscating the source code. Unfortunately, both mobile authenticator applications do not apply advanced obfuscators or ProGuard. Hence, it is relatively straightforward to read and understand the source code. Moreover, since we can make modifications on both APKs, the mobile authenticator applications permit repacking.

With the latest update, Google Authenticator provides secure storage for the shared secret. It does not also allow cloning the database file to another device but retrieving the shared secret from the memory is still vulnerable in Google Authenticator. Although Microsoft Authenticator provides secure storage for the shared secret of the *personal account* and *work or school account* types, it does not encrypt the shared secret for the *other account* type. In contrast to Google Authenticator, Microsoft Authenticator allows cloning the database file to another device for the *other account* type. Similar to Google Authenticator, fetching the shared secret from the memory is possible in Microsoft Authenticator. Figure 4 presents an overview of the security mechanisms of the Google and Microsoft Authenticators.

Mobile Authenticator Application Name	Secure Storage Usage	Secure Memory Usage	Advanced Obfuscators	ProGuard	Anti-repacking	Possibility of Cloning
Google Authenticator	Yes	No	No	No	No	No
Microsoft Authenticator	No (to some extent)	No (to some extent?)	No	No	No	Yes (to some extent)

Figure 4. Security mechanisms of Google Authenticator and Microsoft Authenticator

5 Conclusion

Firstly, the use of an MFA is crucial and beneficial to enhance security as it adds an extra layer of protection in contrast to SFA. Although two of the most widely used mobile authenticator applications still have security vulnerabilities, adding an extra layer of security with MFA will not worsen the situation. This paper has analyzed the security vulnerabilities of Google Authenticator and Microsoft Authenticator with a focus on retrieving shared secret and copying the database file to another device.

In light of the conducted attacks, Google Authenticator allows the retrieval of the plain text version of the shared secret from the memory. In Microsoft Authenticator, the shared secret can be obtained as plain text for the other account type from the storage and memory. Microsoft Authenticator does not also prevent copying the database file to another device, therefore, an attacker can produce the same OTPs without retrieving the shared secret as plain text.

It has been shown that two-factor authentication might immediately become an SFA for the Google and Microsoft Authenticators if the attackers can obtain the shared secret as plain text or clone the application to another device. Hence, considering the vulnerabilities that are highlighted in this paper, additional measures should be implemented to improve security, especially when using these mobile authenticator applications for security-oriented services.

References

[1] APKPure. APKPure. https://m.apkpure.com/. Accessed: 2023-09-04.

[2] Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. Augmented Secure Channels and the Goal of the TLS 1.3 Record Layer. In *Provable Security: 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings 9*, pages 85–104. Springer, 2015.

[3] Divya Buttan. *Hacking the Human Brain: Impact of Cybercriminals Evoking Emotion for Financial Profit*. PhD thesis, Utica College, 2020.

[4] Aniket S. Chaudhari. Security Analysis of SMS and Related Technologies.

Research Master Thesis, Dept. of Mathematics and Computer Science, Eindhoven University of Technology, 2015.

[5] Sanchari Das, Bingxing Wang, and L. Jean Camp. MFA is a Waste of Time! Understanding Negative Connotation Towards MFA Applications via User Generated Content. *arXiv preprint arXiv:1908.05902*, 2019.

[6] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, and Marcel Winandy. Privilege Escalation Attacks on Android. In *Information Security: 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers 13*, pages 346–360. Springer, 2011.

[7] Eldad Eilam. *Reversing: Secrets of Reverse Engineering*. John Wiley & Sons, 2011.

[8] Marc Fischlin, Felix Günther, Giorgia Azzurra Marson, and Kenneth G. Paterson. Data Is a Stream: Security of Stream-Based Channels. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II 35*, pages 545–564. Springer, 2015.

[9] Jason Gu, Veo Zhang, and Seven Shen. ZNIU: First Android Malware to Exploit Dirty COW. *TrendLabs Security Intelligence Blog*, 2017.

[10] Kevin Jensen, Faiza Tazi, and Sanchari Das. Multi-Factor Authentication Application Assessment: Risk Assessment of Expert-Recommended MFA Mobile Applications. *Proceeding of the Who Are You*, 2021.

[11] Piotr Lewandowski, Anna Felkner, and Marek Janiszewski. Security analysis for authentication and authorisation in mobile phone. *Przegląd Elektrotechniczny*, 95(8):132–138, 2019.

[12] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. TOTP: Time-Based One-Time Password Algorithm. Technical report, Internet Engineering Task Force, 2011.

[13] Can Ozkan and Kemal Bicakci. Security Analysis of Mobile Authenticator Applications. In *2020 International Conference on Information Security and Cryptology (ISCTURKEY)*, pages 18–30. IEEE, 2020.

[14] Philip Polleit and Michael Spreitzenbarth. Defeating the Secrets of OTP Apps. In *2018 11th International Conference on IT Security Incident Management & IT Forensics (IMF)*, pages 76–88, 2018.

[15] Michael G. Rekoff. On Reverse Engineering. *IEEE Transactions on systems, man, and cybernetics*, 15(2):244–252, 1985.

[16] Mayur Santani, Justin Hall, Curtis Love, Jason Howell, and Pritam Ovhal. How to use additional context in Microsoft Authenticator notifications - Authentication methods policy. https://learn.microsoft.com/en-us/azure/active-directory/authentication/how-to-mfa-additional-context, 2023. Accessed: 2023-06-02.

[17] Berk Türetken. Screenshots from Authenticators. https://rb.gy/cspau, 2023. Accessed: 2023-10-04.

Blockchain and consensus algorithms: security vulnerabilities and tradeoffs

Chathurangi Edussuriya
chathurangi.edussuriya@aalto.fi

Tutor: Shushu Liu

Abstract

Blockchain technology will be an integral component of Web 3.0. The consensus mechanism is a crucial component of the underlying technology of the blockchain. The architecture of consensus algorithms introduce both distinctive features and exploitable vulnerabilities. Consequently, performance metrics, such as transaction throughput, decentralization, power consumption, scalability, and permission model are compromised by security vulnerabilities. There are numerous applications for blockchain technology, spanning from cryptocurrencies, financial industry, and supply chains to healthcare. Consensus algorithms that are appropriate for applications vary based on the application requirements, performance evaluation of the consensus algorithm, and its security vulnerabilities. This paper examines the functionality of the main consensus algorithms and their security flaws. In addition, this study describes the trade-offs between evaluation metrics and potential security vulnerabilities. This research also reviews the applicability of consensus algorithms to a variety of applications based on their respective needs. This paper serves as a reference paper for the selection of consensus algorithms in both academy and industry

KEYWORDS: Blockchain, Proof-of-X protocols, Cryptocurrency, Consensus algorithms

1 Introduction

Blockchain technology, introduced by Satoshi Nakamato, has paved the way for the emergence of web 3.0 [17]. Web 3.0 technology is regarded as the next phase of the internet, in which data in the virtual space is managed without human intervention. It will employ Decentralized Ledger Technology (DLT) and blockchain alongside big data, machine learning, and artificial intelligence technologies [13]. Therefore, blockchain technology plays a significant role in the process of transforming the internet into an autonomous, intelligent system. Blockchain technology records transactions of a business network in an immutable, distributed ledger. Figure 1 shows the architecture of the bitcoin blockchain. Each block of data in a blockchain contains the hash of the previous block, data, and the own hash of the block [16]. The immutability of the blockchain architecture is achieved by storing the hash of the current block in the succeeding block. Using complex cryptographic functions, the hash value of a block is calculated, making it impossible to replicate. Consequently, if a block is altered, it can be recognized by its hash value.

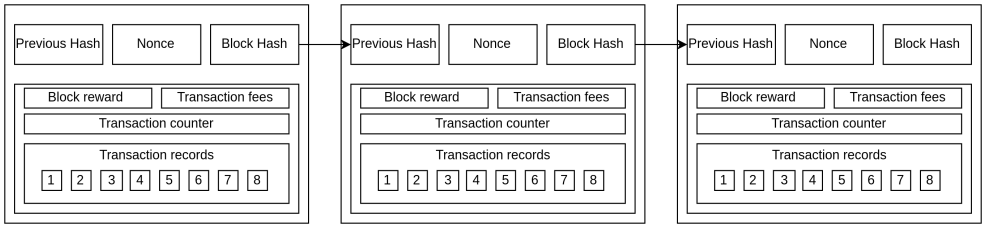


Figure 1. Blockchain architecture

Blockchain is controlled in a decentralized manner, with no single authority in control [4]. When a block is added to the blockchain, all participants must agree to include it in the distributed ledger. This method of validation is known as consensus. The consensus algorithm can be considered the key component of the blockchain. Different proof-of-X protocols, such as proof-of-work, proof-of-stake, and proof-of-authority, have emerged over time [19]. Furthermore, voting based consensus algorithms, such as Practical Byzantine Fault Tolerance and Proof of Elapsed Time [6] were developed. These protocols have a tradeoff between security measures and system performance.

The three primary categories of blockchain are public, private, and consortium. The public blockchain is permissionless, whereas the private blockchain is permissioned. Both types of blockchains comprise the con-

sortium blockchain [20]. Consistent with their respective requirements, these varieties of blockchain employ employ different consensus algorithms.

The research examines consensus algorithms in three key areas. Initial discussion focuses on various consensus protocols and the inherent security flaws of these protocols. Consequently, the study reviews the compromises between efficacy, decentralization, scalability, and security of these protocols. The study concludes by analyzing the applications that employ these protocols in light of their respective compromises.

The paper is organized as follows. Section 2 presents the existing consensus protocols and the inherent security vulnerabilities of these protocols. Section 3 compares the costs associated with the protocols with the security measures. The applications of the protocols are described in Section 4. Section 5 concludes the paper.

2 Consensus algorithms

This section discusses the functionality of the main consensus algorithms and their security flows.

2.1 Proof-of-work (PoW)

Proof-of-work mandates that participants calculate hashing functions until the output contains a minimum number of preceding zeros [11]. These calculations are performed by miners. The successful miner announces the newly discovered hash value to the network. After the hash value has been verified, the block is submitted to the blockchain. Miners must demonstrate that a significant quantity of processing power was used to attain the desired result. Due to this considerable processing power, it is challenging for a single miner to control the blockchain (more than 51%). Consequently, this ensures security of the blockchain. However, this protocol presents security vulnerabilities.

Security vulnerabilities

- 51% attack: The miner obtains more than 50% of the control of the blockchain [8]. As the attacker gains more control over the blockchain network, attacker can determine which blocks are added to the network, preventing other miners from participating. Additionally, the attacker can double-spend the tokens.

- Selfish mining: The miner retains the blocks it has discovered and does not broadcast them. Without interference from other miners, the attacker mines additional blocks. The attacker will only broadcast the blocks it has if the mined blocks form a chain that is longer than the public chain. Therefore, blocks mined by other miners become invalid [23].

- Finney attack: The miner includes a fraudulent transaction in a block it has mined and swiftly spends before discovering the block [2].

- Timestamp attack: The miner modifies the timestamp of the block to an earlier time. It permits the attacker to create new blocks or reverse transactions based on the modified time frame, thus altering the history of the blockchain.

2.2 Proof-of-stake (PoS)

Proof-of-stake consensus mechanism decreases the high computational cost associated with Proof-of-Work. In Proof-of-Stake, participants or validators are chosen based on the quantity of stake they possess, whereas in Proof-of-Work, miners are chosen based on their ability to solve a cryptographic puzzle. In a process known as minting, validators are selected at random [18]. It guarantees that no single entity can acquire control of the blockchain. A transaction fee is paid to the validators whenever it creates a new block. PoS adheres to the longest chain rule, wherein the chain with the most validators and the most blocks is considered to be the valid blockchain. Ethereum, one of the most prominent blockchains, transitioned from PoW to PoS with the release of version 2.0.

Security vulnerabilities

- Nothing at stake problem: To maximize their reward, the validators of the blockchain validate multiple chains simultaneously [14]. Validating a chain takes the stake of the validators into account. Nonetheless, if consensus cannot be reached on the valid chain, multiple chains may be validated, resulting in a double spending attack.

- Long-range attack: The attacker modifies the history of blockchain, spanning several years to decades [14]. By manipulating the history,

the adversary gains the ability to add new blocks or roll back transactions.

2.3 Delegated Proof-of-stake (DPoS)

Delegated Proof of Stake is an enhanced variant of Proof of Stake [24]. A delegate is selected to represent the electors at the consensus meeting. If a delegate fails to perform as anticipated, the electorate may choose a replacement. The delegate distributes the resulting profit to the electorate. However, delegate authority can be abused because they validate newly inserted blocks.

Security vulnerabilities

- Sybil attack: The attacker creates numerous fraudulent participants to increase their voting strength in order to nominate themselves as the delegate [9]. The adversary manipulates the blockchain in order to obtain additional rewards.

- Bribery attack: The adversary induces the participants to designate the adversary as the delegate. In return, the adversary increases the profit of participants by increasing their reward.

2.4 Proof-of-Activity (PoA)

Bentov et al.[5] proposed proof-of-Activity, which addresses the drawbacks of both PoW and PoS consensus protocols. In PoA, the blockchain is secured by validators who validate blocks in accordance with PoS and miners who mine blocks in accordance with PoW. This method is more secure and energy-efficient than PoW and PoS.

Security vulnerabilities

- 51% attack: The adversary with the higher computational capability solves the PoW puzzle to acquire a larger PoS stake. With more ownership in Proof-of-Stake, an adversary obtains control of the blockchain by initiating blocks and collecting rewards.

- Bribery attack: The attacker bribes the validators in order to gain their favor. In exchange for obtaining control of the blockchain, the attacker

rewards validators.

- Eclipse attack: The attacker isolates a specific node or group of nodes from the blockchain and manipulates the observation of the blockchain to conceal or alter the data from the node [12]. For consensus, the node would not obtain a complete picture of the state of the blockchain.

2.5 Practical Byzantine Fault Tolerance (PBFT)

The PBFT algorithm proposed by Castro and Liskov [6] is an implementation of the Byzantine general problem algorithm. Even in the presence of deceptive nodes, this algorithm can reach a consensus. The leader node notifies all blockchain nodes of the implementation of a transaction. The nodes analyze the information and provide approval of the transaction to the leader. The new block is added to the blockchain if the transaction is acknowledged by the majority of nodes. A malicious node can be identified by the leader node, which can then eliminate it from the consensus process.

Security vulnerabilities

- Sybil attack: The attacker creates numerous malicious nodes and acquires the voting power of the blockchain. The PBFT algorithm is designed to function even in the presence of malicious nodes. However, there must be a minimum of 3f+1 trusted nodes, where f is the maximum number of malicious nodes [22].

- Forgery attack: By intercepting the communication between the nodes, the attacker causes a man-in-the-middle attack. The attacker modifies the consensus process by adding or spending defective blocks of the blockchain.

- Denial-of-service (DOS) attack: The attacker floods the blockchain by injecting requests and overloading the nodes.

2.6 Proof of Elapsed Time (PoET)

The PoET algorithm utilizes the same consensus mechanism as the PBFT algorithm, excluding the selection of the leader. Based on a random wait-

ing period, PoET selects the leader of the blockchain [7]. Waiting time of each node is determined by a random number generated by a module of trusted hardware. Once the waiting time has elapsed, the network designates the node that completed the waiting time first as the leader of the blockchain. This procedure ensures no malicious node can obtain control of the blockchain.

Security vulnerabilities

- Time tampering attack: The adversary modifies the trusted hardware module and reduces the waiting time to designate themselves as the leader of the blockchain

- DoS attack: The adversary continuously requests wait times from trusted hardware modules, preventing other nodes from completing a cycle of wait time.

- Sybil attack: The attacker creates multiple nodes in order to enhance the likelihood of being elected leader.

3 Comparative Analysis of the Tradeoffs

Increased algorithmic security is always accompanied by increased costs. Consequently, this section evaluates the protocols based on the transaction rate, decentralization, power consumption, scalability, and permission model with the most prevalent security flaws as shown in Table 1. The security flaws encountered by consensus protocols are outlined in Section 2 and presented concisely in Table 1.

3.1 Transaction throughput

Transactions Per Second (TPS) is a metric that quantifies the number of transactions executed in a single second. Table 1 compares the transaction rates of well-known cryptocurrencies utilizing these consensus protocols [4]. Due to the time required for mining, PoW has a lower transaction throughput, whereas PoS and PoA have a higher transaction rate than PoW. PBFT and PoET, as voting-based consensus algorithms, have higher transaction rates than DPoS-based EOS, which has a rate of 4000 TPS.

Protocol	TPS (cryptocurrency)	Decentralization	power consumption	scalability	permission model	security vulnerabilities
PoW	10 (Bitcoin)	high	high	low	permissionless	51% attack, selfish mining, double spending
PoS	100 (Nxt)	medium	low	medium	permissionless	nothing at stake, long range attack
DPoS	4000 (EOS)	low	low	high	consortium	sybil attack, bribery attack, 51% attack, double spending
PoA	100 (Komodo)	low	low	high	permissioned	51% attack, double spending
PBFT	1500 (Ripple)	low	low	high	consortium	forgery attack, DoS
PoET	2000 (sawtooth)	high	low	medium	consortium	time tampering attack, DoS

Table 1. Evaluation matrix of consensus algorithms along the security vulnerabilities

3.2 Decentralization

Several factors are considered when evaluating the decentralization of consensus algorithms, including the number of nodes required for consensus, the requirements for consensus participants, and the degree of power distribution among consensus participants [4]. The PoW consensus process is highly decentralized, as Participation is open to anyone. PoS and DPoS are semi-centralized because validators must have a stake prior to participation. PoA inherits decentralized characteristics of PoW. PoET is highly decentralized because anyone can participate, whereas PBFT is less decentralized because consensus requires fewer participants.

3.3 Power consumption

Power consumption is a comparison of the quantity of energy required to add one new block to a blockchain using a particular consensus mecha-

nism. PoW consumes the most energy in the consensus procedure due to the complexity of mining. In contrast, PoS, PoA, and DPoS require less energy to achieve consensus. PBFT and PoET are more energy-efficient than PoW because they are designed to outperform it.

3.4 Scalability

PoW has a limited scalability due to the low transaction throughput, whereas PoS has a higher scalability. DPoS is highly scalable owing to its high TPS. In addition to being highly scalable, PBFT quickly reaches consensus. Although it can process large volumes of transactions, the scalability of PoA and PoET depends on the time it takes to reach consensus.

3.5 Permission model

According to their access management, as described in Section 2, blockchains can be categorized into three models: permissionless, permissioned, and consortium. Depending on the security and architecture of consensus mechanisms, various consensus protocols may be utilized in these permission models [15]. Permissionless blockchains use PoW and PoS, whereas permissioned blockchains use PoA. Due to their architectures, DPoS, PBFT, and PoET can be utilized by both permissionless and permissioned blockchains as constortium. Permissionless blockchains include Bitcoin, Ethereum, and Litecoin, whereas permissioned blockchains include Hyperledger Fabric and Corda. Hyperledger Sawtooth and R3 Corda are consortium-based blockchains.

4 Applications of the protocols

The suitability of the protocol that should be utilized by each application varies depending on the requirements of the applications and the performance tradeoffs of consensus as described in Section 3.

PoW is better suited for applications that require increased security and decentralization but less transaction throughput and scalability. The inherent security features of this algorithm make it a good fit for cryptocurrencies [26]. PoS can be utilized by energy-efficient and low-power-consuming applications. Applications that demand a higher transaction throughput can utilize DPoS. This consensus algorithm is ideal for use in industrial blockchain networks [3] because it strikes a balance between

security and scalability. PoA algorithm, which has strong management capabilities in consensus can be used for regulatory compliance and audit.

The PBFT algorithm is optimal for financial systems due to its high transaction throughput and minimal latency. In addition, the high fault tolerance of algorithms makes it appropriate for use in communication networks [4]. PoET can be utilized by applications that prioritize lower power consumption and higher throughput. Applications involving the Internet of Things (IoT) [21], supply chains [10], healthcare [1], and edge computing [25] can benefit from this protocol.

5 Conclusion

The blockchain is an integral part of the endeavor to redesign the current web architecture for the subsequent generation. This paper examines the consensus algorithms utilized by the blockchain. This paper discusses the functionality of the main consensus algorithms and the security flaws. In addition, the security vulnerabilities with other performance metrics, such as transaction throughput, decentralization, power consumption, and scalability of the consensus algorithms are compared to determine the beneficial and negative aspects of each algorithm. Comparisons demonstrate that the PoW is the most secure algorithm. It is, however, less efficient and scalable. The PoS algorithm, which was devised to address the deficiencies of PoW, is more efficient than PoW. In DPoS, the transaction rate is the highest. The PoA is a highly scalable algorithm that incorporates PoW and PoS. Due to its consensus architecture, the PBFT algorithm outperforms Proof-of-X protocols in terms of scalability and energy consumption. PoET, which uses the same consensus architecture as PBFT, has a high degree of decentralization because it permits a large number of participants to take part in the decision-making procedure. The consensus algorithm that is most appropriate varies, depending on the needs of various applications. The suitable consensus algorithm can be selected, by comparing the requirements to the performance metrics and security vulnerabilities of the consensus algorithms.

References

[1] Cornelius C Agbo, Qusay H Mahmoud, and J Mikael Eklund. Blockchain technology in healthcare: a systematic review. In *Healthcare*, volume 7, page 56. MDPI, 2019.

[2] Shubhani Aggarwal and Neeraj Kumar. Attacks on blockchain. In *Advances in Computers*, volume 121, pages 399–410. Elsevier, 2021.

[3] Jameela Al-Jaroodi and Nader Mohamed. Blockchain in industries: A survey. *IEEE Access*, 7:36500–36515, 2019.

[4] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, 2020.

[5] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

[6] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[7] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. On security analysis of proof-of-elapsed-time (poet). In *Stabilization, Safety, and Security of Distributed Systems: 19th International Symposium, SSS 2017, Boston, MA, USA, November 5–8, 2017, Proceedings 19*, pages 282–297. Springer, 2017.

[8] Usman W Chohan. The double spending problem and cryptocurrencies. *Available at SSRN 3090174*, 2021.

[9] John R Douceur. The sybil attack. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1*, pages 251–260. Springer, 2002.

[10] Chathurangi Edussuriya, Kasun Vithanage, Namila Bandara, Janaka Alawatugoda, Manjula Sandirigama, Upul Jayasinghe, Nathan Shone, and Gyu Myoung Lee. Bat—block analytics tool integrated with blockchain based iot platform. *Electronics*, 9(9):1525, 2020.

[11] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[12] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015.

[13] Jim Hendler. Web 3.0 emerging. *Computer*, 42(1):111–113, 2009.

[14] Wenting Li, Sébastien Andreina, Jens-Matthias Bohli, and Ghassan Karame. Securing proof-of-stake blockchain protocols. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, pages 297–315. Springer, 2017.

[15] Du Mingxiao, Ma Xiaofeng, Zhang Zhe, Wang Xiangwei, and Chen Qijun. A review on consensus algorithm of blockchain. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2567–2572. IEEE, 2017.

[16] Paul Müller, Sonja Bergsträßer, Amr Rizk, and Ralf Steinmetz. The bitcoin universe: An architectural overview of the bitcoin blockchain. In *11. DFN-Forum Kommunikationstechnologien*. Gesellschaft für Informatik eV, 2018.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[18] Cong T Nguyen, Dinh Thai Hoang, Diep N Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.

[19] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1):101–128, 2018.

[20] Sunny Pahlajani, Avinash Kshirsagar, and Vinod Pachghare. Survey on private blockchain consensus algorithms. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICI-ICT)*, pages 1–6. IEEE, 2019.

[21] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. Blockchain and iot integration: A systematic survey. *Sensors*, 18(8):2575, 2018.

[22] Soumyashree S Panda, Bhabendu Kumar Mohanta, Utkalika Satapathy, Debasish Jena, Debasis Gountia, and Tapas Kumar Patra. Study of blockchain based decentralized consensus algorithms. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pages 908–913. IEEE, 2019.

[23] Muhammad Saad, Laurent Njilla, Charles Kamhoua, and Aziz Mohaisen. Countering selfish mining in blockchains. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 360–364. IEEE, 2019.

[24] Sheikh Munir Skh Saad and Raja Zahilah Raja Mohd Radzi. Comparative review of the blockchain consensus algorithm between proof of stake (pos) and delegated proof of stake (dpos). *International Journal of Innovative Computing*, 10(2), 2020.

[25] Ruizhe Yang, F Richard Yu, Pengbo Si, Zhaoxin Yang, and Yanhua Zhang. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1508–1532, 2019.

[26] Yong Yuan and Fei-Yue Wang. Blockchain and cryptocurrencies: Model, techniques, and applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(9):1421–1428, 2018.

Psychometry for Researching Usable Security

Fajar Malik
fajar.malik@aalto.fi

Tutor: Sanna Suoranta

Abstract

As software becomes increasingly intricate and plays a greater part in daily lives, it is crucial to verify not only its security, but also its usability. Usability testing is one of many methods to verify that systems are easy to use. One of the approaches is user-based evaluation, where a user directly participates in the evaluation. However, asking directly from users may introduce the problem of lack of motivation to complete the tasks. For this reason, psychometric tools could be preferred, as it produces more authentic evaluation on user experience. However, there has been few human-centered research that focuses on improving usability of security.

This paper aims to review various techniques in psychometry currently used to study usability of security features. The analysis concluded that the use of psychometry has yielded positive outcomes in the field of usable security, facilitating the ultimate goal of making the most secure practice to be the most user-friendly approach. Recent studies suggest that personalization of security features by utilizing psychometry is likely to continue in the future. However, the sparseness of studies in this topic indicates that there is still a long way to go until psychometry could be widely applied in usable security.

KEYWORDS: psychometry, usability, security

1 Introduction

As software becomes increasingly intricate and plays a greater part in daily lives, it is important to verify not only its security, but also the usability of the security features. The term usability describes the capability of a system to provide a condition for its intended users to use it safely, effectively and efficiently, while generating positive response from users. Usability testing is one of many methods to verify that such capability is achieved. One of the approaches to usability testing is user-based evaluation, where a user directly participates in the evaluation [1].

However, asking directly from users introduce the problem of amotivation, the lack of motivation to persist in completing tasks towards a goal. Due to this reason, psychometric tools have been utilized in order to test usability. Psychometry refers to the extraction of qualitative mental and emotional state into quantitative or numerical values. Rusu et al. [2] found that the use of psychometry produces more authentic evaluation on user experience, therefore should be preferred instead of asking users opinion. There is increasing amount of research that investigates usage of psychometric tools in improving usability of systems. Progress has been made to improve usability of security products and user comprehension of security issues [3]. However, there has been few human-centered research that focuses on improving usability of security.

Consequently, this paper aims to review various techniques in psychometry that are currently being used to study the usability of security features. While the topic seem to suggest a broad scope of review, the limited literatures on this subject provide a relatively narrow discussion. This paper is organized as follows. Section 2 first discusses definitions of usability. Subsection 2.1 presents key concepts and theories in psychometry. Subsection 2.2 presents the definition and main concepts in usable security. Section 3 presents researches related to the use of psychometry in usable security. Section discusses the techniques presented in this paper along with their results. Finally, Section 6 presents some concluding remarks.

2 Usability

The term usability describes the capability of a system to provide a condition for its intended users to use it safely, effectively and efficiently, while

generating positive response from users. Nielsen [4] defined 10 general rules of thumb for creating user-friendly and effective interfaces. These heuristics are: displaying system status clearly, aligning the system with the real world, empowering users with freedom and control, maintaining consistency, preventing errors, prioritizing recognition over recall, optimizing flexibility and efficiency of use, focusing on aesthetic and minimalist design, aiding users in recognizing and recovering from errors, and providing comprehensive help and documentation. These heuristics assist designers to create interfaces that are intuitive, efficient, and enjoyable for users, improving the overall user experience.

To verify that these heuristics are achieved, usability testing is generally conducted. One of the approaches to usability testing is user-based evaluation, where a user directly participates in the evaluation [1]. However, asking directly from users introduce the problem of amotivation, the lack of motivation to persist in completing tasks towards a goal. Due to this reason, psychometric tools have been utilized in order to test usability.

2.1 Psychometry

Psychometry is a field of study within psychology that focuses on developing as well as validating questionnaires and tests used for measuring psychological variables, such as personality traits, knowledge, attitudes, abilities, emotions, and mental states [5]. The aim of psychometry is to develop reliable and valid measurement instruments that can be used to measure these variables in individuals and groups. These instruments range from questionnaires to performance tests, and are designed to provide objective and quantifiable data about various psychological constructs.

The main challenge in the psychometry field is the difficulty to institute ground truth, as the psychological variables are qualitative, while the measurement instruments are quantitative [6]. Psychometry could be effectively implemented in human-computer interaction (HCI), where the software retrieves psychometric traits from its users. This data could then be calculated with the aim to adjust the experience to suit the particular user better, therefore improving overall user experience.

2.2 Usable security

Usable security seeks to combine the need for strong security with the need for usability, in order to ensure the security measures are not only effective, but also practical and accessible to the average user. It is important to understand this definition, since providing a high level of security may break usability [7]. For example, having a long and complex password provides strong security. However, this complexity would require more effort from the user to remember it. On the other hand, having a short and readable password is more vulnerable for attackers to guess. The ideal goal is to make the most secure method to be the easiest. Therefore, there are needs to replace the text-based password system, one alternative is graphical password. However, due to its nature of involving graphics, it is vulnerable to shoulder-surfing attacks. To solve this problem, recent researches [8, 9] have developed a new type of personal authentication system using electroencephalography (EEG) signals, which is a measurement of the electrical activity inside the human brain. Compared to the currently widely-used authentication systems, this method allows easier use and it is more difficult for attackers to steal.

3 Psychometry applications in usable security

Psychometric methods have been beneficial in solving usable security problems. There are few researches that have studied the applications of psychometry to investigate usability of security features.

3.1 User comfort and password construction

In order to assess the usability of IT security management tools (ITSM), Jaferian et al. [10] constructed usability herustics that utilized activity theory, a ground-breaking theory that emerged from Soviet psychology. This theory, formulized by Engeström [11] in 1999, indicates that all activity possesses a subject that conducts the activity, and an object which the activity is aimed towards. The heuristics constructed by Jaferian et al. [10] was found to have similar degrees of relevance, applicability, and learnability compared to the 10 heuristics developed by Nielsen [4].

This research by Jaferian et al. [10] then inspired Haque et al. [5], who measured user comfort when interacting with a certain user interface to

create an effective password. The study built a scale to measure user comfort, after which it is used by their users to assess their password construction experience. Using the scale, this research concluded that older users tend to report lower level of comfort. This research, conducted in 2016, also claimed to be the first to investigate the use of psychometric concepts in researching usable security, pioneering research in this area.

3.2 Cognitive factors and text-based passwords

Loos et al. [12] attempted to correlate user cognitive variability factors with passphrase selection. The user cognitive variability factors used in this research is the locus of control personality types. Locus of control describes the degree of control to which a person feels they possess in their behavior [13]. Internally controlled individual view themselves as having ample amount of personal control and tend to be more inclined to take ownership of their actions. For instance, they may attribute their success on a test to their dilligent efforts. On the other hand, externally controlled individuals attributes their behaviors towards external factors or influences, such as attributing their success on a test to the test being easy.

The main inspiration to this research by Loos et al. [12] was to analyse the challenges that emerge between the memorability and usability of passwords. Participants were first asked to perform locus of control personality test to determine whether they are internally or externally controlled types. Having recorded their locus of control personality types, participants were then presented with passphrases constructed from four passphrase types: no vowels (NV), four categories (4C), room objects (RO), and animal associatives (AA). Each of all four types are divided into two: imposed by the researchers and user-created. Figure 1 presents the high score result of recall ability tests. The first letter signifies whether the passphrase type is imposed (I) or created by user (C).

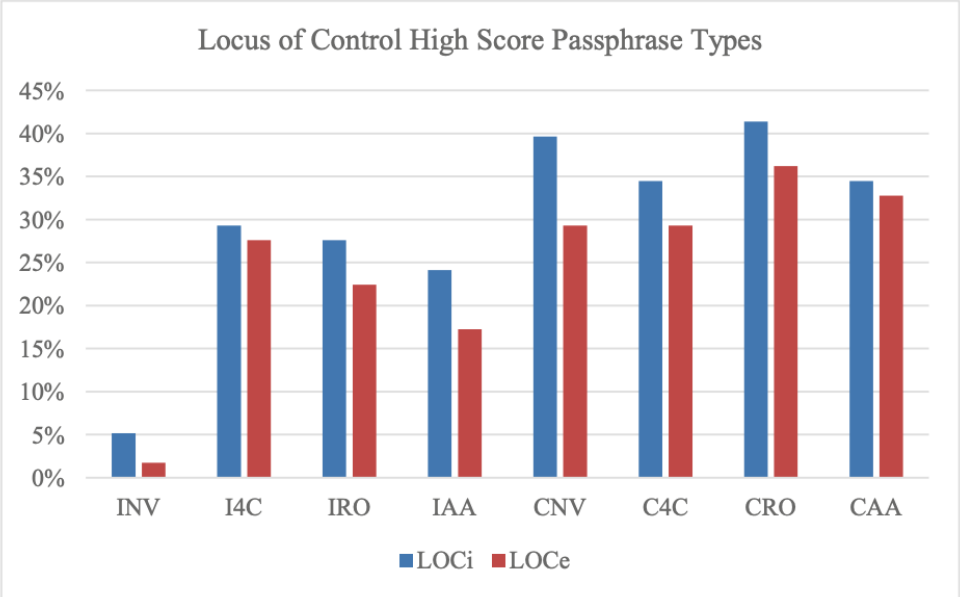


Figure 1. Recall ability high score for all passphrase types [12]

The locus of control traits were found to affect decision making throughout passphrase selection and recall tests. This study concluded that both internally and externally controlled participants gave the passphrase from created room objects (CRO) the highest recall ability. In addition, internally controlled individuals favour the created no-vowel category (CNV), while externals favour the created animal associations (CAA). Meanwhile, both locus of control personalities gave the imposed no vowel passphrase the least favoritism.

3.3 Cognitive factors and CAPTCHA

While Loos et al. [12] correlated cognitive factors with passphrase preference and recall ability, Belk et al. [14] correlated cognitive differences with preference and performance of users in completing Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) tests. CAPTCHA, which was first introduced by Blum et al. [15] in 2003, is currently widely used by online services to determine whether the user trying to access their service is human or an automated robot. The current version generally shows a randomly generated series of letters or numbers which appear distorted, and the user is asked to type what is seen into the text box in order to pass the test and prove their human identity.

Belk et al. [14] took two cognitive factors into consideration, namely cognitive styles and cognitive processing abilities. The cognitive style test

Table 1. Factors investigated in the research [14]

Study	Cognitive Factors	CAPTCHA Design Factors	Usability Factors
A	Cognitive styles	CAPTCHA Type	User preference
			Completion time
			Success rate
B	Cognitive processing abilities	CAPTCHA visual complexity	Completion time
			Success rate

utilized Cognitive Style Analysis (CSA) test first developed by Riding [16]. The CSA test, specifically the Verbalizer-Imager (VI) dimension, aims to distinguish the characteristics of an individual with regard to their information presentation and learning performance. Verbalizer tends to learn best from text, while imagers learn better from graphics. Even though this test was initially found to have low reliability, it was improved by following the guidelines provided by Rezaei and Katz [17].

In the research by Belk et al. [14], cognitive processing abilities, on the other hand, were recorded from the users by testing their ability to remember words and images with varying difficulties. This data was then calculated to split the users into two categories, namely limited and enhanced cognitive processing abilities. Table 1 presents the factor that were investigated in this research. The users were also asked to solve CAPTCHA challlenges with varying difficulties, such as the number of letters or pictures and the amount of distortion. The researchers also collected data on participants' self-reported preference for different types of CAPTCHA.

The findings of this study suggest that users with any combination of the two cognitive factors yield varying performance, favourability, as well as overall impact as they are interacting with various CAPTCHA mechanisms and level of visual complexity. More importantly, this research suggested developing personalized CAPTCHA challenges, as an alternative to the one-size-fits-all CAPTCHA that is widely used nowadays as it may have numerous benefits from the point of view of the users.

4 Discussion

There has been few but increasing amount of research related to the use of psychometry in investigating the usability of security. Majority of these researches have aimed to establish correlations between certain psychometric traits of users and their tendency in using security features, such as CAPTCHA tests [14] and password constructions [5, 12]. The studies presented in the previous section support similar trends in this area.

Specifically, personalization of security features, which involves adjusting the system to accommodate specific users, has shown to have several benefits from both user and security perspectives. Personalization provides efficiency, by allowing users to digest information cognitively. In addition, it also reduces cognitive load, increases user acceptance of the security features, and ultimately enhances the overall user experience.

From a security standpoint, personalization could strengthen the security features, as malicious attackers would need to imitate a specific individual along with their unique personality traits, as opposed to a general person, which is necessary in order to circumvent a credible user model [18]. Recent studies [8, 9, 19] suggest that this trend of personalization by utilizing psychometry for usable security is likely to continue in the future. However, the sparseness of studies in this topic indicates that there is still a long way to go until psychometry could be widely applied in security features.

5 Conclusion

This paper has reviewed various techniques in psychometry that have been employed in the study of usable security. The dicussion presented above indicates that the use of psychometry has yielded positive outcomes in the field of usable security, facilitating the ultimate goal of making the most secure practice to be the most user-friendly approach. The relatively sparse research in this particular area suggests that there is significant potential for growth in the coming years, but there is still a long way to go until psychometry is widely used in security features.

References

[1] J. C. Bastien, "Usability testing: a review of some methodological and technical aspects of the method," *International Journal of Medical Informatics*, vol. 79, no. 4, pp. e18–e23, 2010.

[2] V. Z. Rusu, D. Quiñones, C. Rusu, P. Cáceres, V. Rusu, and S. Roncagliolo, "Approaches on user experience assessment: User tests, communicability and psychometrics," in *Social Computing and Social Media. User Experience and Behavior* (G. Meiselwitz, ed.), (Cham), pp. 97–111, Springer International Publishing, 2018.

[3] C. Carreira, J. F. Ferreira, A. Mendes, and N. Christin, "Exploring usable security to improve the impact of formal verification: A research agenda," *Electronic Proceedings in Theoretical Computer Science*, vol. 349, pp. 77–84, Nov. 2021.

[4] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, (New York, NY, USA), p. 152–158, Association for Computing Machinery, 1994.

[5] S. M. T. Haque, S. Scielzo, and M. Wright, "Applying psychometrics to measure user comfort when constructing a strong password," in *10th Symposium On Usable Privacy and Security (SOUPS 2014)*, (Menlo Park, CA), pp. 231–242, USENIX Association, July 2014.

[6] B. Cowley, M. Filetti, K. Lukander, J. Torniainen, A. Henelius, L. Ahonen, O. Barral, I. Kosunen, T. Valtonen, M. Huotilainen, N. Ravaja, and G. Jacucci, "The psychophysiology primer: A guide to methods and a broad review with a focus on human-computer interaction," *Foundations and Trends in Human-Computer Interaction*, vol. 9, no. 3-4, pp. 151–308, 2016.

[7] R. Focardi, F. L. Luccio, and H. A. Wahsheh, "Usable security for qr code," *Journal of Information Security and Applications*, vol. 48, p. 102369, 2019.

[8] G.-C. Yang, "Next-generation personal authentication scheme based on eeg signal and deep learning," *JIPS(Journal of Information Processing Systems)*, vol. 16, no. 5, pp. 1034–1047, 2020.

[9] Z. Alkhyeli, A. Alshehhi, M. Alhemeiri, S. Aldhanhani, K. AlBalushi, F. A. AlNuaimi, and A. N. Belkacem, "Secure password using eeg-based brainprint system: Unlock smartphone password using brain-computer interface technology," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1982–1987, 2022.

[10] P. Jaferian, K. Hawkey, A. Sotirakopoulos, M. Velez-Rojas, and K. Beznosov, "Heuristics for evaluating it security management tools," in *Proceedings of the Seventh Symposium on Usable Privacy and Security*, SOUPS '11, (New York, NY, USA), Association for Computing Machinery, 2011.

[11] Y. Engeström, *Activity theory and individual and social transformation*, p. 19–38. Learning in Doing: Social, Cognitive and Computational Perspectives, Cambridge University Press, 1999.

[12] L. A. Loos, M.-B. C. Ogawa, and M. E. Crosby, "Cognitive variability factors and passphrase selection," in *Augmented Cognition. Human Cognition and Behavior* (D. D. Schmorrow and C. M. Fidopiastis, eds.), (Cham), pp. 383–394, Springer International Publishing, 2020.

[13] J. B. Rotter, "Generalized expectancies for internal versus external control of reinforcement.," *Psychological monographs*, vol. 80(1), pp. 1–28, 1966.

[14] M. Belk, C. Fidas, P. Germanakos, and G. Samaras, "Do human cognitive differences in information processing affect preference and performance of captcha?," *International Journal of Human-Computer Studies*, vol. 84, pp. 1–18, 2015.

[15] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in Cryptology — EUROCRYPT 2003* (E. Biham, ed.), (Berlin, Heidelberg), pp. 294–311, Springer Berlin Heidelberg, 2003.

[16] R. Riding and I. Cheema, "Cognitive styles—an overview and integration," *Educational Psychology*, vol. 11, pp. 193–215, 01 1991.

[17] A. R. Rezaei and L. Katz, "Evaluation of the reliability and validity of the cognitive styles analysis," *Personality and Individual Differences*, vol. 36, no. 6, pp. 1317–1327, 2004.

[18] C. Fidas and A. G. Voyiatzis, "On users' preference on localized vs. latin-based captcha challenges," in *Human-Computer Interaction – INTERACT 2013* (P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler, eds.), (Berlin, Heidelberg), pp. 358–365, Springer Berlin Heidelberg, 2013.

[19] A. Constantinides, M. Belk, C. Fidas, R. Beumers, D. Vidal, W. Huang, J. Bowles, T. Webber, A. Silvina, and A. Pitsillides, "Security and usability of a personalized user authentication paradigm: Insights from a longitudinal study with three healthcare organizations," *ACM Trans. Comput. Healthcare*, vol. 4, no. 1, 2023.

Web application session management security

Farjad Ali
farjad.ali@aalto.fi

Tutor: Aleksi Peltonen

Abstract

This paper discusses the current state of security in web application session management. Specifically, it highlights the significance of web session protection and the potential risks of session hijacking, which can result in data breaches, monetary loss, and personal harm. The paper presents some vulnerabilities that may lead to session hijacking, the focus is specifically on configuration faults. The paper also presents solutions developed by both academia and industry to address and tackle configuration faults including dynamic analysis tools, and a collection of industry best practices in terms of session management security. Additionally, it discusses a specific analysis method proposed by academia, its limitations and proposes some future research directions based on it. Finally, the paper concludes by provides some concluding remarks.

KEYWORDS: session management, web application, security, OWASP, attack tree, session, session hijacking, configuration faults

1 Introduction

Web applications are an integral part of our daily lives, their use span from online shopping to healthcare services. With the evolution of web-

sites from static pages to full-front applications with complex logic and functionalities, mechanisms known as sessions are required. Sessions are a key component that allows users to authenticate, access preferences, access personalized data and maintain a state throughout their interaction with the application. Therefore, session management is a major aspect of current web applications. Almost a fifth of all cyberattacks target web application vulnerabilities [26], hence, ensuring security for session management is crucial to guarantee a web application's privacy, confidentiality, and overall reliability [13].

This paper aims to review the current state of session management security in web applications by examining the trends in web application security research in order to explore gaps between industry and academia and provide directions for future research. Specifically, this paper discusses vulnerabilities regarding session management security and compares solutions developed by both academia and industry in order to tackle them. By doing so, we hope to provide a comprehensive understanding of session management security in web applications and address the importance of its issues to ensure the privacy, confidentiality, and reliability of web applications.

The rest of the paper is organized as follows. Section two explains the functioning of HTTP sessions, session management, and configuration faults that may cause a vulnerabilities. Section three illustrates mitigations to configurations faults, illustrating solutions developed by both industry and academia. The fourth section discusses these solutions. Finally, the last section provides some concluding remarks.

2 Web application sessions

Web applications use a stateless protocol, HTTP/HTTPS, to communicate with web browsers. This implies that web applications that require tracking a state throughout their lifecycle need to have an additional mechanism to implement it. Currently, this mechanism is known as sessions.

2.1 Session mechanism

Sessions allow grouping HTTP/HTTPS requests with a state for the web applications [12]. Sessions allow an application to implement functionalities such as asking for authentication only once, authorization, storing

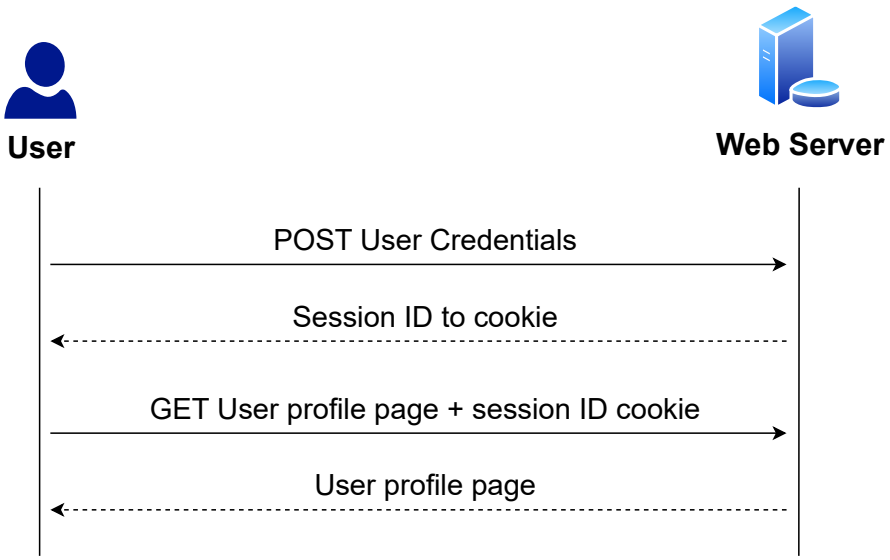


Figure 1. Interaction between a web server and user during authentication [16]

personal UI preferences, and keeping track of the user application data within each session. As illustrated in Fig. 1, at session generation, which typically occurs at either log-in or first visit, the web server saves the session's ID on the user's browser as a cookie. The cookie, with each user request, will be sent back to the server [12], allowing it to keep track of the session. Sessions can be used for unauthenticated and authenticated web applications. Session management refers to managing and maintaining sessions. This typically includes tasks related to authentication, authorization, session tracking, session termination, and session protection.

2.2 Session security

Since sessions are the main mechanism used to maintain a state in web applications, including uses in social media, government, healthcare, and banking, it is crucial to have good protection on the session IDs, especially on authenticated web applications. Plenty of research [14][18][19] has been done to find potential attacks. The most important of them is hijacking, which allows an attacker to retrieve a valid session ID for an active session. When this happens, the attacker can easily impersonate a user by including the stolen session ID for an authenticated HTTP/S request to the server [25]. As illustrated in Fig. 2, the server has no other way to authenticate a user that is sending the request other than using the session ID and, if it is valid, will respond with the content requested. The consequences of an attack like this can be severe since an aggressor could carry out several operations posing as the victim for monetary gain,

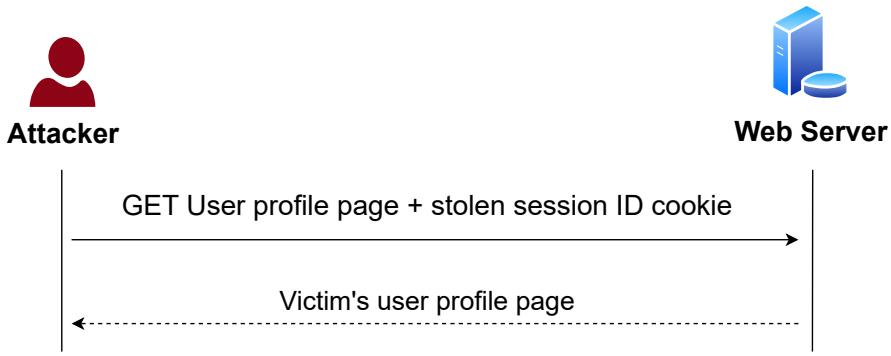


Figure 2. Interaction between a web server and an attacker [16]

cause personal harm, or access sensitive or confidential data.

2.3 Configuration faults

Session management needs to be implemented correctly by following the industry best practices [17][24] to have maximum security. In particular, developers need to focus on the following aspects of a web application: Session ID length, Session ID value, Cookie attributes/flags, Session ID transfer protocol, Communication protocol, Changing session ID after login, Session ID change or expiration after logout, Session ID timeout value, and Cache-Control value [10][11][15]. Fig. 3 illustrates an attack tree that allows a developer to quickly assess if a web application is vulnerable due to any development or configuration issues. The leaves are the specific causes or features while all the other nodes represent the threat [15]. Configuration issues are relatively easy to solve, and current best practices for web app development are constantly updated and maintained by organizations such as Open Web Application Security Project (OWASP) [23]. For session management best practices, OWASP provides a thoroughly compiled list of best practices with quite a clear explanation. [24] All the threats in the attack tree in Fig. 3 are preventable if the best practices are followed. Nonetheless, the human factor and design issues in application development influence the final outcome of an application in terms of security [9]. Vulnerabilities can be introduced involutarily by distractions or errors, and engineers can be not up to date with the latest practices [9].

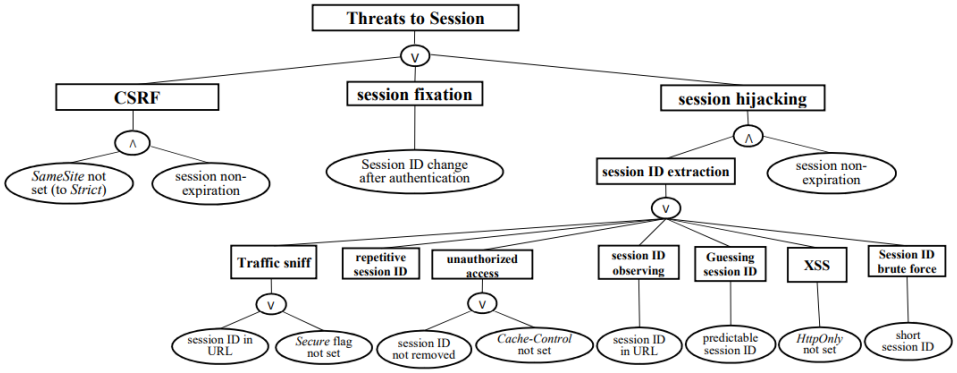


Figure 3. Attack tree to cause session hijacking [15]

3 Mitigation methods

This section illustrates some of the solutions developed by academia and industry to tackle the aforementioned problem. The first part details the OWASP best practices to avoid security issues for session management. The second part discusses a method developed by Garmabi *et al.* [15] and its software implementation to avoid configuration faults while developing a web application.

3.1 OWASP best pratices

OWASP is a nonprofit foundation that works to improve the security of web applications [23]. OWASP offers a series of guidelines for web application security in aspects spanning the whole lifecycle of a web application such as authorization, logging, denial of service, input validation, and session management. The collection of guidelines, namely the OWASP cheat sheet series [24], is written by several industry professionals with expertise in specific topics. The session management guidelines are a comprehensive collection of recommended practices to follow during the development and deployment of a web application to significantly improve session management security. The guidelines include aspects related to session IDs, such as length, entropy, duration, transmission, and storage. Additionally, it provides instructions to implement session attack detection using methods such as anomaly detection and log monitoring [24]. Nonetheless, human engineers still design and develop applications, which means there must be a failsafe to account for the human factor.

3.2 Software analysis

There are several software solutions that perform static and dynamic analysis that allows engineers to check the presence of common anti-patterns, code smells, and faulty configurations. Solutions for code include SonarQube [8], Checkmarx [4], and Fortify [5], while solutions for a deployed web app include OWASP ZAP [7], Netsparker [6], AppScan [2], Acutenix [1], and Burp Suite [3]. While extremely helpful, these applications have limitations as pointed out by Garmabi *et al.* [15], specifically, related to session management issues detection [15]. The mentioned applications fail to detect basic session management security issues including session ID timeouts, insecure communication method, and cache attributes [15].

New approach Garmabi *et al.* [15] propose a new method to perform session management security analysis. In this method, they first extract the session management features present in traffic between a web client and server to associate them with vulnerabilities and weaknesses. Subsequently, the vulnerabilities and weaknesses are mapped in an attack tree, as seen in Fig. 3, which allows for pinpointing specific vulnerabilities for a given attack. The proposed method also assesses the total risk for a specific attack listed in the attack tree. The risk is computed by $R = SP$ where S is the severity of the damage caused and P is the probability of the attack occurring.

Additional rules There are some further rules that need to be followed to compute the risk of an attack. Firstly, each feature represented as the attack tree leaf needs to be assigned the severity of its impact on the attacks. The severity levels used in this specific case are HIGH, MEDIUM, and LOW. There is no specific strategy to assign the severity values to the tree leaves, so an intuitive approach is used. For example, in the case of the "Session ID observing" attack, the only node under it represents the feature to indicate whether the session ID gets used in the URL. Therefore, the severity level of that particular leaf is estimated to be HIGH since it is the only aspect that allows somebody to perform the attack. To calculate the total risk, however, the leaves also need to have a probability value associated with them. This is trivial because they can be either present or not present which makes the probability either 1 or 0 respectively. Lastly, referring to the attack tree in Fig. 3, AND and OR operations are defined. If there is an OR or sum relation between two nodes, the sum of the risk

×	LOW	MEDIUM	HIGH
LOW	LOW	LOW	MEDIUM
MEDIUM	LOW	MEDIUM	MEDIUM
HIGH	MEDIUM	MEDIUM	HIGH

Table 1. Multiplication for risk levels [15]

between two nodes is defined as so $R = R_1 + R_2$. For the AND or multiplication relation, the multiplication of the risk between two nodes is defined in Table 1. The defined AND and OR operations have distributive properties, which means that for example the following expression $R_1(R_2 + R_3)$ can be reduced to $R_1R_2 + R_1R_3$.

The attack tree easily allows mapping detectable properties that lead to attacks. These properties are detected in the HTTP traffic between the client and the web server. The risk level assessment, combined with the attack tree, allows one to effortlessly compare the security levels between examined web applications and to quickly intervene.

Implementation Garmabi *et al.* implement this method using PHP and a MySQL database. The implementation receives the HTTP traffic as a text file input and extracts the features denoted in the leaves of the attack tree. Afterward, the implementation outputs the discovered weaknesses and the total risk level of the examined web application [15].

Method implementation performance The implemented tool has been confronted in terms of session management security measurement with three existing vulnerability scanners, namely Acunetix [1], Netsparker [6], and OWASP ZAP [7]. The target websites used for the comparisons were three simple vulnerable web applications, such as Buggy web application (BWAPP) [22], an extremely insecure test web application provided by OWASP. Results showed that in addition to computing the total risk of a web application, the method implementation also wastly outperformed the current tools used by the industry [15].

Limitations The implementation requires the traffic samples to be input as text files, which can be problematic in specific environments. One example where traffic sample gathering might be challenging is a highly secure environment with restricted access to traffic or web application servers. Another disadvantage is the extra configurations that need to be added to the existing web application deployments to capture the traffic

and convert it into a text file. These limitations are somewhat present in all of the tools that require a traffic analysis to evaluate the security properties of an application, including the ones used in the comparison.

4 Discussion

Session management is a crucial aspect of modern web applications due to its involvement in the entire life cycle. Therefore, it is necessary to guarantee solid security. Engineering guidelines [24][17], such as the cheat sheet series provided by OWASP have allowed engineers to have a shared knowledge of state-of-the-art practices in terms of web application security. Designing and developing a web application has become easier terms of security features since every guide is written and updated by industry experts on specific topics. Although the guidelines are broadly adopted by the industry, software engineering is still affected by the human factor. As a consequence, errors or distractions could bring a web application into an insecure state. Hence, there are several tools available to analyze an application and measure its security properties. Currently, for web applications, there is a gap between academia and industry in security assessment tools, especially for session management security measurement. Garmabi *et al.* [15] propose a contemporary method based on feature extraction, attack trees, and risk evaluation that substantially outperformed the industry tools and provided an overall risk assessment measurement for the examined application. The limitation of this method, however, is in its current implementation, which requires a sample of traffic between client and server in order to analyze it. Not only can this be challenging for applications deployed in severely controlled environments, but it also might require additional configuration in existing deployments. As a future direction, perhaps, a new implementation for this method would improve its effectiveness. The method could be implemented as a static analysis tool powered by data mining [21] or machine learning [20] that detects features directly from the source code of the application. This would allow the measurements to be made more immediately without the need to deploy the application and generate traffic between it and a user. Additionally, it would work instantly on every web application code base, since it would not require deployments or ad-hoc setups. Finally, this method would also allow integration of the static analysis in a CI/CD pipeline with gateway metrics, hence, automatically

analyzing and green-flagging versions that would meet the security requirements set by an administrator. A static analysis approach for the method would save the engineers several hours in manual testing since it would be completely automated by a CI/CD pipeline.

5 Conclusion

The aim of this paper was to find a gap between industry and academia in terms of web application session management security. We found a method proposed by Garmabi *et al.* [15] that performs substantially better than the ones available in industry. This indicates a gap that needs to be taken into consideration by industry and professionals. We also proposed a future research direction in order to improve the illustrated method in order to make it effective.

References

[1] Acutenix. https://www.acunetix.com/. Accessed: April 12, 2023.

[2] Appscan. https://www.hcltechsw.com/appscan. Accessed: April 12, 2023.

[3] Burp suite. https://portswigger.net/burp. Accessed: April 12, 2023.

[4] Checkmarx. https://checkmarx.com/. Accessed: April 12, 2023.

[5] Fortify. https://www.ndm.net/sast/hp-fortify. Accessed: April 12, 2023.

[6] Netsparker. https://www.invicti.com/. Accessed: April 12, 2023.

[7] Owasp zap. https://www.zaproxy.org/. Accessed: April 12, 2023.

[8] Sonarqube. https://www.sonarsource.com/products/sonarqube/. Accessed: April 12, 2023.

[9] Munir Ahmed, Lukman Sharif, Muhammad Kabir, and Maha Al-Maimani. Human errors in information security. *International Journal*, 1(3):82–87, 2012.

[10] Wafaa Al-Kahla, Ahmed S. Shatnawi, and Eyad Taqieddin. A taxonomy of web security vulnerabilities. In *2021 12th International Conference on Information and Communication Systems (ICICS)*, pages 424–429, 2021.

[11] Anuj Baitha and Smitha Vinod. Session hijacking and prevention technique. *International Journal of Engineering Technology*, 7:193, 03 2018.

[12] A. Barth. Http state management mechanism. RFC 6265, RFC Editor, 04 2011.

[13] Stefano Calzavara, Riccardo Focardi, Marco Squarcina, and Mauro Tempesta. Surviving the web: A journey into web session security. *ACM Computing Surveys*, 50:1–34, 03 2017.

[14] Siromani Duddu, Arigela Rishita sai, Ch. L S Sowjanya, G.Ramakoteswara Rao, and KarthikSainadh Siddabattula. Secure socket layer stripping attack using address resolution protocol spoofing. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 973–978, 2020.

[15] Nasrin Garmabi and Mohammad Ali Hadavi. Automatic detection and risk assessment of session management vulnerabilities in web applications. In *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*, pages 41–47, 2021.

[16] Md Maruf Hassan, Shamima Nipa, Marjan Akter, Rafita Haque, Fabiha Deepa, Md Mostafijur Rahman, Md Asif Siddiqui, and Md Hasan Sharif. Broken authentication and session management vulnerability: A case study of web application. *International Journal of Simulation: Systems, Science Technology*, 19, 04 2018.

[17] IBM. Best practices for session management. Available at: https://www.ibm.com/docs/en/was/8.5.5?topic=sessi
best-practices-using-http. Accessed: April 12, 2023.

[18] Vineeta Jain, Divya Sahu, and Deepak Tomar. Session hijacking: Threat analysis and countermeasures. 02 2015.

[19] Mainduddin Ahmad Jonas, Md. Shohrab Hossain, Risul Islam, Husnu S. Narman, and Mohammed Atiquzzaman. An intelligent system for preventing ssl stripping-based session hijacking attacks. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, pages 1–6, 2019.

[20] Ibéria Medeiros, Nuno Neves, and Miguel Correia. Dekant: A static analysis tool that learns to detect web application vulnerabilities. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA 2016, page 1–11, New York, NY, USA, 2016. Association for Computing Machinery.

[21] Ibéria Medeiros, Nuno Neves, and Miguel Correia. Detecting and removing web application vulnerabilities with static analysis and data mining. *IEEE Transactions on Reliability*, 65(1):54–69, 2016.

[22] OWASP. Broken web application. Available at: https://github.com/chuckfw/owaspbwa. Accessed: April 12, 2023.

[23] Open Web Application Security Project (OWASP). Available at: https://owasp.org/. Accessed: April 12, 2023.

[24] Open Web Application Security Project (OWASP). Session management cheat sheet. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Session_Managen Accessed: April 12, 2023.

[25] Cisco Press, Andrew Whitaker, and Daniel P Newman. Penetration testing and network defense.

[26] Terranova Security. Cybersecurity statistics. Available at: https://terranovasecurity.com/cyl security-statistics/. Accessed: April 12, 2023.

An Analysis of Security Vulnerabilities in JWT Implementations and Proposed Mitigations

Dang Hai Luong
hai.luong@aalto.fi

Tutor: Aleksi Peltonen

Abstract

JSON Web Tokens (JWT) have emerged as a prominent authentication method for accessing remote critical systems, offering a convenient and scalable approach to secure data transmission. Despite its widespread adoption, JWTs, like any other authentication method, exhibits certain advantages and disadvantages, including security flaws. This paper delves into the design and inherent security issues of JWTs, providing a systematic analysis of the relevant Request for Comments (RFCs) and recent research papers on the usage of JWTs in software systems. This study highlights the strengths and vulnerabilities of JWTs, underscoring the importance of adhering to best practices to mitigate the associated risks. While JWTs possess flaws, effective solutions and strategies exist to address these challenges. By employing best practices and maintaining a proactive security posture, developers can harness the power of JWTs while minimizing potential security threats.

KEYWORDS: JWT, JWE, vulnerabilities, mitigations

1 Introduction

The JSON Web Token (JWT) is a widely adopted authentication standard for securing user sessions and authorization information. It functions as a secure means of transmitting data between two entities, such as an Application Programming Interface (API) server and a client application, by encoding user claims, such as identity and attributes, into a compact and self-contained structure [25]. The digital signature embedded within the JWT provides a secure method of verifying the authenticity of the token, thereby mitigating threats such as session hijacking, man-in-the-middle attacks, and replay attacks [18].

However, despite its widespread use and security features, JWT tokens are subject to security vulnerabilities that must be understood and properly addressed by developers. This requires a comprehensive examination of the token structure, signature algorithms, and potential attack vectors, as well as the implementation of appropriate mitigation strategies to ensure the secure use of JWT in authentication and authorization processes [1].

This paper comprehensively examines the various security vulnerabilities associated with JWT. The vulnerabilities discussed include improper storage of secret keys, the absence of expiration time, the absence of revocation mechanisms, and the predictability of token generation logic. Furthermore, the paper presents thorough and effective countermeasures to mitigate these risks and enhance the security of JWT implementations.

Section 2 describes the technical details of JWT tokens. Section 3 introduces the best usages and security issues of JWT tokens. Section 4 outlines the effective countermeasures that can be implemented against these attacks. Section 5 closes out the article with some final thoughts.

2 JSON Web Token structure

This section provides an in-depth analysis of the structural composition of JSON Web Tokens (JWT) and its underlying security model.

2.1 JavaScript Object Notation

JavaScript Object Notation (JSON) is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to

parse and generate. JSON is a text-based format, which means that it consists of key-value pairs written as strings. RFC 8259 [2] specifies that keys are always strings, while the values can be strings, numbers, objects, arrays, or a few other simple data types. Here's an example of a simple JSON object:

```
{
  "name": "John Doe",
  "age": 35,
  "email": "john.doe@example.com"
}
```

The widespread use of JSON as a data interchange format in modern web applications has led to its incorporation as the underlying structural format for JSON Web Tokens (JWT). The lightweight and easily readable nature of JSON makes it an ideal candidate for facilitating communication between client and server in these applications [24].

2.2 Content of JWT Tokens

A JSON Web Token (JWT) typically consists of three parts: a header, a payload, and a signature [22].

1. **Header:** The header defines the type of the token (JWT) and the signing algorithm used to secure the token. It is a JSON object that is Base64Url encoded to form a compact and URL-safe string.

2. **Payload:** The payload contains the claims, which are statements about an entity (typically, the user) and additional data. Claims can be used to represent user identity, roles, and permissions, as well as other information. The payload is also a JSON object that is Base64Url encoded.

3. **Signature:** The signature is used to verify that the sender of the JWT is who it claims to be and to ensure that the contents of the JWT have not been tampered with. It is generated using the header and payload and a secret key known only to the server, or a public/private key pair.

Figure 1 outlines an example JWT Token with headers, payload and signature parts.

The three parts are concatenated with dots, forming a compact and self-contained string, which can be transmitted securely between parties. The

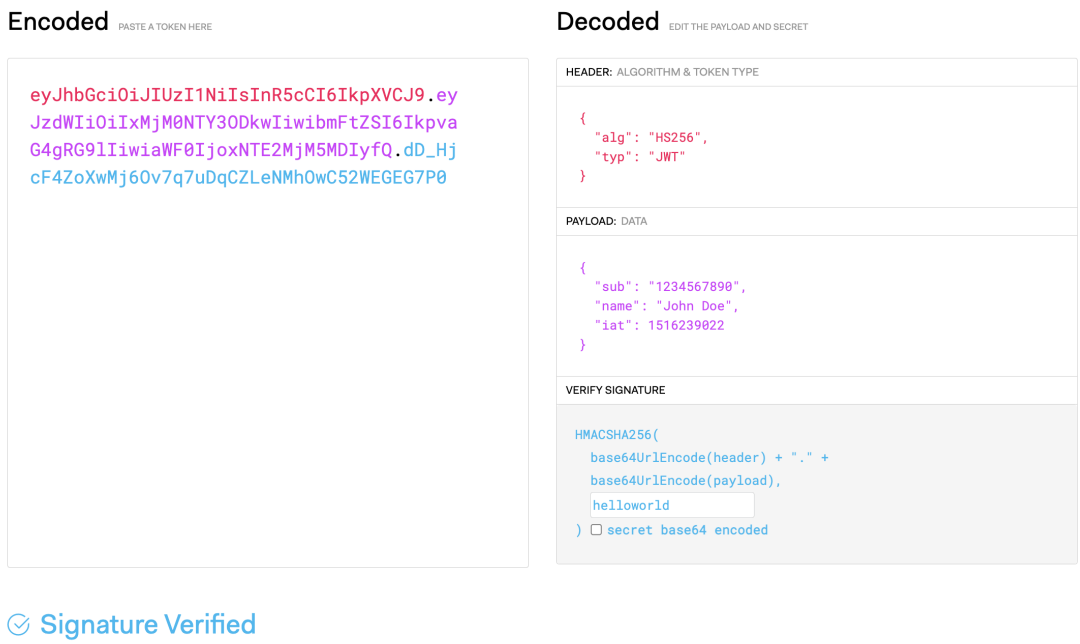


Figure 1. An example of JWT Token

recipient of the JWT can then decode the token and use the information contained within it for authentication and authorization purposes [22].

2.3 JWT token claims

The claims in the payload of a JSON Web Token (JWT) are statements about an entity, typically a user, that can be used for authentication and authorization purposes. Claims can include information such as the user's identity, roles, and permissions, as well as other information that may be relevant to the application. There are three types of claims: registered, public, and private claims [11].

1. **Registered Claims:** These are a set of predefined claims, such as "iss" (issuer), "sub" (subject), "aud" (audience), "exp" (expiration time), and "iat" (issued at time), that are standardized and widely used.

2. **Public Claims:** These are claims that are defined by the application and are not part of the registered claims set. They can be used to convey additional information about the user or the application, such as the user's name, email address, or custom attributes.

3. **Private Claims:** These are claims that are used only between the parties involved in the JWT exchange and are not meant to be shared with

other parties.

The claims contained in the payload of a JWT are encoded as a JSON object and can be read and processed by the recipient of the token.

3 JWT Token usage

The utilization of JSON Web Tokens (JWT) as a mechanism for authentication and authorization in modern systems is a prevalent and powerful approach. However, it is crucial to employ proper usage strategies in order to mitigate potential security vulnerabilities and ensure the secure implementation of JWT in such systems.

3.1 Correct usage of JWT Tokens

JSON Web Tokens (JWT) should be used in a secure manner to ensure the protection of sensitive information and to prevent unauthorized access. The following are general best practices for using JWT:

1. **Signing:** JWT tokens should be signed using a secure signing algorithm, such as Hash-based Message Authentication Code with Secure Hash Algorithm 256-bit (HMAC SHA-256) [10] or Rivest-Shamir-Adleman (RSA) [14]. The secret key used for signing should be kept confidential and securely stored on the server.

2. **Encryption:** If the payload of the JWT contains sensitive information, it should be encrypted using a secure encryption algorithm, such as Advanced Encryption Standard (AES) [17]. The encryption key should also be kept confidential and securely stored.

3. **Validation:** The recipient of the JWT should validate the signature of the token to ensure that it was not tampered with during transmission. The recipient should also validate the claims contained in the payload, such as the expiration time and issuer, to ensure that the token is still valid.

4. **Expiration:** JWT tokens should have an expiration time set, after which they are no longer valid. This helps to prevent replay attacks,

where a token that has already been used is resubmitted for unauthorized access.

5. **Revocation:** JWT tokens should be revocable, so that they can be invalidated if necessary. This can be achieved by maintaining a list of revoked tokens on the server, or by using an OAuth2.0 revocation endpoint [6].

6. **Secure storage:** JWT tokens should be stored securely on the client, such as in an HTTP-only and secure cookie, to prevent theft by malicious JavaScript code.

By following these best practices, JWT tokens can be used in a secure and reliable manner for authentication and authorization purposes in modern web applications and APIs [20].

3.2 Malicious use of JWT Tokens

The utilization of JSON Web Tokens (JWT) as a form of authentication for remote access to critical systems is a widely adopted practice. The tamper-resistant nature of JWT, achieved through digital signatures, prevents malicious forgery and ensures the authenticity of the token [18]. However, tampering with the payload of the JWT can result in an invalid signature and grant unauthorized access to the critical system. Additionally, malicious manipulation of the JWT's issuance time claim, allowing for early use, can lead to malicious acts. Tampering with the user identifier stored within the JWT token can also result in the approval of requests on behalf of other users. This vulnerability was demonstrated in scenario number 5, test 8 of the JWT Token attack scenario. Furthermore, JWT tokens can potentially be utilized for the retrieval of sensitive information from a user's account or for the dynamic generation of data in mobile applications, highlighting the need for proper security measures to be in place. [3].

3.3 JWT Token vulnerabilities

If a JWT Token is not properly secured, an attacker can gain access to a user's account without having to provide any credentials. This is because the token is a self-contained structure that contains all the information

needed for authentication, including the user's identity and access rights [15]. Here are the popular vulnerabilities of JWT tokens:

1. **Insecure storage of secret key:** The secret key used to sign and verify JWT tokens must be kept secure. If an attacker gains access to this key, they can create their own tokens and impersonate a legitimate user [5].

2. **Lack of expiration:** JWT tokens do not have an expiration time by default. This means that if a token is stolen, it can be used indefinitely. To mitigate this, the JWT token should include an expiration time in the claims of the token and validate it on the server [12].

3. **Lack of revocation:** Once a token is issued, there is no built-in mechanism for revoking it. Attackers may use social engineering tactics to trick users into providing their JWT, such as phishing or pretexting. This implies that in the event of a user's account being compromised, it is infeasible to instantaneously inhibit the attacker from utilizing the tokens that have previously been issued [12].

4. **Predictable token generation:** Attackers can predict token generation if the secret key used to sign the tokens is not sufficiently random or is compromised in some way. To prevent this, it is important to use a strong and secure secret key and to rotate it periodically [4].

4 Prevention methods

In order to protect JWT Tokens, it is essential to understand the various aspects of the JOSE (JavaScript Object Signing & Encryption) collection of specifications which define the JWT token format [16]. To secure the transmission of JWTs with confidential information that is unencrypted, it is important to use encryption protocols that provide endpoint authentication, such as Transport Layer Security (TLS), an encrypted JWT and authentication of the recipient should be used [11]. A set of unguessable tokens should be created employing different modes for encryption and token signing listed in RFC 7515 [16, 8], while the "iss", "sub", and "aud" Header Parameter names should be registered for unsecured replicas of

these claims in encrypted JWTs [11].

Furthermore, a rule should be implemented that checks whether a JWT is encrypted before decrypting it [11]. To securely represent claims between two parties, a JWT should be used in OpenID Connect 1.0 [19] or OAuth 2.0 protocols [6], in a compact, URL-safe format. It is advisable to compare the size of the JSON MEC Access Token (JMAT) and the time required for generation and encryption of these tokens using a variety of encryption algorithms and signing modes. The JSON Web Encryption (JWE) approach provides security by encrypting data with a secret key, making it inaccessible to rogue clients or Man-in-the-middle attackers. In the event that an attacker attempts to tamper with the payload, the JWE Authentication Tag verification will fail, causing the server to reject the signature [16].

To further protect JWTs, a hash function that is collision-resistant and produces a unique hash for each object should be used. Rules should be implemented that ensure only authorized JWTs can be used to make Application Programming Interface (API) requests, and that only authorized users can access JWTs [11].

4.1 What kind of encryption algorithm is used for JWT Token security?

In order to ensure the security of JWT Token data, encryption algorithms are used to protect the data from unauthorized access. [21] JSON Web Encryption (JWE) is the primary encryption algorithm used for JWT Token security [16, 21]. JWE is defined in the RFC 7516 [9] and uses a symmetric encryption key to protect the JWT data from unauthorized access [21]. In addition, the symmetric encryption key is derived from the symmetric encryption key used to encrypt the JWT. The key generation process can be done together with the token generation process, and the symmetric encryption key used is the same as the symmetric encryption key used to encrypt the JWT [16]. Unencrypted JWTs, however, should only be used when a JWT is cryptographically protected by other means, and JWT libraries should not consume or generate JWTs using the "none" algorithm unless explicitly requested by the caller [21].

4.2 Use rate limiting and request logging to detect attacks and protect data

In order to protect data and detect malicious attacks, rate limiting and request logging are two important steps. Rate limiting is a technique that is used to restrict the rate of requests sent from a particular source to a server. This technique is used to protect against malicious attacks by limiting the number of requests that can be sent from a specific source in a given period of time [13].

Request logging is also used to protect data and detect malicious attacks. It is the process of logging all requests sent to a server in order to monitor and detect suspicious activities. Using this method can be used to detect malicious requests, such as those sent by an attacker attempting to gain access to the server. It can also be used to detect suspicious activities, such as a user attempting to access data they are not authorized to access [7, 23].

Together, rate limiting and request logging can be used to protect data and detect malicious attacks. They can help to ensure that only authorized requests are sent to the server and that suspicious activities are identified and dealt with quickly and effectively.

5 Conclusion

The utilization of JSON Web Tokens (JWT) has become a prevalent method for authenticating and authorizing access within web applications. the token offers a standardized format for exchanging information between parties, thus facilitating integration with various systems. Despite its advantages, the implementation and usage of JWT are not immune to security vulnerabilities.

Incorrect implementation and usage of JWT can lead to security risks. Nevertheless, these risks can be mitigated through the implementation of appropriate security measures. The utilization of strong signing algorithms can significantly enhance the security of JWT. Furthermore, the implementation of token expiration and revocation mechanisms can prevent replay attacks and guarantee the invalidation of tokens as necessary.

In conclusion, JWT has become a widely adopted method for authentication and authorization in web applications due to its standardization and ease of integration. Although it is not without its security vulnerabilities,

these risks can be effectively mitigated through proper implementation and the adoption of appropriate security measures.

References

[1] Salman Ahmed and Qamar Mahmood. An authentication based scheme for applications using JSON web token. In *2019 22nd International Multitopic Conference (INMIC)*, pages 1–6, 2019. ISSN: 2049-3630.

[2] Tim Bray. The JavaScript object notation (JSON) data interchange format. Num Pages: 16.

[3] Rasa Bruzgiene and Konstantinas Jurgilas. Securing remote access to information systems of critical infrastructure using two-factor authentication. *Electronics*, 10(15), 2021.

[4] Dedipyaman Das, Sibi Chakkaravarthy Sethuraman, and Suresh Chandra Satapathy. A decentralized open web cryptographic standard. *Computers and Electrical Engineering*, 99:107751, 2022.

[5] Manik Lal Das and Navkar Samdaria. On the security of SSL/TLS-enabled applications. *Applied Computing and Informatics*, 10(1):68–81, 2014.

[6] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.

[7] Bartosz Jasiul, Rafal Piotrowski, Przemysław Bereziński, Michal Choraś, Rafal Kozik, and Juliusz Brzostek. Federated cyber defence system — applied methods and techniques. In *2012 Military Communications and Information Systems Conference (MCC)*, pages 1–6, 2012.

[8] Michael Jones, John Bradley, and Nat Sakimura. JSON web signature (JWS), 2015. Num Pages: 59.

[9] Michael Jones and Joe Hildebrand. JSON web encryption (JWE). Num Pages: 51.

[10] Dr. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, February 1997.

[11] Jones M., Bradley J., and Sakimura N. *JSON Web Token (JWT)*, 5 2015.

[12] Ryan Melton. Securing a cloud-native c2 architecture using sso and jwt. In *2021 IEEE Aerospace Conference (50100)*, pages 1–8, 2021.

[13] Jarmo Mölsä. Effectiveness of rate-limiting in mitigating flooding dos attacks. In *Communications, Internet, and Information Technology*, pages 155–160. Citeseer, 2004.

[14] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, November 2016.

[15] Alison PhD Munsch and Peter MBA Munsch. The future of api (application programming interface) security: The adoption of apis for digital communications and the implications for cyber security vulnerabilities. *Journal of International Technology and Information Management*, 29(3), 2021.

[16] Wojciech Niewolski, Tomasz W. Nowak, Mariusz Sepczuk, and Zbigniew Kotulski. Token-based authentication framework for 5g mec mobile networks. *Electronics*, 10(14), 2021.

[17] National Institute of Standards and Technology. Advanced encryption standard (AES), Nov 2001.

[18] Badr Eddine Sabir, Mohamed Youssfi, Omar Bouattane, and Hakim Allali. Authentication model based on JWT and local PKI for communication security in multi-agent systems. In Mohammed Serrhini, Carla Silva, and Sultan Aljahdali, editors, *Innovation in Information Systems and Technologies to Support Learning Research*, Learning and Analytics in Intelligent Systems, pages 469–479. Springer International Publishing, 2020.

[19] N Sakimura, J Bradley, Ping Identity, M Jones, B de Medeiros, and E Jay. Openid connect standard 1.0-draft 15, 2012.

[20] Y Sheffer, D Hardt, and M Jones. Json web token best current practices. *RFC 8725*, 2020.

[21] Y. Sheffer, D. Hardt, and M. Jones. *JSON Web Token Best Current Practices*, 2 2020.

[22] Krishna Shingala. JSON web token (JWT) based client authentication in message queuing telemetry transport (MQTT).

[23] Uttam Thakore. A quantitative methodology for evaluating and deploying security monitors, 2015.

[24] Ming Ying and James Miller. Refactoring legacy AJAX applications to improve the efficiency of the data exchange component. *Journal of Systems and Software*, 86(1):72–88, 2013.

[25] Gongxuan Zhang, Mingyue Zhang, and Xinyi Fan. Improvements based on JWT and RBAC for spring security framework. In Shui Yu, Peter Mueller, and Jiangbo Qian, editors, *Security and Privacy in Digital Economy*, Communications in Computer and Information Science, pages 114–128. Springer, 2020.

A Survey of Deep Learning Based Video Codecs

Henri Katvio
henri.katvio@aalto.fi

Tutor: Matti Siekkinen

Abstract

With resolutions growing in video the bandwidth requirements are ever growing. Transmitting video over the Internet takes a lot of the capacity. To combat this, video codecs are being developed to compress the video for lower bandwidth requirement while retaining the quality of the video. This paper takes a look at deep learning based video codecs to understand the challenges with and different approaches of deep learning video codecs.

KEYWORDS: deep learning, neural network, video, codec, HEVC, VVC, super resolution, frame interpolation

1 Introduction

As of 2023, more than 80% of the internet traffic could be attributed to video content [12]. Thus it is of utmost importance to develop more efficient video codecs. More efficient video codecs mean that the bitrate of the video should be lower without a reduction in perceived video quality. Additionally, Internet of Things (IoT) devices have become prevalent, with Machine to Machine (M2M) connections becoming all the more common in 2023 [6]. In 2023, it is estimated that there are 14.7 billion M2M connections. A lot of these connections are home security and video surveillance

systems, telemedicine and smart car navigation systems.

Traditional video codecs are constantly evolving to be more efficient. However, transformations on the video are not always obvious and can not always be implemented to traditional codecs. Deep learning allows for a few novel approaches, such as pixel probability modelling and auto-encoder [11]. This, however, also enables a novel adversarial attack possibility [4].

This paper explores the most important approaches to reducing the bitrate of the video with deep learning based video codecs as well as takes a deeper look at one of the proposed codecs. Additionally, an attack against deep learning based video coding is quickly presented.

This paper is organised as follows. Section 2 introduces the approaches to reduce the bitrate mainly concerning itself in the algorithms. Section 3 presents one deep learning based video codec more thoroughly, while section 4 introduces an adversarial attack on learning based video codecs and finally section 5 concludes the paper.

2 Approaches to Reduce Bitrate

As over 80% of the internet traffic is video, it is important that the bitrate of the video is reduced to as low as possible. Bitrate means the amount of bits the video takes per second of the video, measured in megabits per second. A common format for storing pictures and video is the YCbCr 420, a colour format that stores luminance and chrominance instead of the raw colour values like RGB does [13]. The bitrate for uncompressed YCbCr 420 video can be calculated as follows: to represent four pixels, there are four Y samples, one Cb sample and one Cr sample, a byte each. This means that six bytes are used, averaging 12 bits per pixel. In FullHD video at 30 frames per second, a single frame consists of 1920*1080*12 = 24883200 bits and a single second 30 times that, meaning 746496000 bits per second, or about 747 Mbps. Since the vast majority of wireless networks, or wired networks to end users cannot transmit that amount of data, a codec is needed that can compress the video to use less data.

There are numerous approaches to reduce the bitrate. [5] uses a predictive encoder, residual encoder and Huffman entropy coding. On the other hand, [12] uses more traditional methods, motion estimation, compression and compensation, transforms and quantization, entropy coding and

frame reconstruction. Each of these steps are done using neural networks. This section presents two different approaches to reducing the bitrate of the video used in deep learning based video codecs.

2.1 Traditional video codecs

The most used video codec in the internet is the H.264 or Advanced Video Coding (AVC) [7]. This can be seen for example with YouTube, where livestreaming is only possible using the AVC format [1]. AVC does a few things to reduce the bitrate [14]. A lot of the information is discarded through quantization, by analysing the information frequency of the frame and then removing unnecessary parts. Of course, since YCbCr takes less space than full RGB colour, it is used instead. The process of transforming from RGB colour space to YCbCr is called Chroma Subsampling. While those techniques work within the context of a single frame, compression is done in temporal dimension too. In AVC, some frames are full frames, but some are only a difference from the last full frame. This means that the these frames containing only motion vector deltas are a small fraction of the space of a full frame. When decoding the video, the delta is added to the last full frame to get a full frame for viewing.

2.2 Super-Resolution Neural Networks

One efficient way to reduce the bandwidth is to drop the resolution of the transmitted video. If the resolution is dropped from FullHD to HD, the required bandwidth for the same uncompressed format would drop with the same proportion than the resolution, meaning (1280*720) / (1920*1080) = 0.444. This means that the resulting bandwidth is less than half of the original bandwidth.

To still achieve the same quality and resolution at the viewers end, techinques such as super-resolution is needed. The idea behind super-resolution is to train a neural network, commonly a deep convolutional neural network (DCNN), to reconstruct the high-resolution image from the low-resolution one. An enchanced deep super-resolution network (EDSR) is presented in [10] that applies to single images. In the work, EDSR is created from few convolutional layers with 64 feature maps, residual blocks and a pre-trained x2 network, that can be replaced to gain different upscaling factors. As the algorithm is essentially conventional ResNet architecture with unnecessary modules removed, it can be trained and used

quicker and the model size is reduced. While the work is created for single images, it can be applied to video by computing the results for consecutive frames.

Nvidia has used the same idea with their deep learning supersampling (DLSS) and AMD with their FidelityFX Super Resolution (FSR). Both technologies show that upscaling with neural networks is both a feasible and a practical approach to improving image quality from a lower resolution image. It is possible to perform the operation quickly enough for framerates of the video. Downside of the technique is the still heavy cost of computing power required.

2.3 Frame interpolation

The bandwidth required to transmit the video is heavily impacted by the framerate of the video. If half of the frames can be dropped in the encoding stage and reconstructed during the decoding stage, a lot of the bandwidth could be saved. There are many ready solutions to do frame interpolation but they are often computationally and/or spatially heavy. For example, Super SloMo, presented in [8], takes up 158.4MB of space [16]. As the research has continued, however, more effective methods have been created that take less space, such as using convolutional neural network based spatial pyramids [16]. This methodology achieves generally either as good or better results as Super SloMo while taking only 9.0MB of space. Of course, the research into the topic continues and more advanced methods are always being developed. Today, there are various different commercial players working on the topic too, such as Runway.

3 Enhancing Versatile Video Coding With Deep Learning

Deep learning based video codecs can also be used to improve traditional video codecs not utilising deep learning. This section presents one such technology: WSE-DCNN-based in-loop filtering for VVC [2]. Before that, Versatile Video Coding is presented.

3.1 Versatile Video Coding

After the resolution of streamed video quickly rose to UHD or 4K and the average television in homes used that resolution, it became clear that more efficient codecs were needed. HEVC or H.265, follower of AVC im-

proved the bitrate for the same quality video by 50%, but it was not quite enough. After a Joint Video Exploration Team comprised of VCEG and MPEG joined forces and created a Joint Exploration Test Model demonstrating more than 30% improvement over HEVC, it was decided that it was a time for a new standard [3]. The resulting standard was named Versatile Video Coding, VVC for short. VVC is also known as H.266.

The format has inherited many high level features from the previous codecs, AVC and HEVC. As an example, the data is processed in coding tree units (CTU) that are further partitioned to coding tree blocks (CTB), one for each luma and chroma components [15, 9]. CTUs can be divided into coding units (CU). To enable random access (meaning the video does not need to always be watched from the start), instantaneous decoding refresh (IDR) or sometimes clean random access (CRA) pictures are used. A difference to previous codecs is that the spatial resolution can change in other places than at an IDR picture too. Reference picture resampling (RPR) allows for down- or upsampling of inter-coded pictures so that additional IDR are not needed for resolution changes mid-stream. VVC also adds support for bitstream extraction and merging (BEAM) operations, meaning that only a subpicture of the whole video can be transmitted at a time, an important feature for a 360° video, where user is looking at only a small portion of the whole picture at a time [3].

In VVC, the last step before acquiring the reconstructed frame is the in-loop filtering. In-loop filtering is not a concept that has been introduced specifically for VVC, but it was used in HEVC previously. The block is comprised of different filters combined together, all being applied to the frame. These filters are luma mapping with chroma sampling (LMCS), long deblocking filters, luma-adaptive deblocking, adaptive loop filter (ALF) and cross-component ALF [9]. The filters are called in-loop filters as they are applied in both encoding and decoding loops. Each of the filters serve a different purpose, but collectively they increase coding efficiency, reduce artefacts and blocking discontinuities, and enhance the reconstructed signal.

3.2 In-loop filter replacement with WSE-DCNN

Bouaafia et al. [2] have developed a way to replace the conventional in-loop filters of VVC with a so called wide-activated squeeze-and-excitation deep convolutional neural network (WSE-DCNN) model. It eliminates

compression artifacts to increase perceived visual quality and end user quality of experience (QoE). As only the in-loop filters of VVC are changed, visual quality could be improved without any increase in bandwidth.

The proposed model has six inputs, three for YUV components and three for quantization parameter and coding unit for luminance and chrominance. As outputs, the model gives the three components luma Y and chroma U and V. The structure of the model is beyond the scope of this paper, but it is available to read in [2].

In the paper, the model is trained offline with TensorFlow GPU framework using an NVidia RTX 2070 graphics card. As the loss function, mean square error is used and peak signal-to-noice ration (PSNR) is used for validation. Bjontegaard delta bit rate (BD rate) is used as the assessment metric. The researchers show that the BD rate reduction for luma Y component is 2.85%, while the reduction for chroma U and V components is 8.89% and 10.05% respectively. At the same time, the time overhead for encoding is increased to 122%. For decoding, the time overhead is increased to 1648.76% due to CPU-GPU memory copy operations [2]. It could therefore be noted that the increase in decoding makes the technology not yet suitable for resource constrained IoT devices. For encoding, the difference seems sufficiently small.

4 Reduction of Video Service Quality via Adversarial Attacks

As with all other machine learning based technologies, video codecs can be targets for attack too. Chang et al. [4], therefore have developed novel adversarial attacks against deep learning video codecs. Their goal is to either reduce the video quality, increase the bandwidth required or both simultaneously, depending on the attack.

The attack is carried out in the first stage of the pipeline from camera to the back-end, meaning the picture is modified before being encoded. Their assumption is this is done using malware or man-in-the-middle scheme. As the goal of the researchers is for the attack to last a longer period of time, the perturbations on the image should be imperceptible to visual examination as to hide the source of the attack.

Applying the perturbations in offline mode, where there is access to the whole video is not time constrained and the attack can be customised to the video. In online mode, there is however a time constraint and no

access to the whole video, but only a single frame at a time. This is the case in, for example, streaming. In this case, a universal perturbation designed beforehand offline is needed. This can be applied to the frame essentially in real time.

The researchers show that their methodology is indeed effective at reducing the quality of experience. Their attack can drop the PSNR by 2.30 - 3.24dB in different scenarios and increase the required bandwidth over 2x in offline scenarios and 1.52x even in unseen models in online scenarios. Additionally, the attack on video quality introduces visible artefacts and noise on the video after decoding. As of April 12th, 2023, a demonstration for the attack is available in https://sites.google.com/view/demo-of-rovisq/home .

5 Conclusion

As the few examples show, deep learning based video codecs can offer advantages over more traditional simple algorithm based ones. With many devices having a lot of computational power, their main idea is to exchange that power to reduced network bandwidth requirement. While this is true for many of the devices, such as smartphones and computers, many devices are computationally limited. These include many multimedia IoT devices which include only a small microcontroller but their functionality relies on sending video over internet. For example baby monitors and surveillance cameras belong to this category. For these, deep learning video codecs are not a good fit. In these environments, replacing just parts of existing video codecs with light pre-trained models, such as WSE-DCNN introduced in subsection 3.2, can offer promising alternatives. With the encoding being the only process done on the device, a 22% increase to encoding time could be acceptable.

Nevertheless, the research to these types of codecs has been ongoing mainly in very recent years and the technology is not yet matured to the point where traditional codecs could be given up entirely in coming years. Algorithm-based video codecs will continue to improve as well, together with deep learning based ones.

References

[1] Choose live encoder settings, bitrates, and resolutions.

[2] Soulef Bouaafia, Randa Khemiri, Seifeddine Messaoud, Olfa Ben Ahmed, and Fatma Ezahra Sayadi. Deep learning-based video quality enhancement for the new versatile video coding. *Neural Computing and Applications*, 34(17):14135–14149, Sep 2022.

[3] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, Oct 2021.

[4] Jung-Woo Chang, Mojan Javaheripi, Seira Hidano, and Farinaz Koushanfar. Rovisq: Reduction of video service quality via adversarial attacks on deep learning-based video compression. *Network and Distributed System Security Symposium 2023*, Dec 2022. arXiv:2203.10183 [cs].

[5] Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma. Deepcoder: A deep neural network based video compression. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, page 1–4, Dec 2017.

[6] Cisco Company. Cisco annual internet report (2018-2023) white paper. 2020.

[7] H Y El-Arsh, A S Elliethy, A M Abdelaziz, and H A Aly. Performance comparison among popular implementations of h.264 encoders. *IOP Conference Series: Materials Science and Engineering*, 1172(1), Aug 2021.

[8] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik G. Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. *CoRR*, abs/1712.00080, 2017.

[9] Marta Karczewicz, Nan Hu, Jonathan Taquet, Ching-Yeh Chen, Kiran Misra, Kenneth Andersson, Peng Yin, Taoran Lu, Edouard François, and Jie Chen. Vvc in-loop filters. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3907–3925, 2021.

[10] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. (arXiv:1707.02921), Jul 2017. arXiv:1707.02921 [cs].

[11] Dong Liu, Yue Li, Jianping Lin, Houqiang Li, and Feng Wu. Deep learning-based video coding: A review and a case study. *ACM Computing Surveys*, 53(1):1–35, Jan 2021. arXiv:1904.12462 [cs, eess].

[12] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. (arXiv:1812.00101), Apr 2019. arXiv:1812.00101 [cs, eess].

[13] Charles Poynton. Merging computing with studio video: Converting between r'g'b' and 4:2:2. Feb 1998.

[14] International Telecommunication Union. Advanced video coding for generic audiovisual services, Aug 2021.

[15] International Telecommunication Union. Versatile video coding, Apr 2022.

[16] Jiankai Zhuang, Zengchang Qin, Jialu Chen, and Tao Wan. A lightweight network model for video frame interpolation using spatial pyramids. In *2020 IEEE International Conference on Image Processing (ICIP)*, page 543–547, Oct 2020.

Comparative of security tools for the cloud

Iikka Näsälä
iikka.nasala@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

The cloud-native technology Kubernetes is getting more and more attention from businesses and organizations. It is a powerful tool that is redefining the way that applications are built and deployed. But regardless of its popularity, it comes with vulnerabilities and many misconfiguration possibilities. The goal of this paper is to investigate various security tools for Kubernetes, including Kube-bench, Kube-hunter, Clair, Istio, Calico, and Falco, and evaluate their capabilities in securing different layers of Kubernetes infrastructure. By employing multiple security tools users benefit from each tool's strengths, leading to a robust, defense-in-depth approach that secures every layer of Kubernetes. The paper is a literature review that mostly investigates the documentation of the tools, in addition to other scientific articles and surveys.

KEYWORDS: Cloud Computing, Kubernetes, Cybersecurity, Security tools, Best practices

1 Introduction

The cloud is a broad term referring to a global network of remote servers and data centers that are accessible over the Internet. These servers are

used for cloud computing, which according to the U.S. National Institute of Standards and Technology (NIST) is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. Nowadays, cloud computing is becoming increasingly popular among businesses, organizations, and other private users, due to its flexibility, scalability, and relatively low cost of use compared to traditional computing [2]. The way the cloud provides scalable on-demand delivery of information technology (IT) resources via the Internet brings an easy approach to storing, sharing, and organizing data without having to manage or maintain the infrastructure [3].

However, cloud computing does not come without concerns, with security being the main one. Many businesses and educational organizations are cautious about depending entirely on cloud computing and transferring their digital resources to third-party vendors. Even if the cloud service provider (CSP) trusts its customers, the users may not trust one another. [4] The security of cloud-native technology Kubernetes (K8s) was declared a major priority in 2022 [5].

This paper examines the problem of the lack of consensus on the defense approach for security threats connected to a cloud-native technology Kubernetes. More specifically, the paper explores different types of open-source security tools for K8s and investigates their compatibility. The paper also describes known K8s vulnerabilities and lists the best security practices to be used when deploying a K8s cluster.

The paper is a literature review and is organized into six sections as follows. Section 2 provides an overview of cloud security and briefly discusses the Kubernetes framework. The third section presents best practices for safe K8s cluster deployment and touches upon vulnerabilities that K8s users should be aware of. Section 4 lists and describes popular open-source security tools, with different focus areas, for K8s. Finally, section 5 analyses the tool usage, and section 6 concludes the literature review.

2 Cloud security overview

As businesses and organizations transition to cloud-based solutions for storing and processing data, understanding the security requirements becomes progressively crucial [4]. Regarding this problem, cloud security has its own tools, protocols, policies, best practices, standards, and compliance procedures for protecting against cyber threats [6]. Although these methods are impressive, they do not come without difficulties.

In the cloud computing environment, the customers take care of their own applications, data, and access control, but no longer own the infrastructure, and thus the traditional security architecture does not apply anymore [7]. A compromise in the availability, integrity, or confidentiality of cloud resources may generate new security issues [8]. Hence it is important for businesses and organizations to have a clear consensus about their responsibilities and what security measures to implement to protect their data [7].

In the following chapters, the paper will shift its focus to the cloud-native technology K8s, as it is the most prominent and popular option for orchestrating container-based solutions, according to Datadog's container report (2022). [9]

3 Kubernetes

Kubernetes is an open-source container orchestration platform that simplifies declarative configuration and automation for containerized workloads and services. It provides the user with a framework with many useful qualities, such as service discovery and load balancing, storage orchestration, secret and configuration management, self-healing, and more. [10] K8s's primary design objective is to facilitate the deployment and management of complex distributed systems while maintaining the advantages of container usage [11].

However, setting up a Kubernetes cluster and configuring it to deploy the desired applications can be challenging and gets more difficult the more complex the distributed systems are. This often results in configurations that focus more on functionality rather than security. Nevertheless, security should be one of the top priorities. The following subsection will shed some more light on the Kubernetes architecture, in order to deepen the understanding of the various threat-prone surfaces.

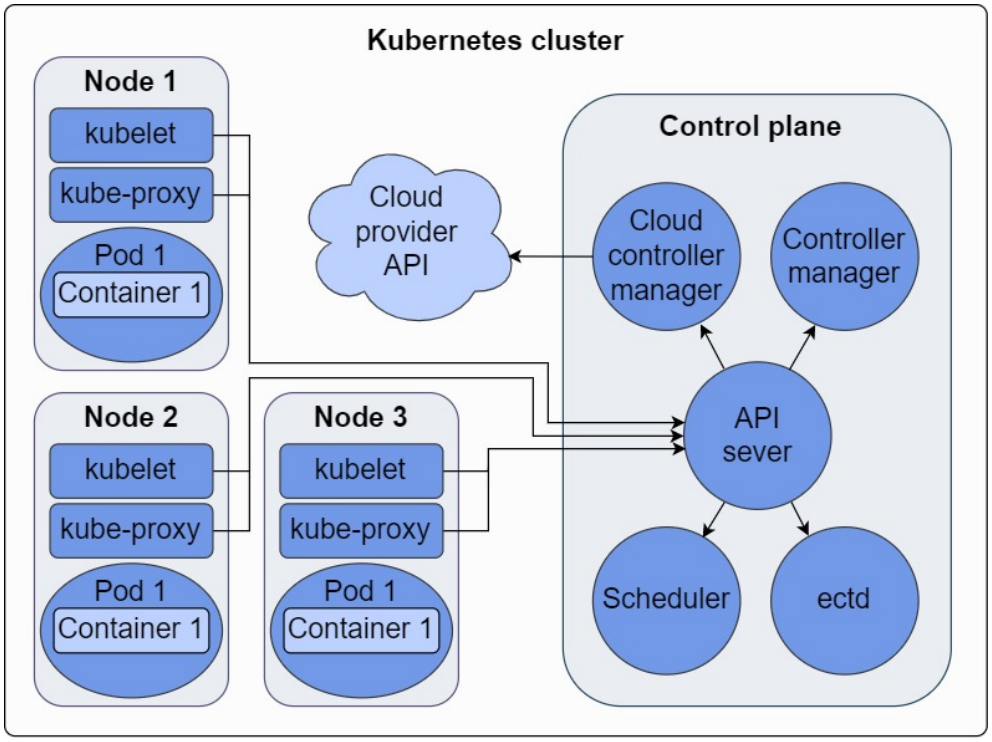


Figure 1. Kubernetes cluster.

3.1 Components

The main purpose of Kubernetes is to deploy, control, and scale containerized applications. This is done by running the containerized workloads on Pods, which are the basic units of K8s deployment. By using Pods, it becomes feasible to logically structure and segregate containers that operate collectively. Nodes are the element that handles the workloads using container-runtime, kube-proxy, and kubelet. [10] Container-runtime refers to the software that runs the containers e.g. Docker, Containerd, or CRI-O. Every node in the cluster runs a kube-proxy, a network proxy that takes care of inside and outside communications, as well as the network rules on nodes in addition to acting as a load balancer. Kubebelet on the other hand is an agent that manages the containers running in the Pod. It also takes care of the communication between the application programming interface (API) server, also known as kube-apiserver. [12] New Pods are scheduled to run on a node in a Kubernetes cluster, as figure 1 demonstrates. These nodes and Pods are managed by the control plane, consisting of etcd, kube-apiserver, kube-scheduler, kube-controller-manager, and cloud-controller-manager. The etcd is a key-value store that contains all data related to the cluster's status. Kube-apiserver is an interface used by admins to control Kubernetes and kube-controller-manager is a

component that handles various control processes, e.g. monitoring and responding to Pod failures. The cloud-controller-manager also runs controller processes, but specific to the cloud provider. It is used to link the cluster to the cloud provider's API. [10]

3.2 Vulnerabilities and best practices

Kubernetes can be easily misconfigured and exposed to attackers. The following subsections will discuss some common vulnerabilities and the corresponding security best practices that every user should know.

3.2.1 Defense-in-depth

K8s deployment can be divided into four layers: (a) the code, (b) the containers the code runs on, (c) the cluster, and (d) the cluster's infrastructure, whether it is on-premise or in the cloud. Securing all of these layers individually is recommended according to the defense-in-depth cybersecurity model. This strategy recognizes the possibility of some of the defenses being breached and instead of counting on a singular security perimeter, it employs additional barriers to thwart an attacker's advancement. [10, 13]

3.2.2 Running containers as non-root

Containers should be run as a non-root user and all privileges should be kept as restrictive as possible. For this purpose, when building an image a service user with minimal permissions should be created and used for running the application. This complicates the attacker's advancements since even if they get inside a container, they still have to gain the root permissions in order to get access the host. Also, the Pod configuration should not allow any unnecessary privileges, since the image configuration can be overwritten by the Pod configuration. [10, 13]

3.2.3 Managing users and permissions

Kubernetes does not manage users natively. Normal user accounts are not represented by objects nor can they be added through an API call. Instead, the addition of users to the cluster is done by importing a list of users or by generating a client certificate for a specific user to access the K8s API server. [10] Any user with a legitimate certificate issued by the cluster's certificate authority (CA) is regarded as authenticated. If this user's identity is stolen, the adversary gets all the privileges the user had. For minimizing the damage, it is important to limit privileges as much as possible, as suggested by the zero-trust cloud approach. [12, 14]

This can be done with role-based access control (RBAC). It is used for regulating access through set permissions in namespaces, a method for dividing cluster resources among numerous users, groups, or applications inside a cluster. Permission should be set for humans, but also service accounts. [10, 15]

3.2.4 Official registries and image scanning

The first stage in the continuous integration and continuous deployment (CI/CD) pipeline is to produce a secure image. Code and libraries should only be downloaded from trusted sources. Unofficial registries and base images can contain malware which might enable an attacker to compromise the container and gain access to the underlying infrastructure. After breaking out of the container the attacker can easily obtain data from the file system or the host's volumes. Further, the kubelet's authentication tokens and certifications, which it uses to communicate with the kube-apiserver can now be compromised through the kubelet's configuration files. Also, unnecessary dependencies should be removed and smaller base images with only the essential tools should be used. Along with using trustworthy sources, image scanning is an important practice for securing containerized applications. [15] Procedures such as identifying vulnerabilities, out-of-date libraries, or configuration errors, can be carried out easily with image scanning. Security tools, such as Clair [16], Trivy [17], and Kube-hunter [18], have a database of constantly evolving vulnerabilities that they use for scanning the image.

3.2.5 Allowing only the necessary connections

In Kubernetes, every Pod can communicate with each other by default. As a consequence, authorized connections must be defined, permitting only the necessary connections, and the cluster's internal communications must be encrypted. [19] Network Policies are a K8s resource, which enables configuring the desired communication rules between the Pods, on the network level. To implement Network Policies a Container Network Interface (CNI) must be chosen. [10] On the other hand, configuration on the service level can be done by adding a service mesh, an infrastructure layer enabling the user to add observability, traffic management, and security, to the desired application. Moreover, a service mesh often takes care of more complicated operational needs such as end-to-end authentication, rate restriction, access control, encryption, and A/B testing. [15] Popular service meshes for K8s include Istio [20], Consul [21], and Link-

erd [22].

3.2.6 Harden the control plane

As mentioned in the components section, the control plane is a central part of Kubernetes and needs to be well-protected because of its vulnerable capabilities. Important procedures to follow when securing the control plane include configuring TLS encryption, establishing reliable methods for authentication, preventing unwanted modifications to kubeconfig files, and disabling internet access along with other unnecessary networks. [10] The most crucial component of the control plane to protect is the etcd key-value storage. It stores secrets and other K8s configuration data. It can be run inside or separately managed outside of the cluster, but nevertheless, it should be isolated behind a firewall and allowed to access only from the API server with proper authentication. [15] The secrets stored in the etcd are only base64 encoded strings and can be decoded by anyone with permission to read the secrets. It is recommended to use a third-party key management service (KMS) to keep sensitive data outside of the cluster or at least configure K8s to encrypt data at rest. [10]

3.2.7 Audit logging

Kubernetes audit logs document cluster actions, memory, and CPU usage, but are not enabled by default. These attributes are not enough to assure full system security, hence logging should be expanded to every level of the system and should include information about critical actions, such as API request history, changes in privileges and permissions, network traffic, and operating system calls in a cluster. Once collected, these logs ought to be combined and analyzed by a security tool to ensure the system's security. [15]

4 Security tools for Kubernetes

While Kubernetes offers many built-in security features, it is also highly configurable. This means that there are many opportunities to unintentionally create vulnerabilities in a cluster. This section provides a more thorough examination of some static and dynamic K8s security tools.

4.1 Kube-bench

Kube-bench is an open-source analysis tool that examines Kubernetes clusters against the Center for Internet Security (CIS) Kubernetes benchmarks and provides a report of any security issues or misconfigurations found. It runs mostly static, but can also be used for scanning an active cluster. The CIS benchmarks are a set of security configuration guidelines that are designed to improve Kubernetes deployments. [23]

4.2 Kube-hunter

Unlike Kube-bench, Kube-hunter is a security tool that focuses on identifying potential security vulnerabilities and threats in Kubernetes environments by dynamically searching for weaknesses in the system. It is a pragmatic, open-source tool from Aqua Security, with a wide range of passive and active tests. These tests vary from accessing the API server and discovering open ports to retrieving log information and privilege escalations. [18]

4.3 Clair

Clair is a Red Hat-initiated open-source static analysis tool for monitoring container security. It scans the container image on each layer to produce a list of all the dependencies required by the image. Then after comparing each dependency to public vulnerability databases it notifies the user of flaws that may pose a threat. One aspect that sets Clair apart from other scanners is that it recognizes which of the current container layers are vulnerable when new vulnerabilities are uncovered and added to the databases. It does it without having to re-scan the system and notifies the user. [16]

4.4 Istio

As briefly explained earlier, Istio is a service mesh, an infrastructure layer for managing communications between microservices. Its basic architecture consists of a sidecar proxy in each Pod and the Istiod, which is a control plane component for managing and injecting all the proxies into the Pods. These proxies work together to create a mesh network that filters network traffic between the microservices. In addition to traffic management Istio also provides security features, such as authentication, autho-

rization, and communication encryption, and observability features, such as logging, tracing, and monitoring to view how service activity affects both upstream and downstream performance. [20]

4.5 Falco

Falco is an open-source runtime security tool made to watch over and spot malicious activity, such as privilege escalation, unanticipated socket changes or network connections, mode and ownership changes, and altered login binaries in Linux and Kubernetes environments. Falco was initially created by Sysdig, but was given to the CNCF and is currently being incubated. Falco also provides alerts and notifications to help users respond quickly to potential threats. [24]

4.6 Calico

Calico is a highly scalable, high-performance, open-source networking and network security solution for Kubernetes. Its advanced security features include network encryption, threat detection, and fine-grained network policy enforcement using Kubernetes NetworkPolicies or Calico's own policy language. In addition to network security, Calico also provides service load balancing and observability tools like flow logging and packet capture. Calico is developed and maintained by Tigera. [25]

5 Analysis

Kubernetes makes it possible to easily deploy, scale, and manage containerized applications. However, with this power comes also the need for robust security measures. For this purpose, various security tools have emerged. Regardless of these tools simply reporting issues, rather than attempting to fix them, they are still a valuable source of information for the users to better understand and secure the Kubernetes environment.

This paper covers a range of tools from static to dynamic analysis and from container security to network scanning. These tools are strong by themselves, but will also integrate well together. For example, a possible way to provide a more comprehensive networking and security solution for Kubernetes is to combine Istio's secure Pod-to-Pod or Service-to-Service communication with Calico's enforced network policies to prevent unauthorized access to the cluster. "Calico policy integrates with Istio to allow

you to write policies that enforce against application layer attributes like HTTP methods or paths as well as against cryptographically secure identities" [25]. While there are some overlappings, such as load balancing and observability tools, between the features provided by Calico and Istio, they have different focus areas and are thus often used together.

Another security tool that can be integrated to protect K8s, at the same time as Calico and Istio, is Falco. Where Calico and Istio are in charge of securing the networks, Falco provides container runtime security by monitoring the Kubernetes cluster for potential security threats. Falco is highly configurable and can be customized with user-defined rules to meet the specific needs of different environments. [24] Where Falco detects runtime behavior anomalies, it does not check the configuration of the Kubernetes cluster. If there are any possibilities of misconfigurations a tool like Kube-bench should be used to ensure that the cluster is configured securely and that known vulnerabilities have been addressed.

Each of these security tools has its own set of features and capabilities that address specific areas of security, such as network security, runtime security, or configuration management. By utilizing a combination of security tools, users can leverage the unique strengths of each tool, achieving a comprehensive defense-in-depth approach that secures every layer of Kubernetes.

6 Conclusion

This paper reviewed a set of common security tools for Kubernetes and evaluated their usage together. The paper clearly demonstrates that Kubernetes can be a complex environment to configure perfectly and monitoring it for security purposes is highly recommendable. The paper also provided an overview of security best practices, to illustrate the state of Kubernetes security.

Kubernetes security should be approached with a defense-in-depth strategy that involves securing every layer of the Kubernetes infrastructure. This means that users should use a combination of security tools that work together to provide a comprehensive security solution. The tools discussed in this paper cover a range of security features and can help users better secure their Kubernetes clusters.

References

[1] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing," 2011.

[2] J. Surbiryala and C. Rong, "Cloud computing: History and overview," in *2019 IEEE Cloud Summit*, pp. 1–7, IEEE, 2019.

[3] R. Mogull, J. Arlen, F. Gilbert, A. Lane, D. Mortman, G. Peterson, M. Rothman, J. Moltz, D. Moren, and E. Scoboria, "Security guidance for critical areas of focus in cloud computing v4. 0," *Cloud Security Alliance*, 2017.

[4] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information Sciences*, vol. 305, pp. 357–383, 2015.

[5] A. Mayr, "Kubernetes in the wild report 2023." `https://www.dynatrace.com/news/blog/kubernetes-in-the-wild-2023/`, Feb 2023. [Online] Accessed: Apr. 6, 2023.

[6] Z. Talab, "Traditional security vs. cloud security overview." `https://www.developer.com/security/`, December 2021. [Online] Accessed: Feb. 25, 2023.

[7] G. S. Puri, R. Tiwary, and S. Shukla, "A review on cloud computing," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 63–68, 2019.

[8] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications*, vol. 79, pp. 88–115, 2017.

[9] Datadog, "9 insights on real-world container use." `https://www.datadoghq.com/container-report/`, November 2022. [Online] Accessed: Mar. 13, 2023.

[10] Kubernetes. `https://kubernetes.io/`, January 2023. [Online] Accessed: Feb. 27, 2023.

[11] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes: Lessons learned from three container-management systems over a decade," *Queue*, vol. 14, no. 1, pp. 70–93, 2016.

[12] P. Raj, S. Vanga, and A. Chaudhary, *Kubernetes Architecture, Best Practices, and Patterns*, pp. 49–70. John Wiley & Sons, Inc, 2023.

[13] K. Feldsher, "Kubernetes security best practices." `https://coralogix.com/blog/kubernetes-security-best-practices/`, Oct 2022. [Online] Accessed: Feb. 27, 2023.

[14] S. Sarkar, G. Choudhary, S. K. Shandilya, A. Hussain, and H. Kim, "Security of zero trust networks in cloud computing: A comparative review," *Sustainability*, vol. 14, no. 18, 2022.

[15] Cybersecurity and Infrastructure Security Agency, "Kubernetes Hardening Guide," tech. rep., National Security Agency, August 2022.

[16] Clair, "Clair documentation." `https://quay.github.io/clair/`. [Online] Accessed: Apr. 3, 2023.

[17] Aqua Security, "Trivy." https://aquasecurity.github.io/trivy/v0.39/docs/. [Online] Accessed: Apr. 6, 2023.

[18] Aqua Security. https://aquasecurity.github.io/kube-hunter/, May 2022. [Online] Accessed: Apr. 1, 2023.

[19] M. S. I. Shamim, F. A. Bhuiyan, and A. Rahman, "Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices," in *2020 IEEE Secure Development (SecDev)*, pp. 58–64, IEEE, 2020.

[20] Istio. https://istio.io/, 2023. [Online] Accessed: Mar. 6, 2023.

[21] HashiCorp Consul. https://developer.hashicorp.com/consul/docs. [Online] Accessed: Apr. 4, 2023.

[22] Linkerd. https://linkerd.io/2.13/overview/. [Online] Accessed: Apr. 4, 2023.

[23] Aqua Security. https://github.com/aquasecurity/kube-bench, April 2023. [Online] Accessed: Apr. 11, 2023.

[24] Falco, "The falco project." https://falco.org/docs/, 2023. [Online] Accessed: Apr. 6, 2023.

[25] Calico. https://docs.tigera.io/calico/latest/about/. [Online] Accessed: Apr. 6, 2023.

Quantum Natural Language Processing

Ioana Moflic
ioana.moflic@aalto.fi

Tutor: Alexandru Paler

Abstract

The accurate understanding of natural language is a challenging task for classical computers. Quantum computing has the potential to significantly improve the accuracy of natural language processing. This paper reviews quantum computing and quantum natural language processing (QNLP). The potential of QNLP is highlighted by implementing, and successfully evaluating, a novel, state-of-the-art method for generating song lyrics.

KEYWORDS: quantum computing, natural language processing

1 Introduction

Quantum computing could facilitate progress in areas where research is arduous and would benefit from a high speed of execution, such as natural language processing.

Natural Language Processing (NLP) is a branch of computational linquistics which focuses on making computers understand human language and speech [13]. The most common tasks of NLP are dialogue management, question answering, speech recognition, etc.

The computational advantage of quantum computers allows conducting complex simulations. Quantum computing was born in the 1980's when

Paul Benioff described the first quantum computer model [1]. Quantum computers are based on the principles of quantum mechanics. Quantum mechanics is a branch of physics that describes the properties of nature at the atomic and subatomic level. The latter arose around the 1900s when the science of classical physics did not seem to provide sufficient explanations for all experimentally observed physical phenomena.

Quantum computers and classical computers share some similarities, but are different. Classical computers use bits to encode classical information, which take either the value 0 or value 1. Unlike classical computers, quantum computers use quantum bits, also known as *qubits*. Qubits are considered the fundamental unit of quantum information. Qubits mimic the unique behavior of subatomic particles that can simultaneously exist in several states, which means that a qubit can be 0, 1 or a superposition (linear combination of 0 and 1 [17]).

Current approaches to NLP use rule based modeling of natural language borrowed from linguistics together with various machine learning models to enhance the capability of machines to analyze and interpret the content and nuances of human language [13]. The most common tasks of NLP are dialogue management, question answering, speech recognition, etc. Moreover, ChatGPT [15], use textbooks, articles and Internet sources to generate dialogues with humans.

Recently, NLP tasks started to be run on quantum computers [6]. NLP tasks that can be run on quantum computers are commonly referred to as Quantum Natural Language Processing (QNLP) [6].

The first question answering task being was performed on a real quantum computer in 2020 [16]. With the help of diagrammatic structures borrowed from ZX-Calculus [7], the authors of [16] show how human language can be translated to quantum computers, and machine learning tasks, such as classification, are performed by encoding these structures into parametrised quantum circuits.

Quantum Machine Learning (QML) [10] refers to a family of variational quantum circuits that perform and improve machine learning tasks with the help of quantum computers. The parameters of the circuits are updated during the optimisation procedure and are learnt using various differentiation techniques, just as in the classical setting. NLP tasks can be translated to quantum computer software in the form of variational circuits. Such circuits contain parameterised quantum rotation gates, where the parameter of the rotation is its angle (Figure 1a).

This paper is an overview of the state of the art QNLP methods on real hardware. Section 2 presents the theoretical foundations needed for the implementation of QNLP, and Section 3 reviews and discusses novel and original results beyond the state of the art.

2 Background

This section reviews QML and QNLP and introduces the necessary background on quantum circuits, quantum gates and the ZX-Calculus, which is used as a diagrammatic representation of quantum processes.

2.1 Quantum circuits

The software used in quantum computing is presented in general in the form of quantum circuits (e.g Figures 1 b), 1 c)). The quantum circuit model represented by sequences of quantum gates and measurements arranged and read from left to right along the time axis. Quantum circuits are similar to a set of assembly language instructions sent to the Quantum Processing Unit (QPU). The instructions are known as quantum gates. Although there are infinitely many quantum gates, it is agreed that there exists a finite set of gates, namely the Clifford+T group, that acts as a generator used to represent any quantum computation. The Clifford group is formed of a few standard gates: X, Y, Z (the Pauli gates), S, Hadamard, and CNOT and the T gate [17].

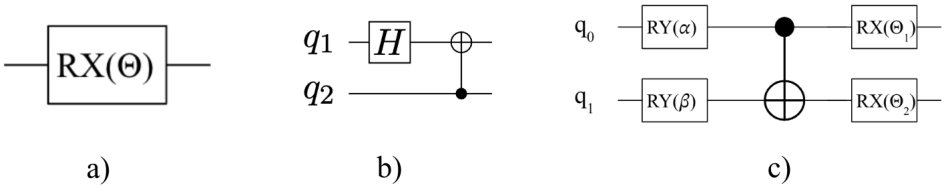


Figure 1. The horizontal line is a wire that holds a qubit. Information is processed from left to right. a) The box represents a rotation gate around the X axis with the angle θ. b) A quantum circuit consisting of two qubits and two gates: a one qubit Hadamard gate and a two qubit gate, known as CNOT gate. c) A simple Quantum Neural Network (QNN). The parameterised RY gates act as a data encoder, and the CNOT gate, together with the parameterised RX gates from the trainable circuit, also called ansatz.

2.2 ZX-Calculus

The ZX-Calculus is a graphical language that describes quantum interactions between qubits used for quantum circuit optimisation, measurement

based quantum computation, quantum error correction and more [19].

A ZX-diagram is a graphical representation of a quantum computation which is often described as a map between qubits. The translation between a quantum circuit and a ZX-diagram is straightforward, and can be realised by associating each quantum gate in the Clifford+T group [17] with its representatives in ZX-Calculus (Fig. 2).

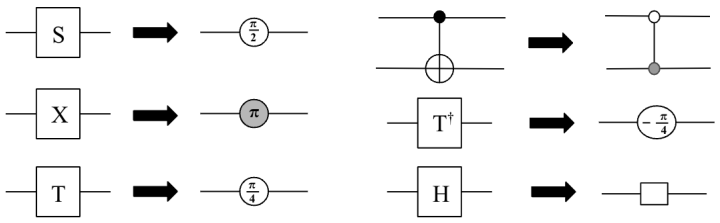


Figure 2. The translations of the Clifford gate group: gates S, X, Hadamard, T, T daggger, CNOT, into their respective ZX-diagrams.

The main benefit of translating quantum circuits to ZX-diagrams is that certain circuit optimisation patterns or gate identities are easier to perform on the ZX-diagram rather than directly on the quantum circuit. A ZX-diagram has the structure of an undirected graph, where nodes are colored either white or grey and are annotated with the phase angle of the gate. The colour of a dot is either white for the S gate, the T gate, and the control of a CNOT, or grey for the X gate or the target of a CNOT. One of the most important components of a ZX-diagram are *spiders*, which are of two types: Z-spiders (Figure 4) and X-spiders (Figure 3).

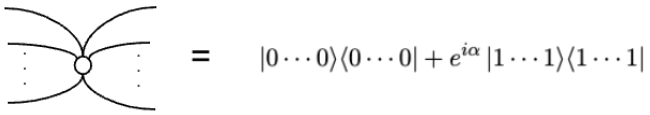


Figure 3. The X-spider is depicted on the left hand-side and the state vector it represents, on the right hand-side.

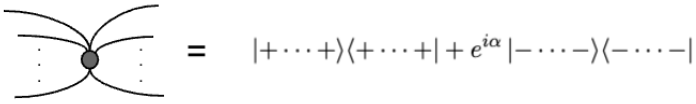
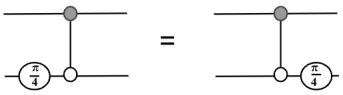


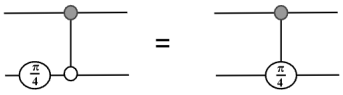
Figure 4. The Z-spider is depicted on the left hand-side and the state vector it represents, on the right hand-side.

Z-spiders are the white dots seen in the figures above, and X-spiders are the grey ones. The name "spider" is inspired by the arbitrarily many wires (inputs and outputs) that these dots have. The name Z-spider is derived from operating with respect to the eigenbasis of the Z matrix: $|0\rangle$ and $|1\rangle$. In turn, the X-spider operates with respect to the eigenbasis of

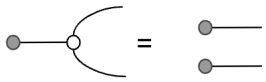
a) Dots of the same color commute through one another



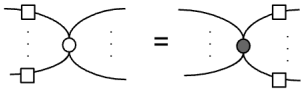
b) Dots of the same color fuse



c) A dot with a single wire copies through a dot of the opposite colour



d) A Hadamard gate is commuted through a dot by changing its colour



e) Dots with two wires can be removed

Figure 5. A few of the most common optimisation patterns used in ZX-Calculus.

the X matrix: $|+\rangle$ and $|-\rangle$. Here, $|0\rangle$, $|1\rangle$, $|-\rangle$ and $|+\rangle$ are defined as column vectors

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \; |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

The simplicity of ZX-Calculus comes from a set of predefined patterns used to transform the diagram, that are easier to use than the ones used in the quantum circuit setting (see Figure 5).

2.3 Quantum Machine Learning

Quantum Machine Learning (QML) is of a family of machine learning tasks that can be performed on a quantum computer. QML consists of hybrid methods (require both classical and quantum processing of data), combined with the computationally demanding subroutines being performed on the quantum device. Figure 6 is the QML training loop.

The domain is new and continuously evolving. Therefore a preliminary classification, according to [10], is:

1. *Quantum Boltzmann Machines (QBM)* are similar to classic Boltzmann machines. QBMs are based on the Ising Hamiltonian instead of the classical Boltzmann distribution.
2. *Variable depth quantum circuits (vVQC)* are an adaptation of the Variational Quantum Classifier (VQC) that claims to resolve its in-

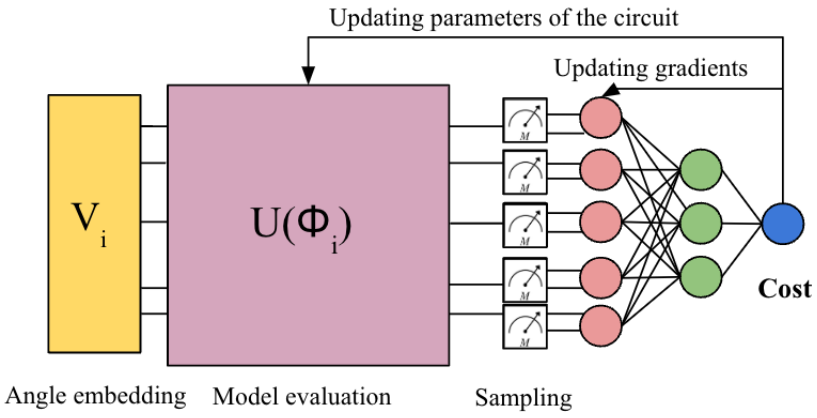


Figure 6. The general framework of a quantum machine learning model is composed of a variational quantum circuit that consists of two sections: 1) one that performs angle embedding of the data, denoted V, and 2) the quantum model, also known as ansatz, denoted U. After measurement, data is processed by the classical machine learning model, which outputs the result of a cost function at each training step. The hybrid model updates the parameters and the gradients of the variational circuit towards the minima of the cost function.

flexibility problem by varying the size of the circuit.

3. *Hybrid quantum autoencoders (HQA)* encode data in the classical latent space using amplitude quantum states. The encoder learns characteristics of the data by clustering similar data points.
4. *Quantum reservoir computing (QRC)* uses multiple small scale quantum systems that communicate, and are used for forecasting.
5. *Quantum multiclass classifier (QMCC)* is used to classify multiple class data by optimizing the adjustable parameters of variational quantum circuits.
6. *Support vector machines with a quantum kernel estimator (QSVM-Kernel)* classifies by encoding data in a quantum state space and searches for the hyperplane that linearly separates data.
7. *Quantum neural networks (QNN)* are similar to classical neural networks, but use a quantum perceptron model.
8. *Hybrid k-neighbours-nearby model (HKNN)* use the SWAP test circuit as a similarity measure between quantum states to perform the K-Nearest-Neighbours (KNN) algorithm.
9. *Orthogonal neural networks* enforce orthogonality on the parameters of the network.
10. *Quantum fully self-supervised neural networks (QFS-Net)* use qutrits (three-level quantum system where basis states are $|0\rangle$, $|1\rangle$ and $|2\rangle$) to encode quantum states in a way that enables a self-organized counter-propagation between the layers of the network.
11. *CNOT neural networks (CMN)* are only composed of CNOT gates

and measurement gates and are used to implement discrete Boolean functions that benefit from computational complexity improvements.

12. *Quantum convolutional deep convolutional neural networks (QDCNN)* employ quantum convolutional layers inspired from variational circuit where parameters are updated using hybrid optimization.
13. *Quantum backpropagation neural networks (QBP)* are based on the quantum neural model and classical backpropagation.
14. *Feed foward neural networks (ffNN)* are based on an efficient method of approximating inner products between vectors and benefit from computational complexity improvements because the computed values are stored on QRAM [11].
15. *Quantum generative adversarial networks (QGAN)* learn the probability distributions of the data points by encoding them into quantum states. QGANs use both variational quantum circuits and classical neural networks.
16. *Quantum recurrent encoding-decoding neural networks (QREDNN)* use a quantum neuron to jointly train the encoder and the decoder of the autoencoder.

2.4 Quantum Natural Language Processing

Quantum Natural Language Processing (QNLP) is a technique which proposes the modelling of natural language in a diagrammatic form, which is then processed by the quantum computer. There exist many state-of-the-art techniques in NLP [3, 9, 4], and it is widely accepted that the area would benefit of quantum speed-up [20].

QNLP [6] is the quantum adaptation of classical NLP and is different from NLP in the sense that meanings of words are encoded as state vectors into the quantum computer. The advantage of QNLP is the composition of word meanings that further gives meaning to more complex structures such as sentences and texts. The interaction of words that gives meaning to sentences is achieved by the DisCoCat (Categorical Compositional Distributional) [5] algorithm. Its generalization, the DisCoCirc (Circuit-shaped Compositional Distributional) algorithm [5], is able to give meaning to texts by composing meaning of sentences.

The DisCoCat model is based on grammatical algebra that is borrowed from linguistics. The author of [5] states that pregroup grammar, proposed by Joachim Lambek in the 1950s [14], is the foundation of the

DisCoCat algorithm for language modelling. The unexpected similarities that pregroup algebra used in linguistics and circuit diagrams have led to the idea of modelling language using Lambek's findings into circuit diagrams. The advantage is that the circuit diagrams are then translated to quantum computers to solve various NLP tasks.

Although believed to be impossible to achieve on near-term quantum devices, it was recently proved that NLP is "quantum native" [6, 18]. This indicates that human language is easier and more natural to translate to quantum computers than it is in the classical setting, which in turn requires heavy data preprocessing.

Diagrammatic reasoning (sections 2.1 and 2.2) has a very important advantage, that it combines the meaning of the words with the linguistic structure and grammar of the sentences into one [6]. Apart from DisCoCat, Categorical Quantum Mechanics (CQM) and ZX-Calculus are also theories used in modelling language as quantum processes [7]. Out of the three, ZX-Calculus has the role of translating and optimizing linguistic diagrams to quantum circuits, as seen in Figure 2 [6].

3 Discussion

This section introduces software libraries that enable preprocessing and training of QNLP tasks in the form of variational quantum circuits, as well as a novel, state of the art QNLP experiment.

3.1 Lambeq

Lambeq [12] is a Python library that contains modules that convert text into diagrams, tensor networks or variational quantum circuits that can be processed on a quantum computer. Lambeq facilitates the preprocessing step of QNLP tasks that rely on the DisCoCat framework. Lambeq is useful for syntacting parsing of natural language, rewriting and optimization of string diagrams, as well as for creating and manipulating the ansatz. Lambeq has the following pipeline [12]:

1. Depending on the compositional model selected, a specific parser can be used for constructing the syntax tree of the input sentence.
2. DisCoPy [8] is used for converting the syntax tree into a string diagram, according to the DisCoCat algorithm.
3. The string diagram is optimised by the application of rewrite rules

that aim to eliminate redundant connections.

4. The resulting string diagram is converted into a tensor network (for classical simulation) or a parametrised quantum circuit (for running on real hardware).

3.2 Pennylane

Pennylane is a Python library used for hybrid classical-quantum machine learning tasks. In the same way as classical machine learning relies on heavy GPU computations, quantum machine learning involves running the same computations on quantum hardware. Pennylane enables this by relying on quantum differentiable programming similar to other deep learning frameworks, such as NumPy, PyTorch, Tensorflow or JAX.

Pennylane integrates classical deep learning Python libraries together with libraries that enable running quantum circuits on real hardware. Pennylane allows the creation of an end-to-end quantum machine learning application. The framework also has support for simulating the circuits classically, by using the **lightning** module.

Quantum circuits, also called QNodes in Pennylane, are used to define the variational quantum circuit and the device interface that will execute it: classical or quantum. The library integrates quantum computations by adapting the circuit to various library-dependant data-structures or optimizers. Optimizers update both the parameters of the variational circuit and the gradients towards the minima of the cost function used to assess the performance of the model (as seen in Figure 6). Pennylane does this by making the gradients of the variational circuit visible to the classical library, which will employ its own optimizers for optimizing the circuit parameters.

3.3 Experiment

This review paper includes a proof-of-concept implementation of a QNLP model, namely a Quantum Long Short Term Memory (LSTM) model, that generates songs based on an available dataset of lyrics.

LSTM is a type of Recurrent Neural Network (RNN) used for generating sequences of words in which the relations between the words are kept long term during training. This allows for using LSTMs for complex NLP tasks such as sentiment analysis, question answering or text generation, where the first one is a many-to-one task and the last two are many-to-

many tasks. Many-to-many tasks are different from many-to-one tasks in the sense that they require a human-like responses such as sentences or lengthy texts, rather than just a one-word classification. Two types of RNNs are used to account for the relations between words that generate meaningful sentences: LSTMs and Gated Recurrent Units.

Our implementation uses a quantum variant of the LSTM layer, also known as QLSTM, in which the input, output, forget and update components are replaced with trainable quantum circuits. This is achieved by employing Pennylane for the translation between the variational quantum circuit and a PyTorch layer [2], which enables the integration of the QLSTM layer seamlessly into the classical NLP model. These LSTM layers are converted to quantum circuits by first embedding the data tensors into the angles of the single qubit rotation gates, and then applying an embedding quantum layer that entangles the qubit wires with the scope of creating connections between the state vectors that represent the words.

The implementation does not currently use the DisCoCat model for the preprocessing step, but its classical variant, the bag-of-words model. The model is predicting verses of songs based on a "prompt" word. Preliminary results (see Figure 7) show that this is functioning as expected.

```
You between my nothing couldn't again coat in you
You the the I it to I the many someone you
You years fall the with someone I my you I many
You my me nd rags be sewed made kind in will


You summer Bible she thing lonely it of many found
You said can no in in with me you 'em my
You travelled she see of she crashing many told
You Only never (Mmm) to (Mmm) it couldn't sewed
```

```
So again to got many the it the my than many
So me a alone I the my the will me have
So youth explain alone I in many she you you in
So have you in a many my love while you it

So in be that I you
So for you again you
So while alone had can't of
So you it many many talk
```

Figure 7. Example lyrics generated by our QNLP model.

4 Conclusion

Classical NLP is challenging and computationally demanding. Quantum computers could offer a computational advantage by speeding up and improving NLP accuracy. QNLP is the quantum version of classical NLP, and its main advantage is the use of a more natural way of representing human language for computers.

This paper reviews the concepts of quantum computing, quantum machine learning, ZX-Calculus and of QNLP. The potential of QNLP is illustrated by implementing a novel generative model of song lyrics. Future work will focus on improving the accuracy of the generated lyrics.

References

[1] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22:563–591, 05 1980.

[2] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. Akash-Narayanan, Ali Asadi, Juan Miguel Arrazola, Utkarsh Azad, Sam Banning, Carsten Blank, Thomas R Bromley, Benjamin A. Cordier, Jack Ceroni, Alain Delgado, Olivia Di Matteo, Amintor Dusko, Tanya Garg, Diego Guala, Anthony Hayes, Ryan Hill, Aroosa Ijaz, Theodor Isacsson, David Ittah, Soran Jahangiri, Prateek Jain, Edward Jiang, Ankit Khandelwal, Korbinian Kottmann, Robert A. Lang, Christina Lee, Thomas Loke, Angus Lowe, Keri McKiernan, Johannes Jakob Meyer, J. A. Montañez-Barrera, Romain Moyard, Zeyue Niu, Lee James O'Riordan, Steven Oud, Ashish Panigrahi, Chae-Yeun Park, Daniel Polatajko, Nicolás Quesada, Chase Roberts, Nahum Sá, Isidor Schoch, Borun Shi, Shuli Shu, Sukin Sim, Arshpreet Singh, Ingrid Strandberg, Jay Soni, Antal Száva, Slimane Thabet, Rodrigo A. Vargas-Hernández, Trevor Vincent, Nicola Vitucci, Maurice Weber, David Wierichs, Roeland Wiersema, Moritz Willmann, Vincent Wong, Shaoming Zhang, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2022.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[5] Bob Coecke. The mathematics of text structure. *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, pages 181–217, 2021.

[6] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis, and Alexis Toumi. Foundations for near-term quantum natural language processing. *arXiv preprint arXiv:2012.03755*, 2020.

[7] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[8] Giovanni de Felice, Alexis Toumi, and Bob Coecke. Discopy: Monoidal categories in python. *arXiv preprint arXiv:2005.02975*, 2020.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] David Peral García, Juan Cruz-Benito, and Francisco José García-Peñalvo. Systematic literature review: Quantum machine learning and its applications. *arXiv preprint arXiv:2201.04093*, 2022.

[11] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16), apr 2008.

[12] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. lambeq: An efficient high-level python library for quantum nlp. *arXiv preprint arXiv:2110.04236*, 2021.

[13] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744, 2023.

[14] J. Lambek. Pregroup grammars and chomsky?s earliest examples. *Journal of Logic, Language and Information*, 17(2):141–160, 2008.

[15] Christoph Leiter, Ran Zhang, Yanran Chen, Jonas Belouadi, Daniil Larionov, Vivian Fresen, and Steffen Eger. Chatgpt: A meta-analysis after 2.5 months, 2023.

[16] Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, and Bob Coecke. Grammar-aware question-answering on quantum computers. *arXiv preprint arXiv:2012.03756*, 2020.

[17] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.

[18] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, aug 2018.

[19] John van de Wetering. Zx-calculus for the working quantum computer scientist. *arXiv preprint arXiv:2012.13966*, 2020.

[20] William Zeng and Bob Coecke. Quantum algorithms for compositional natural language processing. *Electronic Proceedings in Theoretical Computer Science*, 221:67–75, aug 2016.

Programming Orchestration of Data Analysis Workflows in Edge Cloud Continuum

Ishani Bhardwaj
ishani.bhardwaj@aalto.fi

Tutor: Linh Truong

Abstract

The proliferation of computing-capable devices in recent years has facilitated the emergence of a new computing paradigm at the Edge. Despite being resource-limited compared to cloud servers, Edge devices offer lower latency, making the combination of cloud and Edge computing paradigms an attractive infrastructure for data analysis workflows that have diverse requirements, which cannot be easily achieved using either paradigm in isolation. Therefore, effective orchestration of workflows in Edge-cloud computing environment presents significant challenges that must be addressed in order to realize the full potential of edge-cloud continuum. This paper examines the challenges associated with programming and building workflows, and discusses various workflow management systems that can be leveraged to support automation of data analysis tasks.

KEYWORDS: edge-cloud continuum, data analysis workflow, orchestration

1 Introduction

Over the years, the rapid expansion of the Internet of Things (IoT) and cloud resources have resulted in a significant increase in the number of

internet-connected devices situated on the network edge. The data generation rate has also increased; hence, processing and computation of insights from the data became complex. Orchestration of data analysis workflow can serve as a beneficial abstraction that performs computational tasks and controls interdependencies among the tasks.

In edge computing, the computation takes place near the data source, which includes devices embedded with sensors, thus enabling data analysis, information monitoring and sharing [9]. Edge computing comprises numerous benefits, such as high responsiveness, scalability, potential for enforcing data privacy restrictions, and the capability of masking cloud outages [17]. Continuum computing refers to a computing paradigm where distributed resources can be seamlessly combined and orchestrated to support dynamic and complex application workflows [4]. The goal of continuum computing is to provide a flexible and scalable infrastructure for data-driven applications, where resources can be allocated and managed dynamically to meet the changing needs of the workflow. By leveraging analytics in the edge cloud continuum, organizations can take advantage of the benefits of both the network edge and cloud analytics, including real-time processing, scalability, cost saving, reliability, data locality, and low latency. This can lead to improved data-driven decision-making and more effective use of data-driven applications.

This paper discusses programming orchestration of data analysis workflows in the edge cloud continuum. It examines the challenges associated with building and management of data analysis workflows in the edge-cloud continuum and explore the tools and techniques available to address these issues.

The paper is structured as follows: Section 2 briefly introduces the architecture and expansion of the edge cloud continuum; Section 3 covers various challenges in orchestrating data analysis workflows; Section 4 evaluates different tools and techniques to orchestrate data analysis workflows; Section 5 discusses the challenges and tools presented in this paper; finally, Section 6 provides concluding remarks.

2 Edge computing: literature work

2.1 Origin and expansion of edge computing

Edge computing has its roots in the late 1990s with the introduction of content delivery networks (CDNs), which aim to improve web and video content. Akamai was one of the pioneers in CDNs that brought the idea of edge computing during early 1999 for improving the performance of websites, and prevent them from crashing or slow response time [6]. CDNs use edge nodes deployed close to the end-users to cache and serve content, which can significantly improve web performance by reducing latency and bandwidth requirements [17]. It also offers some level of customization, such as adding location-specific content, which can enhance the user experience. During the early 2000s, CDNs developed further to enable the hosting of distributed applications and their components on edge servers, providing high fault tolerance and scalability [14].

In recent years, edge computing has expanded beyond CDNs to include a wide range of applications and use cases, from IoT devices to autonomous vehicles and more. This marked the beginning of the shift towards more decentralized and distributed computing architectures, where computing resources are located closer to end-users to provide better performance, lower latency, and improved user experience [5]. Since then, edge computing has continued to evolve, driven by cutting edge technologies, such as IoT, artificial intelligence (AI), and 5G, among others. According to Statista, the number of IoT connected devices worldwide was 13.1 billion in 2022, and it is expected to exceed 29 billion by 2030 [19]. With such a vast number of devices generating data, transmitting data to the cloud for processing would be slow and could cause network congestion [25]. By leveraging edge computing, the amount of data that needs to be sent to the cloud would reduce, thereby enhancing performance and data security.

2.2 Architecture of edge cloud continnum

Edge-cloud continuum computing is a hybrid approach that combines the strengths of both edge computing and cloud computing to process and analyze data efficiently. The architecture typically consists three main layers: the device layer, the edge layer and the cloud layer as shown in Figure 1 [11]. The cloud layer comprises of large-scale data centers or

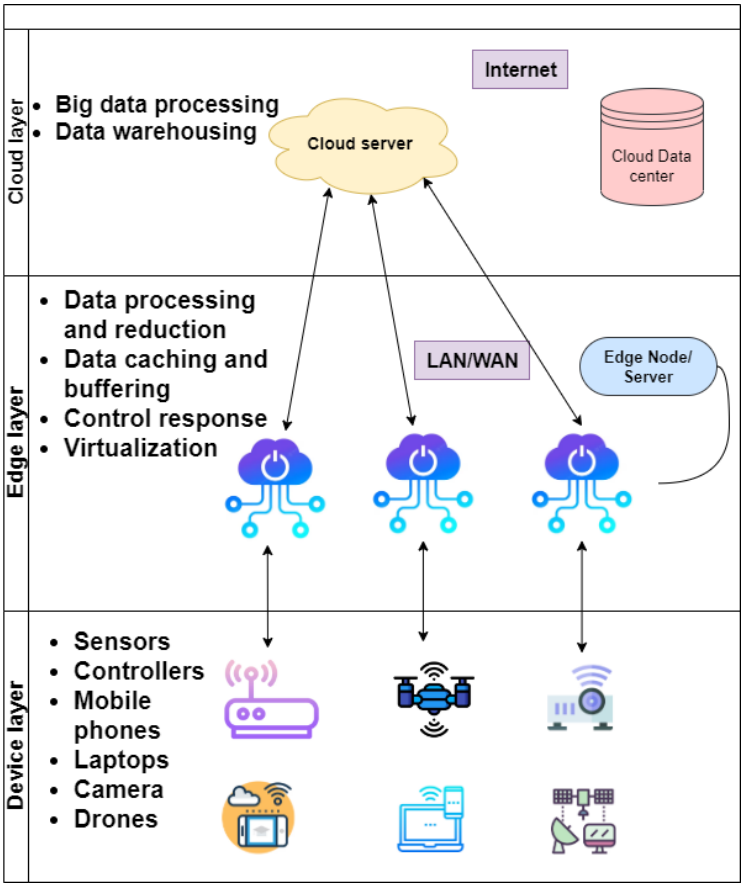


Figure 1. Edge cloud continuum architecture Adapted: [1]

remote servers that provide high-performance computational and storage resources for data processing and warehousing. The edge layer consists of edge nodes, which are small-scale data centers or cloudlets that are located closer to end-users and can provide low-latency and high-bandwidth computing and storage resources [17]. The device layer involves electronic devices, such as smartphones, tablets, IoT sensors, and other edge devices that are capable of generating, processing, and transmitting data. The working of the edge-cloud continuum can be summarized as follows:

- Data is collected from edge devices such as sensors, cameras, and mobile devices present in device layer. This data may be raw data or pre-processed data.

- The edge layer processes and analyzes the data that is collected from the device layer. This layer is composed of edge servers, edge nodes and other computing resources located closer to the devices. It can perform initial skimming and pre-processing on site to reduce the volume of data that requires transmission to cloud queue. This decentralization allows

a reduction in data latency.

- After the completion of data processing and analysis at the edge layer, data beneficial for real-time analysis is processed and strategic information is sent to the reference cloud provider for detailed data-analysis, storage, and visualization. The cloud layer provides advanced machine learning algorithms and other analytical tools to process the data and provide valuable insights. It can also store the data for long-term analysis.

Overall, the architecture of the edge cloud continuum is designed to provide a flexible, high-performance and scalable infrastructure that has a broad range of practical applications and services.

2.3 Data analysis workflows

Data analysis workflow refers to a systematic process of transforming raw data into meaningful insights. It typically involves a sequence of computational tasks, connected through control-flow (operations or tasks specified by edges) or data-flow (data movement), that are undertaken to extract, process, clean, transform, and analyze data to obtain actionable insights [10]. There are four primary categories of data analysis, which include descriptive, diagnostic, predictive, and prescriptive analysis [8].

- Descriptive Analysis: It involves summarizing and presenting key performance indicatiors (KPIs) of a dataset. The goal of descriptive analysis is to provide insights into what has happened in the past and what is currently happening.

- Diagnostic Analysis: It is used to detect the root cause of a problem or to understand why a particular outcome occurred. It involves analyzing data to determine the factors that contributed to a particular event or outcome.

- Predictive Analysis: It is a technique that utilizes statistical algorithms and machine learning methods to examine data and anticipate future results. The objective is to make predictions about future occurrences based on past data patterns and trends.

- Prescriptive Analysis: It involves using optimization and simulation techniques to identify the best course of action to achieve a desired outcome. It takes into account various constraints, such as limited resources, and suggests the best possible solution to optimize the outcome.

Utilizing different data analysis methods, workflows can be divided into two main categories: abstract and executable workflows [18].

An abstract workflow, which is resource independent, provides a high-level description of the data analysis process, outlining the steps involved in cleaning, preprocessing, transforming, and analyzing the data. It is typically represented in YAML format and can be generated using the Workflow API. An executable workflow, on the other hand, is a detailed, resource-dependent and automated workflow that performs the steps outlined in the abstract workflow by mapping the tasks to the target resources. By using an executable workflow in the edge-cloud continuum, data analysis tasks can be executed on both edge devices and cloud resources, depending on the available resources and the specific requirements of the task. This approach enables more efficient and scalable data analysis, with the ability to perform tasks closer to the data source when needed, leading to faster and more reliable results. Liew et al. [10] analyzed several Workflow Management Systems (WMSs), e.g., Airavata [13], Kepler [12], KNIME [7], Pegasus [18], Taverna [24], and Swift [23], that are commonly used to orchestrate the distribution of tasks to resources based on availability and dependency. The paper focus on Pegasus, KNIME Edge and KubeEdge WMS which can be utilized to manage and execute complex workflows that involve both edge and cloud resources, allowing for more efficient and effective data analysis.

3 Orchestration of data analysis workflows: key challenges

There has been a surge of interest in processing big data, as it offers the potential to uncover valuable insights that can transform corporate businesses, government schemes, and research outcomes. This has resulted in the development of new programming, communication, and processing technologies, such as Hadoop, Storm and Spark, as well as cloud computing services. Analytical applications have evolved to comprise multiple analytical steps that run as workflows, which are markedly different from traditional workflows, presenting researchers and data scientists

with the challenge of managing and orchestrating their execution effectively. These challenges include edge-cloud integration, data security and consistency, workflow scheduling and management, and the adoption of heterogeneous data analysis tasks.

3.1 Edge-cloud integration

The integration of edge and cloud computing results in a complex execution environment that requires the consideration of various factors to effectively utilize both computing paradigms. The challenges include visibility of available resources, efficient scheduling decisions, resource provisioning, software systems to handle failures, and capturing performance metrics for optimization [18]. The limited resources and dynamic nature of edge devices, combined with the high computing and data movement costs of the cloud, make it challenging to balance the trade-off between these two computing paradigms. However, with proper planning and management, the benefits of edge-to-cloud computing can be realized, including reduced latency, improved reliability, and increased security.

3.2 Data security and consistency

Data analysis workflows enable the combination of diverse processing tasks in data pipelines, which can involve distinct types of data-flow, including stream or batch processing. These different data flows are related with various data programming models that constitute a part of the workflows. For instance, stream processing frameworks, such as Apache Storm and message queuing systems, for example, Apache Kafka receive data streams via Kafka producer or spout, respectively. Conversely, batch processing frameworks, such as Apache Hadoop require large datasets to be stored in cloud storage and then sent to the Hadoop cluster using HDFS [2]. Since data workflows are typically executed in the cloud, the diverse storage requirements of the different tasks in the workflow present various challenges for managing cloud storage resources to meet their needs. These challenges may include issues, such as data security, data consistency, data durability, and data transfer costs. As a result, researchers and data scientists must carefully consider the storage requirements of different workflow tasks and adopt appropriate strategies and tools for managing cloud storage resources effectively.

3.3 Workflow scheduling and management

When data workflow scheduling involves the use of edge resources, new challenges arise in efficiently mapping and deploying data analysis tasks on edge devices that may have limited resources. Ensuring isolation, orchestration, and scheduling in containers becomes critical [16], necessitating the use of lightweight hardware and containerization software [22]. To optimize resource selection and allocation, researchers and data scientists must consider the hardware limitations and heterogeneity of the tasks involved in the workflow.

3.4 Adoption of heterogeneous data analysis tasks

One of the major challenges researchers face is to adapt heterogeneous data processing and analysis tasks from virtual machine to container based workload for Kubernetes, which requires creating efficient container images for the workloads. Additionally, management of containers on the edge resources is complex because edge resources have a dynamic nature and their performance fluctuates. Defining runtime configurations and deploying them into the container environment is crucial. Container management is further complicated by the need to maintain Service-Level Agreement (SLA) and Quality of service (QoS) requirements on constrained resources and respond to unexpected changes at runtime [15]. Overall, the process of orchestration becomes increasingly complex when scheduling in edge environment is virtualized that uses lightweight containers. Researchers and data scientists must develop effective strategies to address these challenges and ensure the efficient deployment and management of big data workloads on edge resources, while maintaining SLA and QoS requirements.

4 Tools and Techniques for orchestrating data analysis workflows

To address the challenges of programming orchestration of data analysis workflows in the edge cloud continuum, several tools and techniques have emerged [10]. This section focuses on Pegasus [18], KNIME [7], and KubeEdge [21] that enable the orchestration of data analysis workflows, but they differ in their approaches and the steps involved. Here is a comparison of the various steps involved in each platform:

Define the Workflow: This step is common across all three platforms. Users need to identify the data sources, the data processing steps, and the output targets to define the workflow.

Create the Workflow: Pegasus and KNIME Edge provide graphical user interfaces (GUI) for creating workflows, while KubeEdge relies on Kubernetes deployment files. KNIME Edge also provides the ability to import workflows created in KNIME Analytics Platform.

Configure Nodes: This step involves configuring the individual nodes that make up the workflow. In KNIME Edge and KubeEdge, this involves selecting the data processing algorithms and configuring their parameters, while in Pegasus, this involves defining the workflows using a directed acyclic graph (DAG).

Manage Resources: This step involves optimizing the use of resources available on edge devices, such as CPU, memory, and power. Pegasus support tools and KubeEdge provide Edge controller, metadatasyncservice, and Edgecore for managing resources, while KNIME Edge provides various extensions to optimize the workflow for the resources available on the edge device.

Package the Workflow: This step involves packaging the workflow so that it can be deployed on the edge device. In Pegasus, this process requires the creation of scripts that are dependent on resources, and which are used for job submission and data movement. In KNIME Edge, this involves selecting the nodes and data sources required for the workflow and packaging them into a container. In KubeEdge, this involves creating Kubernetes deployment files for the workflow.

Deploy the Workflow: In Pegasus, this involves submitting the generated scripts to the HTCondor DAGMan. In KNIME Edge, this involves deploying the workflow on KNIME Server. In KubeEdge, this involves deploying the Kubernetes deployment files to the edge device.

Manage Data Flows: This step involves managing the flow of data between the various components of the workflow. In Pegasus and KubeEdge, this involves managing data transfer between jobs using suitable data transfer protocols. In KNIME Edge, this involves specifying the input data sources, applying data transformation algorithms to preprocess the data, and specifying the output data destinations.

The data analysis methods in the aforementioned WMS depends upon the type of data and requirements of the researchers and data scientists. Furthermore, programming workflows efficiently can significantly reduce

the challenges mentioned in section 3. All the above mention WMS support edge-cloud integration and provide scalability and fault tolerance. A recent development in Pegasus-Kickstart has been the introduction of GPU monitoring extensions, which provide improved performance of workflows [3].

5 Discussion

In contrast to cloud computing, edge-cloud computing extends the capabilities of cloud computing by facilitating data processing and analysis at the endpoints and edges. This proximity eliminates the need for data to traverse over a network to a remote data center or cloud, thereby significantly reducing latency. However, there are significant challenges in programming orchestration of data analysis workflows in the edge cloud continuum, such as edge-cloud integration, data security, workflow sechedluing, and heterogeneous data analysis tasks. Additionally, network infrastructure at the edge is often characterized by limited bandwidth and unstable connectivity. Based on the analysis of WMSs, this paper proposes that to mitigate the impact of these limitations, it is essential for edge nodes to be autonomous, enabling them to operate independently of the network and prevent interruptions due to the lack of connectivity. The data transferred from edge devices for analysis should be secured by encryption/decryption strategies to avoid any attack on data in network layer. Data storage in cloud is secured, hence, complicated data analysis can occur in edge-cloud continuum environment.

The choice of WMS depends upon the use case, all the tools explained in section 4 can be used to orchestrate workflows in both Edge and Cloud. Pegasus provides built-in support for resource management and job execution through HTCondor, KNIME Edge can be resource-intensive but uses it's own built-in workflow engines to support orchestration, parallel processing and streaming. On the contrary, KubeEdge requires significant setup, configuration and orchestration is done through kubernetes deployment files. All the WMSs provides fault tolerance and supports containerization.

According to Gartner, it is estimated that by 2025, a significant majority of data, i.e., approximately 75%, will be processed outside of the traditional data center or cloud [20]. This suggests that edge computing, with its ability to process data closer to the source, is likely to gain increased

adoption in the coming years.

6 Conclusion

In conclusion, programming orchestration of data analysis workflows in the edge cloud continuum requires careful consideration of the resources available on each device and the heterogeneity of devices in the system. However, with the right tools and techniques, it is possible to build and manage data analysis workflows that can run seamlessly across different devices. As edge computing continues to grow, it is crucial to develop new tools and techniques that can be used to manage the complexity of the edge cloud continuum.

References

[1] Hossein Ashtari. Edge computing vs. fog computing: 10 key comparisons, Feb 2022.

[2] Mutaz Barika, Saurabh Garg, Albert Y. Zomaya, Lizhe Wang, Aad Van Moorsel, and Rajiv Ranjan. Orchestrating big data analysis workflows in the cloud: Research challenges, survey, and future directions. *ACM Comput. Surv.*, 52(5), sep 2019.

[3] Henri Casanova, Ewa Deelman, Sandra Gesing, Michael Hildreth, Stephen Hudson, William Koch, Jeffrey Larson, Mary Ann McDowell, Natalie Meyers, John-Luke Navarro, George Papadimitriou, Ryan Tanaka, Ian Taylor, Douglas Thain, Stefan M. Wild, Rosa Filgueira, and Rafael Ferreira da Silva. Emerging frameworks for advancing scientific workflows research, development, and education. In *2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pages 74–80, 2021.

[4] Ali Reza Zamani Anthony Simonet Daniel Balouek-Thomert, Eduard Gibert Renart and Manish Parashar. Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows. *The International Journal of High Performance Computing Applications*, 33(6), 2019.

[5] A. Davis, J. Parikh, and W. E. Weihl. Edgecomputing: Extending enterprise applications to the edge of the internet. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers amp; Posters*, WWW Alt. '04, page 180–187, New York, NY, USA, 2004. Association for Computing Machinery.

[6] John Dilley, Bruce M. Maggs, Jay Parikh, Harald Prokop, Ramesh K. Sitaraman, and William E. Weihl. Globally distributed content delivery. *IEEE Internet Comput.*, 6:50–58, 2002.

[7] Alexander Fillbrunn, Christian Dietz, Julianus Pfeuffer, René Rahn, Gregory A. Landrum, and Michael R. Berthold. Knime for reproducible cross-

domain analysis of life science data. *Journal of Biotechnology*, 261:149–156, 2017.

[8] Bunmi Funmilola and Amos David. Evaluation of diagnostic analysis and predictive analysis for decision making. 08 2019.

[9] In Lee and Kyoochun Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, 2015.

[10] Chee Sun Liew, Malcolm P. Atkinson, Michelle Galea, Tan Fong Ang, Paul Martin, and Jano I. Van Hemert. Scientific workflows: Moving across paradigms. *ACM Comput. Surv.*, 49(4), dec 2016.

[11] Dongqi Liu, Haolan Liang, Xiangjun Zeng, Qiong Zhang, Zidong Zhang, and Minhong Li. Edge computing application, architecture, and challenges in ubiquitous power internet of things. *Frontiers in Energy Research*, 10, 2022.

[12] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency Computation*, 18(10):1039–1065, August 2006.

[13] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon E. Pierce, Chris Mattmann, Raminderjeet Singh, Thilina Gunarathne, Eran Chinthaka Withana, Ross Gardler, Aleksander Slominski, Ate Douma, Srinath Perera, and Sanjiva Weerawarana. Apache airavata: a framework for distributed applications and computational workflows. In *Grid Computing Environments*, 2011.

[14] Erik Nygren, Ramesh Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *Operating Systems Review*, 44:2–19, 01 2010.

[15] R. Ranjan, Saurabh Garg, Ali Khoskbar, Ellis Solaiman, Philip James, and Dimitrios Georgakopoulos. Orchestrating bigdata analysis workflows. *IEEE Cloud Computing*, 4:20–28, 01 2017.

[16] T Ramalingeswara Rao, Pabitra Mitra, Ravindara Bhatt, and Adrijit Goswami. The big data system, components, tools, and technologies: a survey. *Knowledge and Information Systems*, pages 1–81, 09 2019.

[17] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[18] Ryan Tanaka, George Papadimitriou, Sai Charan Viswanath, Cong Wang, Eric Lyons, Komal Thareja, Chengyi Qu, Alicia Esquivel, Ewa Deelman, Anirban Mandal, Prasad Calyam, and Michael Zink. Automating edge-to-cloud workflows for science: Traversing the edge-to-cloud continuum with pegasus. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 826–833, 2022.

[19] Lionel Sujay Vailshery. Iot connected devices worldwide 2019-2030, Nov 2022.

[20] Rob van der Meulen. What edge computing means for infrastructure and operations leaders. *Gartner Research*, 2018.

[21] Rafael Vaño, Ignacio Lacalle, Piotr Sowiński, Raúl S-Julián, and Carlos E. Palau. Cloud-native workload orchestration at the edge: A deployment review and future directions. *Sensors*, 23(4), 2023.

[22] David von Leon, Lorenzo Miori, Julian Sanin, Nabil El Ioini, Sven Helmer, and Claus Pahl. A lightweight container middleware for edge cloud architectures. In *Fog and Edge Computing*, 2019.

[23] Michael Wilde, Mihael Hategan, Justin M. Wozniak, Ben Clifford, Daniel S. Katz, and Ian Foster. Swift: A language for distributed parallel scripting. *Parallel Computing*, 37(9):633–652, 2011.

[24] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1):W557–W561, 05 2013.

[25] Shihong Zou, Xitao Wen, Kai Chen, Shan Huang, Yan Chen, Yongqiang Liu, Yong Xia, and Chengchen Hu. Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter. *Computer networks*, 67:141–153, 2014.

A comparison of classification approaches in likelihood-free model selection

Jana Fischer
jana.fischer@aalto.fi

Tutor: Ayush Bharti

Abstract

Model selection is the task of choosing the best-fitting model for a given data set. Due to a complex model or complex data, the likelihood function of the model can become intractable, and therefore methods like Approximate Bayesian Computation (ABC) are needed for model selection. ABC-RF is an approach that uses the random forest (RF) classifier in combination with ABC for the model selection. This paper examines if other classifiers are suitable to replace the RF classifier in ABC-RF.

KEYWORDS: ABC, likelihood-free, model selection, random forest

1 Introduction

In many fields of science and engineering, models are used to describe or examine a phenomenon. These models can be built with already existing expert knowledge. For training, the models require access to the data-based likelihood function. However, if the models get too complex, the likelihood function gets intractable and cannot be used. Intractable likelihood functions occur for complex models with many parameters, as well as for complex training data [2]. In addition, the model may contain many hidden states which make the computation of the likelihood

function difficult [1].

The problem of intractable likelihood functions is solved by Approximate Bayesian Computation (ABC). In ABC, data is simulated from a sampled parameter vector θ and the parameter vector is accepted or rejected depending on the similarity of simulated and observed data [1].

In addition to the parameters θ, the model structure itself has an impact on the quality of the result. Model structure refers to both the type of model and its hyperparameters. In classic machine learning tasks, the model and hyperparameter selection is often performed with methods, such as k-fold cross-validation or information criteria, such as the Akaike information criterion (AIC) or Bayesian information criterion (BIC) [2]. However, all these approaches use the likelihood function and are inappropriate for models with intractable likelihoods. The previously introduced likelihood-free ABC method can also be used for model selection.

The basic ABC approach faces several challenges, such as the choice of hyperparameters and summary statistics. Pudlo et al. [8] use an RF classifier to avoid the challenge of summary statistic selection. Since its introduction, ABC-RF was applied in various fields.

A random forest (RF) is not the only classifier suitable for a large amount of data. However, comparison studies for different classifiers have not been made. Therefore, this paper compares the ABC algorithm for model selection with different classifiers.

This paper overviews ABC and ABC-RF and examines an approach where the RF is substituted with other classifiers. It focuses on the performance and computation time of different classifiers. Section 2 gives an overview of ABC and likelihood-free model selection. Section 3 describes the ABC-RF method and classifiers that can replace the RF in the selection process. An experiment and its results are described in Section 4 and discussed in Section 5.

2 ABC methods

This section gives a motivation for the use of ABC methods and describes its usage in parameter estimation and model selection. In addition, the section describes the drawbacks of ABC and overviews approaches to solving these problems.

2.1 Background on Bayesian inference

Parameter estimation approaches use observed data y and a parameter vector θ that describes the model in detail. It is assumed that y is sampled from an unknown distribution and therefore is stochastic. In addition, the model parameters θ are considered to be stochastic, and thus follow a distribution $p(\theta)$. The aim is to fit the model to the observed data y, finding a parameter vector θ such that the model output with θ and the data y are similar. The probability of a parameter vector θ is computed depending on the data points y with Bayes' theorem:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{1}$$

This method is known as Bayesian inference [1]. In Equation 1, $p(\theta|y)$ is the posterior distribution, $p(y|\theta)$ is the likelihood of the model depending on θ, and $p(\theta)$ is the prior distribution. Intuitively, $p(\theta)$ is the prior knowledge about the model, and with the data and likelihood $p(y|\theta)$ the prior knowledge is updated with each data point y. The denominator $p(y)$ of Equation 1 is named *evidence* and scales the product of likelihood and posterior distribution to create a probability distribution.

The Bayesian inference method can only be applied if the likelihood function $p(y|\theta)$ can be evaluated. However, $p(y|\theta)$ often is complex or intractable and the posterior $p(\theta|y)$ cannot be computed in closed form [11] nor can samples be obtained from it. Therefore, an approach to compute the posterior $p(\theta|y)$ without using the likelihood is needed.

2.2 ABC

The ABC method was first introduced by Rubin [10] in 1984. The key idea of the method is that the samples x are simulated from a distribution $p(\theta, y)$, and if the samples are similar to the observed data, the distribution $p(\theta|y)$ is adjusted to the samples. The accepted parameters θ follow the approximate posterior probability distribution $p(\theta|y)$ [1]. The terms *ABC* and *likelihood-free inference* are often used interchangeably [6]. The basic ABC algorithm is denoted *Rejection sampling ABC* [11]:

Algorithm 1 Rejection sampling ABC

Require: Given are observations y.

while fewer than N samples are accepted **do**

Draw parameters $\theta_i \sim p(\theta)$

Simulate $x_i \sim p(\cdot|\theta_i)$

If $\rho(S(x_i), S(y)) > \epsilon$, reject x_i

end while

In Algorithm 1, $S(\cdot)$ is a summary statistic, ρ measures the distance and ϵ is the threshold for accepting or rejecting a sample x_i. A summary statistic $S(\cdot)$ is used because x and y often are high-dimensional data sets. In high dimensions, the curse of dimensionality leads to distances in high dimensions being exponentially more difficult to estimate. Summary statistics usually reduce the dimensionality of the data vectors x and y [1]. Due to the dimensionality reduction, information gets lost when using summary statistics. To counteract this effect, a combination of several statistics can be used. The combination can preserve the information from high dimensional data x and y.

A combination of summary statistics, however, causes another problem. Assume that several statistics are used and the rejection rate of the algorithm is 90%, then it is up to 99% for the combination of two statistics and up to 99.9% for three statistics, where the combined rejection rate depends on the correlation of model parameters [1]. As a consequence, sparse and correct statistics should be chosen. Selecting the summary statistics is a difficult task in ABC and an insufficient choice of statistics can result in a poorly performing model [9].

The first rejection sampling ABC algorithm from 1984 used the equality of sample x_i and observed data y as a criterion for accepting x_i [10]. However, this results in a high rejection rate and a large amount of generated samples. Therefore, the distance measure ρ and tolerance region ϵ are applied [8]. A larger threshold ϵ leads to a smaller sample rejection rate, but also to a larger approximation error. One approach to finding a good balance is to use a list of decreasing threshold values instead of a fixed ϵ [6]. In addition, if a combination of summary statistics is used, ϵ might need to be different for each summary statistic due to different scaling [1, 6]. An approach to solve these challenges regarding the selection of a good ϵ value is to use the k nearest neighbors of the observed data y. This method will be further described in Section 3.1.

2.3 ABC for model selection

Until now, the described ABC method only focused on parameter selection. However, it can also be applied for model selection. In this task, several models m are compared and the one that best fits the observed data y is chosen. Therefore, the parameter selection ABC needs a previous model sampling step [8].

In the end, the model with the highest probability is selected. For the model selection task, the posterior probabilities $p(\theta|y, m)$ do not need to be estimated [5]. Therefore, the set of summary statistics that is most important for parameter selection differs from the summary statistics suitable for model selection [5]. Experiments by Pudlo et al. [8] confirm this.

2.4 Challenges of ABC for model selection

The ABC approach faces two major challenges: the computational and the calibration challenge [8]. The computational challenge means that many simulations are needed. Therefore the computational cost is high in particular for large data sets. A lower computational cost can be achieved by increasing the threshold ϵ. This, however, increases the approximation error.

The calibration challenge includes the choice of hyperparameters and in particular the choice of summary statistics, which is a difficult task. A poor choice of summary statistics can result in poor performance of the model. In addition, ABC is applied in a wide field. Therefore, it is difficult to generalize in terms of summary statistics and also distance measures $\rho(.)$, because they depend on the problem and data [6]. In addition, as mentioned in Section 2.3, the suitable summary statistics for parameter selection might not be suitable for model selection.

3 ABC model selection with classifiers

For ABC model selection, an estimation of the posterior probabilities $p(\theta|y, m)$ is not needed. Therefore, classification approaches can be used. This section first introduces the classification view on ABC and then the ABC RF method, and finally compares two other classifiers that can be used in ABC methods.

3.1 Classification view on ABC

The choice of the threshold ϵ is not trivial. In addition, the best value for ϵ can vary for different summary statistics or different iterations of the algorithm. An alternative approach to using a fixed threshold ϵ is the k-nearest neighbors (kNN) approach. Instead of rejecting all models with a distance $\rho(S(x_i), S(y)) > \epsilon$, the k models with the smallest distance are returned, and therefore the threshold ϵ is chosen dynamically depending on the simulations x_i [5]. Those k returned models can be seen as an approximation for the posterior distribution $p(m|y)$. The kNN ABC algorithm for model selection is given in Algorithm 2 [5]:

Algorithm 2 kNN ABC

Require: Given are observations y and a list of models m.

for $i = 1$ to N **do**

Draw model $m_i \sim p(m)$

Draw parameters $\theta_i \sim p(\theta|m_i)$

Simulate $x_i \sim p(y|\theta_i, m_i)$

end for

return the k models m_i with the smallest distances $\rho(S(x_i)|S(y))$

Before returning the k nearest neighbors in the last step, Algorithm 2 creates a *reference table*. This table contains all model-parameter-sample combinations (m_i, θ_i, x_i) and the distances on all summary statistics for each sample x_i [8].

The kNN ABC can be seen as a classification task on the data from the reference table. However, due to the curse of dimensionality, a large number of summary statistics in this approach still leads to high computation time and poor model performance [5].

3.2 ABC-RF

In model selection, the posterior probability $p(\theta|y, m)$ does not need to be estimated. Therefore, a classification approach can be used for this task. Pudlo et al. [8] use a random forest (RF) as classifier. As in Algorithm 2, a reference table is generated for N simulations. In contrast to previously described methods, there is no rejection of samples. The whole reference table is used to train the RF classifier. The classes are the models m_i, and the training data are the summary statistics on the samples x_i. To select a model, the summary statistics of the observed data y are given into the

trained RF and the prediction is the selected model.

A random forest is an ensemble of decision trees (DTs) and uses the concepts of bootstrapping and boosting [3]. In detail, N DTs are trained on the data and, after training, predict the class with a majority vote. The N trees are trained on different subsets of the training data. In addition, each set of training samples only uses a subset of the features. Therefore, an RF needs three hyperparameters: The number N of trees, the number N_{sample} of training samples per tree and the number N_{feature} of training features per tree [8].

An RF was chosen as the classifier, because due to the majority vote the output also gives information about how often the predicted model is selected and how often other models are selected. However, the model vote frequency in RF and the posterior probability are not directly connected [8].

Another reason for RFs is their good generalization ability. Neither strong feature correlations nor noise have a large impact [8]. In addition, a large set of summary statistics can be used, which solves the calibration challenge of ABC in terms of summary statistic selection. In an experiment by Marin et al. [5], several features containing only noise were added to the model choice task. A comparison of ABC-RF and kNN-ABC showed that the prior error rate grew much slower for the RF approach than for kNN-ABC. Another advantage is that ABC-RF needs fewer training samples than other methods [8].

3.3 Other classifiers

The RF is only one classifier in a wide range of other classifier models. In this section, two of them will be described in more detail: the neural network (NN) and the support vector machine (SVM).

A NN consists of layers of neurons and weighted connections between the neurons [2]. To learn non-linear behaviour, the neurons have a non-linear activation function. In NN training, the error between output and true class is back-propagated through the network and the weights between the neurons are adapted.

The idea of support vector machines (SVMs) is to use lines, planes or hyperplanes to separate samples of different classes [2]. The goal is to find the hyperplanes with the maximum margin, which is the distance to the closest samples. The classes are often not linearly separable. Therefore, the data is non-linearly embedded into higher-dimensional space

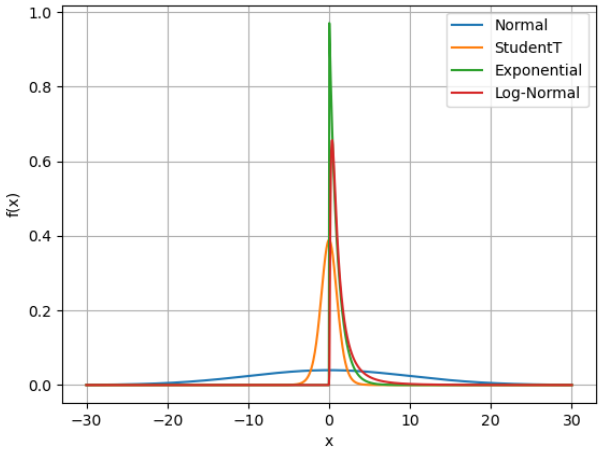


Figure 1. Distributions for the generated observed data.

with kernel methods. The separating hyperplanes are then computed in high-dimensional space and are transformed back into the original space. A commonly used type of kernel is the radial basis function (RBF) kernel, which depend on geometric distance.

4 Experiments

The experiments compare the performance of the RF, NN and SVM classifiers. Four models are used for the model selection: Model 1 is a normal distribution with mean μ and variance σ, with uniform prior distributions [-5, 5] on μ and [10, 20] on σ. The observed data y is generated with $[\mu, \sigma]$=[0, 10]. Model 2 is a student-t distribution with mean μ and ν degrees of freedom, and the prior distributions are [-5, 5] on μ and [1, 10] on ν. For model 2, the observed data is generated with the parameters $[\mu, \nu]$=[0, 10]. Model 3 is an exponential distribution with parameter λ and the prior distribution is exponential with parameter [1]. The parameter value for data generation is λ=1. Model 4 is a log-normal distribution with parameters μ and σ and uniform prior distributions [0, 1] for μ and [1] for σ. The observed data is generated with $[\mu, \sigma]$=[0, 1]. For each experiment, n=100 samples of the distribution are generated as observed data y. Figure 1 shows a plot of the four distributions.

The summary statistics used are $\sum_{i=1}^{n} x_i$ and $\sum_{i=1}^{n} x_i^2$, which refer to the mean and variance of the samples x. In addition, the reference table is normalized before the training of the classifier. In the ABC step of the model selection, N=10000 samples are generated, which means 2500

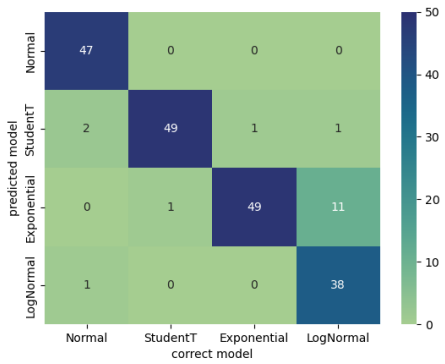


Figure 2. Confusion matrix for the RF classifier. (accuracy 0.915)

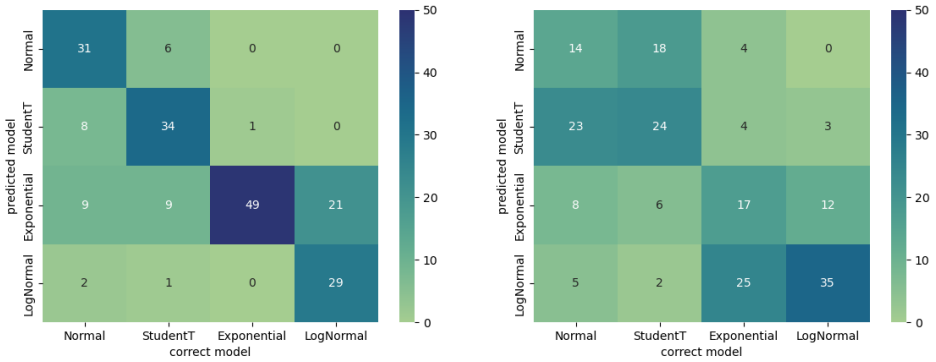
samples per model.

The model selection is performed with three different classifiers: The RF classifier, as used by Pudlo et al. [8], a NN and an SVM. The model selection is performed using the abcpy package of Dutta et al. [4] and `scikit-learn` [7] for the classifiers. The parameters of the classifiers are mostly kept at the default settings. The RF uses 100 decision trees (DTs) and each tree is trained on all data for one of the summary statistics. The NN classifier is a Multilayer Perceptron (MLP) with one hidden layer containing 100 neurons. It uses the rectified linear unit (ReLU) as an activation function for the hidden layer. The SVM uses an RBF kernel. There are 50 runs per model and classifier.

5 Results and discussion

For each classifier, a confusion matrix of the correct and predicted models is created. Figure 2 shows the confusion matrix for the RF classifier. Out of 50 models, 47 were correctly predicted for the normal distribution and 49 for both the student-t and exponential distributions. In the log-normal distribution, some models were incorrectly classified as exponential. This might be due to the fact that the distributions are quite similar, as Figure 1 shows. Overall, the RF reaches an accuracy of 0.915 for model selection.

Figure 3 shows the confusion matrices for the model selection with the NN and SVM classifiers. They both perform poorer than the RF classifier, which also becomes apparent in the accuracy values. The results of the NN classifier are similar to the RF for the exponential and log-normal distribution. However, it wrongly predicts many of the normal and student-t



(a) NN (accuracy 0.715) **(b)** SVM (accuracy 0.45)

Figure 3. Confusion matrices for the NN and SVM classifiers.

models. The SVM classifier predicts both the normal and student-t distribution mostly as student-t and second-most as normal. It similarly predicts the exponential and log-normal model mostly as log-normal and second most as exponential. A possible explanation is that the SVM weights the first summary statistic, which is related to the mean, higher than the second. The mean values are 0 for the normal and student-t distribution and larger than 0 for the exponential and log-normal distribution (see Figure 1).

The computation times for the model selection are $2.25\,\mathrm{s}$ for RF, $21.53\,\mathrm{s}$ for NN and $21.38\,\mathrm{s}$ for the SVM. Therefore, the RF classifier needs only about a tenth of the other classifiers' time for model selection. In addition, the training of RF classifiers can easily be parallelized, because each decision tree is trained individually. Therefore, RFs are a good choice when it comes to large data sets in model selection.

6 Conclusion

This paper gives an overview of likelihood-free model selection with the ABC-RF approach. The approach solves several problems of the basic ABC, including the difficult task of choosing summary statistics. In experiments, the RF classifier was replaced with two other classifiers, NN and SVM. However, the results show that the RF outperforms the other classifiers both in terms of accuracy and computation time.

References

[1] Mark A. Beaumont. Approximate bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41(1):379–406, 2010.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[4] Ritabrata Dutta, Marcel Schoengens, Jukka-Pekka Onnela, and Antonietta Mira. ABCpy: A User-Friendly, Extensible, and Parallel Library for Approximate Bayesian Computation. *Proceedings of the Platform for Advanced Scientific Computing Conference*, 6 2017.

[5] Jean-Michel Marin, Pierre Pudlo, Arnaud Estoup, and Christian P. Robert. Likelihood-free model choice, 2016.

[6] Osvaldo Martin, Ravin Kumar, and Junpeng Lao. *Bayesian modeling and computation in Python*. Texts in statistical science series. CRC Press, first edition edition, 2022.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Pierre Pudlo, Jean-Michel Marin, Arnaud Estoup, Jean-Marie Cornuet, Mathieu Gautier, and Christian P. Robert. Reliable ABC model choice via random forests. *Bioinformatics*, 32(6):859–866, 2016.

[9] Christian P. Robert, Jean-Marie Cornuet, Jean-Michel Marin, and Natesh S. Pillai. Lack of confidence in approximate bayesian computation model choice. *Proceedings of the National Academy of Sciences*, 108(37):15112–15117, 2011.

[10] Donald B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

[11] S. A. Sisson, Y. Fan, and M. A. Beaumont. Overview of approximate bayesian computation, 2018.

Animating interactions using neural networks

Janne Hölttä
janne.s.holtta@aalto.fi

Tutor: Henry Mauranen

Abstract

This literature review examines the latest developments and applications of neural networks in animating interactions. By analyzing selected articles, it is evident that neural networks have contributed significantly to generating more realistic and interactive virtual environments through the creation of more accurate and responsive animations of objects. Among the different types of neural networks, GAN-based approaches have shown great promise in the image generation field while different neural network approaches such as ManipNet have shown advantages in animating interactions. However, when it comes to scenarios with multiple animated objects involved, different neural network approaches struggle to model complex relationships in more complex interactions with dynamic objects. Additionally, creating image generation models that work well in real-life scenarios is a challenging task due to the unpredictable and diverse nature of real-world situations.

KEYWORDS: Neural network, Animation, Object

1 Introduction

Recently animation interactions has been an active research area in image recognition, facial animation and interactive character animations [11]. As deep learning techniques continue to improve, there has been a growing interest in using neural networks in animating interactions. Among other techniques, the ability of neural networks to learn complex patterns and relationships in data has played a significant role in the creation of more engaging and realistic virtual environments. In particular, the use of a variety of neural network architectures, including those that utilize GANs, have shown great promise in modeling and generating realistic animation images [10]. In this literature review, we explore the latest developments and applications of neural networks in animating interactions. Specifically, we analyze the methodology and results of selected articles and finally, discuss about the relevance and contribution of the selected articles. This literature review aims to provide insights into the current state-of-the-art techniques and challenges while highlighting their importance in this area.

2 Background

One of the articles discussed in Result section introduces neural network approaches utilizing auto-regressive models to generate facial expressions. They are used for predicting a sequence of values or events where each prediction is based on the previous predictions in the sequence. For example, the authors of the article "Example-Based Facial Animation of Virtual Reality Avatars Using Auto-Regressive Neural Networks" use an auto-regressive approach to generate facial animation [10].

The authors of the article 2021, Zhang et al. introduces neural network architecture that is based on ResNet architecture [14]. ResNet (Residual Network) is a specific type of neural network which is designed to address the problem of degradation in deeper neural networks. As more layers are added to the network, it becomes difficult to train them and the accuracy starts saturating and then degrading which is especially due vanishing problem. ResNet introduces residual connections that skip over a few layers and allow the gradients to flow directly from the output of one block to the input of another. The skip connections address the issue of vanishing gradient by providing alternate shortcut paths for the gradient to flow

through [4].

Autoencoder is a type of neural network that operates in an unsupervised manner and has the capability to compress and encode input data efficiently. It aims to learn the most salient features of the input data and create a reduced encoded representation of it. The network then tries to reconstruct the original data from this reduced representation, with the objective of minimizing the difference between the original input and the reconstructed output. Autoencoder consist of four parts: Encoder, Code, Decoder, Loss function. The encoder compresses the input data into a lower-dimensional representation or code, the code represents the compressed version of the input data, the decoder reconstructs the data from the code, and the loss function measures the difference between the reconstructed data and the original input data [3].

VAE refers to variational autoencoder which address the issue of non-regularized latent space in autoencoder and provides the generative capability to the entire space. VAE is trained with regularization to prevent overfitting and ensure that the learned latent space has useful properties for generative modeling [1]. Variational autoencoders can be used in generative modeling task such as generating new images. For example, the authors of the article 2021, Paier et al. utilize VAE among with other neural networks to generate facial expressions [10].

In addition to VAE the authors of the article 2021, Paier et al. combine GAN in their proposed model. Generative Adversial Network (GAN) is an approach to generative modelling utilizing deep learning methods, such as convolutional neural networks (CNN). GANs have been used for tasks such as image synthesis and creating art [5]. GAN method is also utilized in the article 2020, Malik et al. and in the article 2021, Belova et al.

GANs consists of two neural networks: a generator and a discriminator.The generator is trained to create fake data that are similar to real data, while the discriminator is trained to distinguish between the fake and real data. The two networks are trained together in a game-like process, where the generator tries to create more realistic fake data and the discriminator tries to correctly identify the fake data. This competition between the two networks leads to the generator becoming better at creating realistic fake data [5].

In the article 2021, Starke et al. the authors use LSTMs as a part of their proposed model to predict motion of a virtual character [11]. Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN)

architecture that is capable of storing information over long sequences of data and capturing long-term dependencies. It uses a memory cell and gates to control the flow of information, allowing it to avoid the vanishing gradient problem that traditional RNNs face when dealing with long sequences. This makes it effective for tasks such as natural machine translation and speech recognition [6].

3 Methodology

The methodology for this literature review involved conducting a literature search using the IEEE Xplore digital library with the keywords "neural netwrork", "animation" and "objects". The search was limited to peer-reviewed journal articles published between 2020 and 2023 which gave me about 32 results. After an initial screening of the search results based on relevance, abstracts and full-text articles 10 were chosen for the study. Articles were included if they described the use of neural networks in the context of animating objects, such as character animation and image animation. Exclusion criteria included articles that did not use neural networks, were not related to animation, or did not involve object. The selected articles were then analyzed to identify common neural network architectures and techniques used in animating objects. The findings of this study contribute to the understanding of the current state-of-the-art in neural network-based animation techniques and provide insights for future research in this area.

Additionally, article 2021, Starke et al. and article 2021, Zhang et al. were chosen as primary sources. The reason for selecting these articles was that they were provided by the school, and they were deemed relevant to the research topic. Also, I selected the article "Example-Based Facial Animation of Virtual Reality Avatars Using Auto-Regressive Neural Networks" by W. Paier, A. Hilsmann, and P. Eisert because I found it to be very interesting and relevant to my research topic. All of these three articles involve the use of neural networks in animation, which aligns with the research's focus on the application of neural networks in this field. Moreover, the articles provide an in-depth analysis of the methodologies used and the results obtained, making them a suitable source for drawing insights into the subject matter.

4 Results

In this section, we present the results of analysis of the selected articles on neural network-based animation of interactions. The structure is divided into two subsections based on the neural network approach used: Other approaches and GAN approaches. The subsection Other approaches includes articles that utilize architectures such as ManipNet, Long Short-Term Memory (LSTM) networks and ResNet, while the GAN approaches subsection includes articles that utilize Generative Adversarial Networks (GANs).

4.1 Approaches that utilize GAN

In the article 2021, Paier et al. the authors introduce autoregressive neural network that generates sequences of facial expressions from a learned latent representation. The network is trained on a dataset of example facial animations and can then generate new sequences by conditioning on user-defined input features. Facial expressions are generated by variational autoencoder based on a given input sequence while generative adversarial network refines the generated facial expressions to make them more realistic [10].

The goal of the proposed framework is to perform examaple-based animation which consist of three stages. In the first stage authors creates database for relevant facial expressions using captured multiview video footage. In a second step, an animatableface model is created from the captured data. To efficiently use the extracted face performance data, a neural face model variational autoencoder is trained to synthesize both geometry and texture from a low-dimensional facial expression vector. In the third step of the proposed framework, an animation network is trained on annotated sequences of latent expression vectors. These vectors are annotated with textual labels that describe the facial expression. During training, the animation network learns how to predict the subsequent facial animation parameters from the previous parameter [10].

The proposed system can synthesize realistic facial animations from a sequence of semantic expression labels after training. It several has advantages such as animating visual speech directly from text and performing fast facial animation based on a high-level description of the content [10].

The generative adversial network (GAN) is also used in the paper 2021,

Belova et al. to generate anime opening frames. The authors used a dataset of existing anime opening frames to train the GAN model. The discriminator of the GAN approach is trained to differentiate between real images from the original dataset and images generated by the generator. On the other hand, the generator is trained to create images that are realistic enough to trick the discriminator into thinking that they are real. In this way, the generator and the discriminator work together to improve the quality of the generated images until they are visually similar to the real images from the dataset. The discriminative network is constructed using convolutional layers while the generative network is implemented on the basis of deconvolution layers, using the idea opposite to convolutions [2].

The article 2020, Malik et al. presents a method for generating deepFake-based animations on driving videos and detecting such generated animations. In the the authors proposed model they animated a source image on driving video and using conditional generative adversarial networks to create new predicted images based on two inputs: a target image and several source images. The GAN was trained to learn the mapping between these inputs and to generate a new predicted image that combined the appearance of the target image with the facial expressions and movements from the source images [8].

Also, in the paper 2022, Manjula et al. the authors utilize generative adversial networks (GAN) architecture to generate deep fake images. The model is trained on a dataset of real images to learn the distribution of the real images. During the training process, the generator network is optimized to produce images that are similar to the real images, while the discriminator network is optimized to accurately classify the generated images as fake or real [9].

The article 2020, Wang et al. proposes a self-supervised approach for adapting the pose of a source image to a target domain for image animation. In the authors proposed model framework called DIPA-GAN (Domain Independent Pose Adaptation) the architecture consists of three networks: a appearance encoder, a pose encoder, and a video generator. The appearance encoder is responsible for extracting the appearance features of the input image. It takes as input the source image and generates a high-level feature representation of the appearance information. The pose encoder is responsible for extracting the pose information of the source image. It takes as input the source image and generates a high-level fea-

ture representation of the pose information. The video generator takes as input the appearance and pose feature representations generated by the appearance and pose encoders, respectively, and generates the animated output image. The proposed approach allows for the creation of realistic animated images with a desired pose, and the DIPA-GAN architecture is shown to perform this conveniently in terms of animation quality and pose adaptation accuracy [13].

4.2 Other approaches

In the article 2021, Zhang et al. propose a deep neural network model called ManipNet for synthesizing natural hand-object interactions in virtual environments. The model employes a deep neural network to learn the spatial features of hand-object interactions from data. The authors demonstrate that their model is capable of synthesizing various manipulation movements [14].

ManipNet is a autoresgressive model that consists of multiple fully connected layers incorporating both convolutional and recurrent neural networks. ManipNet predicts hand-object interactions over time using a residual dense network architecture [14]. Residual dense networks are based on the ResNet architecture. In addition to skip connections residual dense networks use dense connections which enable information to flow more efficiently through the network [4]. ManipNet takes the previous hand pose, sensor features, and control signals such as past and future trajectories of both wrists and the object as input and predicts the distance between the fingers and the object as well as a new hand pose [14].

Also, in the article 2021 by Xiuling Tian proposed method the author utilized residual networks for evaluating image aesthetic quality using a multi-task residual network. The proposed model is trained on a dataset of images with aesthetic quality labels, using a combination of classification and regression loss functions. The proposed network is designed to simultaneously perform two tasks: aesthetic quality classification and image score regression. The approach solves the degradation problem by using residual learning unit. This unit allows the network to learn residual functions, which can help to preserve important features as the network gets deeper. By preserving these important features, the performance of the network can be improved even with increased depth. The article also mentions that if the later layers of the network are identity mapping (which means they simply pass the input through unchanged),

then the network becomes a shallow network. This means that the residual learning unit is necessary for the network to have depth and be able to learn more complex representations of the input data [12].

In 2020, Ma et al. presented a method for detecting moving objects in video sequences using a 3D convolutional neural network (CNN). The paper proposes a pixel-level classifier for moving object detection, which uses a 3D convolutional neural network to classify pixels as foreground or non-foreground. The network has six layers, including four alternating 3D convolutional and pooling layers, a fully connected layer, and a softmax classifier. The input to the classifier is a video cube centered on a pixel, and the output indicates whether the pixel is a moving object. The model achieves classification by dividing pixels into a two-class model of foreground and non-foreground, with the final output of the model being the classification result. [7].

The article 2021, Starke et al. proposes a new framework for simulating interactive scenes between virtual characters and objects. The authors introduce a neural network architecture, called a neural state machine, that combines the advantages of state machines with deep learning techniques. Neural state machine consists of Motion Prediction Network and Gating Network. The Motion Prediction Network component is responsible of generating predictions for the autoregression of the character-related information. In addition, it defines the geometry of the surrounding environment and high-level instructions such as the goal location and action state. The Motion Prediction Network uses this information to predict the character's motions in the current frame. The Gating Network determinate blending coefficients of each expert (sub-network) which allows for the creation of a Motion Prediction Network that adapts to the current context. The Gation Network is used to generate more accurate predictions for the character's motion [11].

5 Discussion

Many of the articles discussed in this paper utilized GANs in their proposed model or framework. Along with multiple different neural network architecture approach GANs have shown great potential in generating animation interactions. Many studies have successfully applied GANs to generate various types of image animation such as facial expressions and deep fake images [9].

Despite the significant progress made in this field, several challenges and limitations remain, especially in the character-scene interaction field where interactions can involve multiple objects. For example, in the article 2021, Zhang et al. the authors listed limitations for their proposed model ManipNet such as that the model requires a large of amount of training data to learn the intricate relationship between the hand and the object. In the article 2019, Starke et al. also highlighted the problem where they model fails to adapt to geometry that is rather different from that in the training set [11]. Additionally, Zhang et al. acknowledged that their approach is not suitable for real-time applications due to the computational complexity of the model [14].

The authors of the paper 2021, Paier et al. faced similar limitations of their proposed model than the articles mentioned earlier. When generating facial expressions the first problem is that the approach relies on a limited set of training data which limit the generalizability of the generated animations to other facial expressions. Second, the approach struggle to work in real-world scenarios, because the input images are well-aligned and frontal which may not be the case all the time. In addition, the method may generate unrealistic or exaggerated facial expressions, which may not be suitable for all applications [10].

6 Conclusion

As a conclusion, among with other types of neural networks, GAN-based approaches have shown great success in image generation field. In addition, different neural network architectures such as ManipNet have shown significant advances in animating interactions. However, when multiple animation objects involved, different neural network approaches find it difficult to model complex relationships in more complex interactions with animation objects. Additionally, image generation in real-life scenarios is harder to implement due to unpredictable and diverse situations in real life making it difficult to capture and represent all possible variations in the training data.

References

[1] Humam Alwassel, Rui Zhang, Zeyu Wang, and Xiaofei Liu. Bimodal variational autoencoder for audio-visual speech recognition. *ResearchGate*, 2021.

[2] Polina Belova, Ksenia Urkaeva, and Anna Gamova. Generating "ideal" anime opening frames using neural networks. In *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pages 229–232, 2021.

[3] C. Borghesi and Z. Lu. Autoencoders. *ResearchGate*, 2019.

[4] Wen Chen, Yifan Wang, Jiarui Xu, and Jizhe Zhao. Neural network-based animation synthesis: A review. *Applied Sciences*, 12(18):8972, 2022.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *ResearchGate*, 2014.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *ResearchGate*, 1997.

[7] Linfei Ma, Fei Xu, Taowei Li, and Haisu Zhang. A moving object detection method based on 3d convolution neural network. In *2020 7th International Conference on Information Science and Control Engineering (ICISCE)*, pages 55–59, 2020.

[8] Yushaa Shafqat Malik, Nosheen Sabahat, and Muhammad Osama Moazzam. Image animations on driving videos with deepfakes and detecting deepfakes generated animations. In *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pages 1–6, 2020.

[9] A K Manjula, R. Thirukkumaran, K Hrithik Raj, Ashwin Athappan, and R Paramesha Reddy. Deep fakes image animation using generative adversarial networks. In *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pages 1–6, 2022.

[10] Wolfgang Paier, Anna Hilsmann, and Peter Eisert. Example-based facial animation of virtual reality avatars using auto-regressive neural networks. *IEEE Computer Graphics and Applications*, 41(4):52–63, 2021.

[11] Alexander Starke, Tobias Heck, and Dietrich Paulus. Neural state machine for character-scene interactions. In *Proceedings of the 20th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pages 225–232, 2021.

[12] Xiuling Tian. Using multi-task residual network to evaluate image aesthetic quality. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 171–174, 2021.

[13] Chaoyue Wang, Chang Xu, and Dacheng Tao. Self-supervised pose adaptation for cross-domain image animation. *IEEE Transactions on Artificial Intelligence*, 1(1):34–46, 2020.

[14] Jie Zhang, Liancheng Zhu, Ruiqi Gao, and Yue Gao. Manipnet: Neural manipulation synthesis with a hand-object spatial representation. *ACM Transactions on Graphics*, 40(4):Article 46, 2021.

Overview of Adversarial Attacks for Neural Networks Classifiers

Javier Rosales
javieralberto.rosalesflores@aalto.fi

Tutor: Blerta Lindqvist

Abstract

Neural networks can classify images with human-like performance, this is why image classifiers are now being used in all kinds of tasks, including critical ones such as medical diagnosis, malware detection, and autonomous driving. Nevertheless, image classifiers are vulnerable to evasion attacks causing them to misclassify inputs. These attacks modify images in subtle ways to cause errors. These altered images, also called adversarial examples, can be generated by various adversarial attacks. These attacks can be developed with or without using information about the target system. The high success rates of these attacks and the ability of adversarial examples to successfully attack many classifiers and not only the targeted ones make adversarial attacks a concerning matter for image classifier systems.

KEYWORDS: adversarial attacks, machine learning, attacks, security, neural networks, DNN, classifiers

1 Introduction

Neural networks, a type of machine learning system, have evolved to have near-human-level performance on multiple natural visual recogni-

tion tasks [8]. Image classification systems, one of these tasks where the system labels images, are already in use in a wide range of applications and fields.

In cybersecurity criminals typically use pre-existing malicious code to build new variants of the malware. This practice makes it highly important to classify the malware families to apply effective malware mitigation and prevention strategies. Chaganti et al. [3] propose EfficientNetB1, an efficient neural network model, which performs malware family classification using malware byte-level image representation. EfficientNetB1 achieved an accuracy of 99% on the Microsoft Malware Classification Challenge (MMCC) using malware image representation with fixed image width.

Image classifiers are also a widespread tool for medical image processing [11]. This technology has been implemented to support medical diagnoses such as the severity stage of diabetic retinopathy from retinal fundoscopy, lung diseases from chest X-ray, or skin cancer from dermoscopic photographs. The deep learning-based diabetic retinopathy diagnosis system was the first application approved by the US Food and Drug Administration (FDA).

Advanced Driving Assitance Sytems (ADAS) is constantly evolving and implementing new technologies [15]. Self-governing cars are an upcoming trend that implements image classifiers. The ability of these cars to detect, accurately classify, and act upon the different traffic signs can be the difference between life and death. This is why traffic sign classification needs to have real-time performance [4] while using images or videos from different distances as input. Even if traffic signs are designed to stand out and be easy to detect it is still a challenging task due to the wide variety of colors, shapes, environmental conditions, occlusion, and illumination.

Nevertheless, neural network classifier systems are not perfect. Neural networks learn uninterpretable solutions with counter-intuitive properties [14]. These properties make classifiers vulnerable to malicious inputs, also called adversarial examples, which can make classifier systems give wrong and undesired outputs. In [5] it is observed that the robustness of adversarial examples unveils a vulnerability in neural network systems that could be especially concerning for security-critical tasks where this kind of system is implemented.

This paper reviews the current research on the strongest evasion adversarial attacks for classifying neural networks aiming to give the reader a

better understanding of how these adversarial attacks work and how they could mean harm. This paper will cover the basics of classifying neural networks under attack, and what kinds of adversarial attacks there are.

2 Background

2.1 Neural Netwroks

As described by Carlini and Wagner [2] a neural network is a mathematical function F that takes an input $x \in \mathbb{R}^n$ and produces an output $y \in \mathbb{R}^m$.

$$F(x) = y$$

The function depends on the model parameters Θ, and neural networks can consist of one or more layers of functions.

$$F_i(x) = \sigma(\theta_i \cdot x) + \hat{\theta_1}$$

Where σ represents the non-linear activation function, θ_i represents the matrix of model weights and $\hat{\theta_1}$ a vector of model biases. This paper will focus on neural networks used as classifiers with multiple classes.

The resulting output vector y is treated as a probability distribution, where each value represents the probability that the input belongs to a specific class. The classifier assigns a label $C(x)$ to the input based on the class with the highest probability.

2.2 Adversarial Attacks

Deep neural networks have excellent performance in visual recognition problems, but they also have some drawbacks, such as the uninterpretable solutions they learn and the counterintuitive properties of these solutions. One of the strongest ones is the instability of neural networks concerning small perturbations on their inputs [14]. Inputs that are indistinguishable from natural data and are classified incorrectly due to perturbations that maximize the prediction error are named "adversarial examples". The algorithms that produce these examples are called adversarial attacks [12].

When the attack is aimed only to give a wrong classification, it's called an untargeted attack; this attack is less powerful than targeted attacks, where a specific classification is a target. An example is developed to be classified as the target class [2].

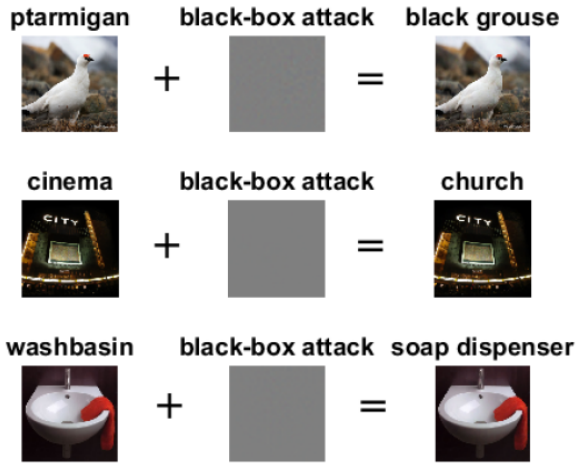


Figure 1. Natural images give the wrong classification after a black-box attack [5].

Adversarial attacks can be classified as black-box and white-box attacks. Black-box attacks characterize because the attacker has no knowledge regarding the trained model, parameters, the training dataset, or any other information available to a user with full access to the model. Black-box attacks are common in online machine-learning services. In a white-box attack, opposite to a black-box, the attacker has all the information and access to the trainer model, such as the training data set, network structure, parameters, or weights [10].

Adversarial training presents a phenomenon called transferability where adversarial examples transfer between independently trained networks with entirely different training sets. This is concerning for practical applications since it implies that other deep networks, which are not the target of adversarial training, can be vulnerable to the same examples [13].

3 Adversarial Algorithms

3.1 Fast Gradient Sign Method

Goodfellow, Shelns, and Szegedy [7] developed the fast gradient sign method (FGSM) for linear models showing that adversarial examples are not exclusive to nonlinear neural networks. Individual input features have limited precision; digital images frequently utilize only 8 bits per pixel, so all information below 1/255 is disregarded. Therefore, when an input x is added, a perturbation n, which is smaller than the precision of the features, the classifier should respond in the same way to the original input

x and the adversarial input $x\,'$. Nevertheless, when the input x goes under many microscopic perturbations, these changes add up to one large change in the output, a misclassification. These ideas are the base of the fast gradient sign method, which is defined as:

$$n = \epsilon sign(\nabla_x J(\theta, x, y))$$

Where ϵ is a negligible value to the sensor associated with the problem, θ represents the parameters of the model, x the input, y the targets associated with x, if any, and J the cost used to train the neural network. This method obtains an optimal max-norm perturbation n [7].

3.2 L-BFGS

L-BFGS is an optimization algorithm that uses limited computer memory to approximate the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS). Szegedy et al. [14] used box-constrained L-BFGS to create adversarial examples by modeling it as a constrained minimization problem, as shown below.

$$\text{minimize } \| x - x\,' \|_2^2$$

$$\text{such that } C(x\,') = l$$

$$x\,' \in [0,1]_n$$

This method aims to find another image $x\,'$ that is similar to x under the L_2 distance that is labeled differently by the classifier. The problem by itself is very tough, so Szegedy et al. added the $loss_{F,l}(x\,')$ function.

$$\text{minimize } \| x - x\,' \|_2^2 + loss_{F,l}(x\,')$$

$$\text{such that } C(x\,') = l$$

This function maps the image to a positive real number[2]. With this method, Szegedy et al. created adversarial examples that were given a wrong classification, like the following.

Figure 2. Adversarial example A[14].

Figure 3. Adversarial example B[14].

The previous figures include on the left an image correctly classified as a car, in the center the adversarial example that was not classified as a car, and on the right, the maximized absolute value of the difference between the first two images.

3.3 Jacobian-based Saliency Map Attack (JSMA)

JSMA is a white-box attack algorithm developed for image classification deep neural networks. It modifies some pixels in an image to create adversarial examples. It does this by constructing a saliency map using the Jacobian matrix, which characterizes the relationship between the input and output of a targeted deep neural network (DNN). The most significant pixel is identified and modified in each iteration based on the saliency map. The algorithm re-computes the saliency map and uses the DNN derivative with respect to the input image to indicate modifications for adversarial attacks. JSMA is not limited to image classification but has also been applied to other machine learning tasks, such as malware classification, and different DNN architectures, such as recurrent neural networks (RNNs) [5].

3.4 Deepfool

DeepFool is an untargeted attack algorithm that aims to find the least amount of distortion in an image that will result in misclassification. It is inspired by linear classification models where the separating hyperplanes indicate the decision boundaries of each class. The algorithm projects an image to the closest separating hyperplane to find the minimum distortion. A modified version of DeepFool has been proposed for DNNs to handle the nonlinearity of classification [14].

3.5 Carlini Wagner

As Szegedy with L-BFGS, Carlini, and Wagner also formulated the problem of finding an adversarial example for an image x. D stands for a distrance metric, L_0, L_2, or L_∞.

$$\text{minimize } D(x, x+\delta)$$

$$\text{such that } C(x+\delta) = t$$

$$x + \delta \in [0,1]_n$$

where the objective is to encounter a δ that minimezes $D(x, x+\delta)$ while x is fixed. Nevertheless, the constraint $C(x+\delta) = t$ is very non-linear, so Carlini and Wagner defined an objective function f such that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$. This new function changes the formulation of the problem to:

$$\text{minimize } D(x, x+\delta) + c \cdot f(x+\delta)$$

$$\text{such that } x + \delta \in [0,1]_n$$

To guarantee the modifications result in a valid image, the problem had a box constraint on δ: for all i, $0 \leq x_i + \delta_i \leq 1$.

Carlini and Wagner created attacks for L_0, L_2, and L_∞. The L_0 and L_2 attacks found examples with 2 to 10 times less distortion than Deepfool, JSMA, and L-BFGS and a 100% success probability. The L_∞ has comparable quality to the other attacks but with a higher success rate. The new attacks can use any image and find an adversarial example for any target with a runtime of no longer than a few minutes. Making the attacks unrestrictive for any attacker [2].

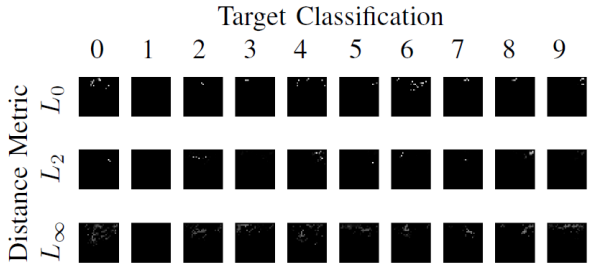


Figure 4. Targeted attacks for each of the 10 digits where the starting image is black for each of the three distance metrics.[2].

3.6 Projected Gradient Descent

The projected gradient descent (PGD) attack is a variant of the fast gradient sign method (FGSM) attack, previously discussed in this paper, consisting of a multi-step projected gradient descent on the negative loss function [13]. The perturbed data in each step is defined as follows.

$$x^t = \Pi_{x+S}(x^{t-1} + \alpha \cdot sign(\nabla_x J(\theta, x^{t-1}, y)))$$

where Π_{x+S} represents the projecting perturbatin into the set S and α the setp size. PGD is a more powerful adversary than FGSM [9].

3.7 Auto Projected Gradient Descent

Croce and Hein [6] identified weaknesses in the projected gradient descent (PGD) attack; the fixed step size, the agnostic budget, and the trend unawareness. The fixed step size greatly influences the algorithm's performance, and it is suboptimal since it does not guarantee convergence. The agnosticism of the attack budget does not ensure better results, as shown in [6] decreases significantly after a few iterations. Finally, the algorithm is unaware of how the optimization is evolving. Therefore, it is unable react to the optimization outcomes. To fix the identified weaknesses, Croce and Hein suggest dividing the available N iterations into phases.

1. Exploration: where a set of *good* initial points is searched.

2. Exploitation: where the accumulated knowledge is maximized.

The shift between phases is handled by progressively reducing the step size. A smaller step maximizes the objective function locally. On the other hand, a larger step allows so move quickly in S. The step decrease is made based on the optimization trend; if the objective value grows sufficiently fast, the step is not adjusted; otherwise, it is decreased. When the step size is adjusted, the maximization restarts from the best point encountered.

3.8 Square Attack

The square attack is a black-box query-efficient attack that implements random search, an iterative technique in optimization. The algorithm

samples a random update δ in each iteration, and if it improves the objective function, it is added to the current iterate x'. Random search does not rely on any gradient information from the objective function. The attack uses two sampling distributions, one for l_∞ and l_2, which are motivated by how neural networks with convolutional filters process images and the shaper of l_p-balls for different p. The suggested scheme varies from the classical random search by how are perturbations x' constructed. In each iteration, the perturbation is within the boundary of l_∞ or l_2 ball before the projection onto the image domain $[0,1]^d$, where d denotes the input dimension. All changes are localized in the image squares formed by contiguous pixels; thus, the attack's name [1].

3.9 Zero Order Optimization

The Carlini and Wagner (C&W) attack is the base of the Zero Order Optimization (ZOO) attack. C&W is a very strong white-box attack, but ZOO is designed as a black-box attack that modifies the two key aspects: a loss function $f(x,t)$ that is only dependent on the class label t and the output of a deep neural network F ; and computing an approximate gradient, not by using the actual backpropagation of the attacked DNN but by the use of a finite difference method and then solving the optimization problem with zeroth order optimization [5].

The proposed loss function is defined as:

$$f(x,t) = max\{max_{i \neq t} log[F(x)]_i - log[F(x)]_t, -k\}$$

$Log(0)$ is defined as - ∞ and $k \geq 0$. Since $log(\cdot)$ is a monotonic function for any x and y greater than 0, $log(y)$ is greater than $log(x)$ whenever $y \geq x$. Because of it $max_{i \neq t} log[F(x)]_i - log[F(x)]_t \leq 0$ giving x the highest confidence score for the class t. Pin-Yu et al. [5] found that the log operator was crucial for the black-box attack since DNN with extensive training yield probability distributions from $F(x)$, where the confidence score of one class is significantly higher than those of the other classes. With the log operator and its monotonicity, the dominance effect is diminished without losing the order of confidence scores since k ensures a constant gap between $max_{i \neq t} log[F(x)]_i$ and $log[F(x)]_t$

4 Conclusions

Neural network classifiers have excellent performance and a wide range of applications. They are already being used in critical tasks that make them a target for malicious users that would benefit from errors in these systems. A hacker that needs malware to go undetected, a patient that wants a wrong diagnosis, or a criminal that wants a stop sign to be misclassified to name a few.

Adversarial attacks are a reality, numerous algorithms can provide adversarial examples with high success rates. These examples can be undetectable to the human eye since the perturbations can be microscopic. Furthermore, these examples can be transferred to other classifiers and have successful results. Resulting in different systems being vulnerable to these adverbial examples, even if they are not a target and their information is not public.

This is why, when implementing a neural network classifier, it is important to implement the appropriate security measures to ensure misclassification, and suspicious behaviors do not go undetected. Classifications systems should not be blindly trusted and security checks should be in place to mitigate these attacks.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search, 2019.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

[3] Rajasekhar Chaganti, Vinayakumar Ravi, and Tuan D. Pham. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. *Journal of Information Security and Applications*, 69:103306, September 2022.

[4] Lingying Chen, Guanghui Zhao, Junwei Zhou, and Li Kuang. Real-time traffic sign classification using combined convolutional neural networks. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 399–404, 2017.

[5] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, nov 2017.

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, 2020.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] Tianjin Huang, Vlado Menkovski, Yulong Pei, and Mykola Pechenizkiy. Bridging the performance gap between fgsm and pgd adversarial training, 2020.

[10] Sara Kaviani, Ki Jin Han, and Insoo Sohn. Adversarial attacks and defenses on AI in medical imaging informatics: A survey. *Expert Systems with Applications*, 198:116815, July 2022.

[11] Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, 110:107332, February 2021.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[15] Njayou Youssouf. Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4. *Heliyon*, 8(12):e11792, December 2022.

Microservices: Describing usage based upon granularity.

Jawad Zaheer
jawad.zaheer@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

KEYWORDS: microservices, granularity, monoliths, machine learning, semantic clustering, Application Programming Interface (API)

Microservices are an increasingly used architectural pattern, hence choosing the most optimal value for it affects the software quality making it a widely researched topic. This paper conducts a literature review by identifying different approaches used by different authors to address granularity with reference to microservices using different machine learning and clustering techniques. We also evaluate the different performance and quality attributes associated with these approaches. Finally, we discuss these approaches and concluded granularity has no standard definition and that there are trade-offs between the different approaches used. Hence, a standard definition regarding microservice granularity, based on its size and complexity is proposed. Thus, we concluded that achieving low coupling, complexity, and high cohesion should be our goal which can further lead to improved performance, maintainability, and scalability for our application.

1 Introduction

Microservices are a software development approach where a large system is divided into smaller and independent services communicating and collaborating with each other, thus forming a complete application. Each microservice focuses on a specific task and can be developed, tested, and deployed independently of the others. This architecture allows for more flexible and scalable development and deployment, as well as easier maintenance and testing [3]. At present, many companies including, Netflix, eBay, and Amazon have transitioned their software and services to cloud servers, owing to their capability to adapt computing resources based on usage needs. Microservices Architecture, as described by Fowler [6], involves creating a collection of services operating like a single application. All these services communicate using methods, like, an HTTP resource Application Programming Interface (API), and run independently in separate processes [5].

Some challenges pertaining to the maintenance of distributed systems include agility, reduction of costs, and granularity [20]. Service modularization is also a major challenge since it involves figuring out which are the correct modules with the right size and the correct distribution of responsibilities. Front-End Integration User Interfaces (UIs) are very critical in microservice architectures because they are not the focus of many of the designers of the microservice approach since they are mostly architects referring to back-end portions of the application leading to systems having a monolithic front-end using a number of back-end microservices. In addition, resource management of different resources in this architecture also poses a significant challenge since necessary filters and alerts have to be defined, so as to notify developers whenever something goes wrong. Finally, building fault-tolerant services by developers in order to avoid inter-process communication failures [10].

This paper studies the problem of determining whether to use the microservices architecture in a system based on measuring the granularity of the services involved. We can define granularity based on size (number of operations and microservices) and complexity (coupling, cohesion, and dependencies). Having a lower complexity of independent services, higher cohesion, and low coupling between the different microservices involved is one of the major goals of granularity. The challenge is to find the correct boundary for each service that has a concrete purpose and is

decoupled from other services while at the same time maximizing system modularity and minimizing complexity.

As noted by Hassan et al. [8], granularity levels determine the size as well as the scope of the service-provided functionality. Adjusting granularity parameters can involve either combining or breaking down microservices to either a finer or coarser level. As suggested by Homay et al. (2020) [9], difficulty in determining the right amount of granularity for any number of services lies in determining the correct boundary size for each one, ensuring each has a clear purpose and is independent as possible, while maximizing system modularity and minimizing complexity.

The paper begins by explaining the difference between the monoliths and microservices, then discusses the different approaches that could be used to measure granularity using a systematic literature review, and finally, we would be discussing those and provide our conclusions.

2 Microservices vs Monolith

A monolithic architecture includes an application with a set of different services deployed in a single codebase rather than in a distributed system. These services communicate with external systems or consumers through interfaces, such as Representational State Transfer (REST) Application Programming Interfaces (APIs), various HTML pages, or different types of Web Pages [21].

Microservices architecture offers several benefits. Firstly, it allows for technology heterogeneity, allowing different services in the system to use different technology to achieve their goals and performance. Secondly, it provides resilience, if one component fails, it does not impact the entire system. Thirdly, It allows for easier scalability, by allowing only those services that need it rather than the entire monolithic application, thus leading to higher hardware usage. Fourthly, It allows for services to be deployed independently without affecting each other's performance. Fifthly, it aligns with the organizational structure, reducing personnel working on a single application codebase [14].

In contrast, monolithic architecture involves tightly coupling tens or hundreds of services within a single codebase, making it difficult for teams to coordinate on different independent services within the same development environment. This has led many companies to adopt the microservices architecture in favor of the monolithic one, thus facilitating better

team collaboration [18].

3 Survey Methodology

This section reviews literature using an approach introduced by Kitchenham [11]. Firstly, this section addresses the microservices granularity problem by identifying the different proposals, then identifying the metrics to be used to evaluate it and finally, then evaluate the quality attributes for those works.

3.1 Addressing Microservices Granulairty

Traditionally, the granularity of microservices was measured using trial and error approaches which depended much on the experience of the architect or developer. Moreover, it was defined according to the total lines of code, and on the basis of implementation, business capabilities, and, finally domain driven design [3]. Each paragraph below describes granularity based upon manual, automatic, and semi-automatic approaches where manual approaches are techniques performed by the developer, automatic techniques are those defined by some algorithm and semi-automatic are those which include part of the technique designed by the developer and another part by the algorithm.

A few methods have been proposed to determine how small a microservice ought to be in order to select the most optimal granularity. Some of the manual techniques proposed by Shadija et al. [17] used the university admission system as a case study and proposed a technique of evaluating a microservice in a single container using two different containers. Hassan et al. [7] provided a reference architecture using architecture definition language (ADL), which describes an architectural system based upon its elements and their relationships, for selecting the optimal granularity. Moreover, Munezero et al. [13] used a domain-driven design method, focusing on software modeling with the domain according to the input from that particular domain experts, for granularity selection, and Tyszberowicz et al. [19] used a functional decomposition method that split microservice bases upon functional capabilities into independent units.

Some of the automated approaches proposed by De Alwis et al. [4] used functional heuristics and microservice discovery algorithms on the source code to split it into different microservices for enterprise systems. It used

open-source projects, such as SUGAR CRM as a case study for evaluating this method. Mazlami et al. [12] used graph-based clustering algorithms on the source code of Git repositories for developing clusters to determine the granularity.

Semi-automated approach by Nunes et al. [15] used a clustering algorithm on aggregated domain entities as a methodology to determine granularity for call graphs and source code from different repositories, and by Ren et al. [16] used K-means clustering, Markov Chains to represent migration characteristics, which defined the factors for determining the optimal granularity for microservice applications.

3.2 Evaluating Microservices Granularity

Software metrics used at levels of design, deployment, and implementation of software and its testing and maintenance allow monitoring its different characteristics, which enables taking preventive actions in case something goes wrong.

Metrics mostly used to measure and evaluate granularity includes coupling, performance, and cohesion metrics followed by complexity measurement metrics. Coupling measures the dependencies of a software component on another which means highly coupled components cannot function independently. Thus, the low coupling is a goal between different microservices. Similarly, high cohesion between internal modules within a microservice is also one of the most important goals of a microservice. Performance Metrics, such as the number of API calls or requests, execution time, maximum request and response time, along with the number of packets sent and received are some of the most critical points for a microservice-based application. Finally, complexity metrics should also be low for different microservices enabling lesser rewrites of code thus, making them easier to improve and extend.

Coupling classes together that have the same meaning is defined as semantic coupling. A score is computed using this technique that determines how related files are in relation to the domain [12]. Candela et al. [2] defined structural coupling as classes that are outside of a package referenced by classes that are inside it divided by the total number of packages. Mazlami et al. [12] defined logical coupling with the value of one if classes (A1, A2) change together in a certain commit. Thus, they used logical coupling aggregate defined as the sum of the logical coupling for each pair of classes.

Silhouette score defined by Nunes et al [15] is the difference between the mean of the nearest-cluster distance x and the mean of the intra-cluster distance y divided by the greatest value of both. This score has a value between -1 to 1, representing incorrect clustering and highly dense clustering, respectively. Moreover, granularity metrics, such as how many microservices are used, determine the services that are part of the application or the system in question.

Shadija et al. [17] used performance metrics by measuring the response time as well as the number of API calls for different services to determine the efficacy of their technique. In [4], structural coupling and cohesion metrics, along with performance metrics, such as the number of requests and execution time were used to determine the efficiency of their heuristic and discovery algorithms. Furthermore, in [12], logical and semantic coupling was used to determine granularity efficacy for their clustering algorithm. Nunes et al. [15] used the coupling metrics Silhouette Score for its clustering algorithm, and Ren et al. [16] used granularity metrics such as a number of interfaces and microservices for evaluating their Markov Chains and K-means clustering algorithms.

3.3 Evaluating Quality Attributes Defining Granularity

Quality attributes such as availability, scaling, maintainability, performance, security, and fault tolerance are important attributes pertaining to any software application. The granularity of microservices directly affects these attributes such as increasing the microservices of a software application would increase its maintainability due to an increase in the cost of testing the application.

These software quality attributes can be divided into two broader categories namely run-time characteristics that include scalability, performance, reliability, availability, and functionality, which are features observable during the execution of software. and the other category would be software described using artifact which would include maintainability, re-usability, and modularity which are features not observable during software execution [1]. The below papers mostly focused on the run-time characteristics for evaluating the quality metrics for defining granularity.

Dharmendra Shadija et al. [17] explained the granularity and its effect on application latency by simulating a microservice-based application inside a single container and another one in multiple ones. Alwis et al. [4] addressed the scalability, performance as well as availability of an appli-

cation through the identification of parts in a consumer-based system that could be redesigned in a microservices-based architecture with high availability, scalability, and processing efficiency using a microservices-based discorvery algorithm. Genc Mazlami et al. [12] presented logical, semantic, and contributory coupling embedded in a graph-based clustering algorithm and then, measured its performance characteristics. Finally, Ren et al. [16] measured scalability and performance characteristics of run-time logs and the source code using a semi-automatic program analysis method that used a function call graph and Markov chain for representing migration characteristics, and a k-means method for hierarchical clustering. This method was useful for migrating legacy monolithic applications to a microservices architecture as well.

4 Discussions

The trend of utilizing artificial intelligence techniques for the identification of microservices or to determine their granularity is on the rise. This includes employing clustering algorithms based upon machine learning techniques as well as genetic algorithms, enabling usage of semantic similarity for grouping microservices related to similar entities. While Domain Driven Design (DDD) as well as domain level engineering techniques are still quite popular, migration of software systems necessitates a systematic evaluation of architectural decisions in order to evaluate associated risks and trade-offs between different services. This process begins with a monolithic system to be composed, and critical data sources such as the databases and the execution stack trances help identify and evaluate potential microservices. Development of microservice-based applications relates with agile development and practices, only run-time, development, deployment or production related artifacts are focused by the currently proposed methods, which may not be readily available during the design phase when starting a project from scratch.

Determining appropriate granularity is also essential, and it should be defined based on the application characteristics, available resources, operational trade-offs and non-functional requirements. Testing and deployment become more complex when dealing with numerous or larger microservices, making it important to determine the correct number of microservices needed since that will later impact upon continuous development. Few papers above used agile development artifacts as input data,

making it necessary to propose new practices or agile methods to evaluate the microservices that will make up the application. Case study is the most commonly used validation method and performance and coupling being the most frequent ones. The most addresses quality attributes includes scalability and performance alongside modularity, maintainability and fault-tolerance.

The gaps highlighted in the text are areas where further research is needed to propose new studies and ideas. Specifically, there is a need for research on techniques used to evaluate the impact of granularity on tests, using security controls and fault-tolerance mechanisms. Additional metrics are needed for the development team and process categories. There are also quite fewer proposed methods for determining granularity during testing or development phases of a software. Hence, there is a need to research for methods utilizing artifacts for agile development as inputs to figure out new agile practices for defining or assessing microservices' granularity. Finally, agile software development has need been the focus of any of the proposals identified in our review.

5 Conclusion

This literature review aimed to identify the existing research on microservice granularity, including methods and techniques for determining it. The review found that there is currently no standard definition of microservice granularity, and there are gaps in knowledge regarding the trade-offs between development and operation, continuous improvement, and conceptual reuse. To address these gaps, a definition of microservice granularity based on its size, complexity, and dependencies was proposed. The goal is to achieve low complexity and coupling along with high level of cohesion, which can lead to improved performance, maintainability, scalability, and cost-effectiveness, particularly in cloud-based deployments.

References

[1] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture In Practice*. 01 2003.

[2] Ivan Candela, Gabriele Bavota, Barbara Russo, and Rocco Oliveto. Using cohesion and coupling for software remodularization: Is it enough? *ACM Trans. Softw. Eng. Methodol.*, 25(3), jun 2016.

[3] R. Chen, Shanshan Li, and Zheng Li. From monolith to microservices: A dataflow-driven approach. *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, pages 466–475, 2017.

[4] Anuruddha De Alwis, Alistair Barros, Artem Polyvyanyy, and Colin Fidge. Function-splitting heuristics for discovery of microservices in enterprise systems. pages 37–53, 11 2018.

[5] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, pages 195–216, 2017.

[6] Martin Fowler and James Lewis. Microservices. 2014.

[7] Sara Hassan, Nour Ali, and Rami Bahsoon. Microservice ambients: An architectural meta-modelling approach for microservice granularity. 04 2017.

[8] Sara Hassan, Rami Bahsoon, and Rick Kazman. Microservice transition and its granularity problem: A systematic mapping study. *Software: Practice and Experience*, 50, 06 2020.

[9] Aydin Homay, Mario de Sousa, Alois Zoitl, and Martin Wollschlaeger. Service granularity in industrial automation and control systems. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 132–139, 2020.

[10] Pooyan Jamshidi, Claus Pahl, Nabor das Chagas Mendonça, James Lewis, and Stefan Tilkov. Microservices: The journey so far and challenges ahead. *IEEE Softw.*, 35:24–35, 2018.

[11] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33, 08 2004.

[12] Genc Mazlami, Jürgen Cito, and Philipp Leitner. Extraction of microservices from monolithic software architectures. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 524–531, 2017.

[13] Immaculée Josélyne Munezero, Doreen-Tuheirwe Mukasa, Benjamin Kanagwa, and Joseph Balikuddembe. Partitioning microservices: A domain engineering approach. In *2018 IEEE/ACM Symposium on Software Engineering in Africa (SEiA)*, pages 43–49, 2018.

[14] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 1st edition, February 2015.

[15] Luis Nunes, Nuno Santos, and António Silva. *From a Monolith to a Microservices Architecture: An Approach Based on Transactional Contexts*, pages 37–52. 09 2019.

[16] Zhongshan Ren, Wei Wang, Guoquan Wu, Chushu Gao, Wei Chen, Jun Wei, and Tao Huang. Migrating web applications from monolithic structure to microservices architecture. In *Proceedings of the 10th Asia-Pacific Symposium on Internetware*, Internetware '18, New York, NY, USA, 2018. Association for Computing Machinery.

[17] Dharmendra Shadija, Mo Rezai, and Richard Hill. Microservices: Granularity vs. performance. arXiv, 2017.

[18] Vindeep Singh and Sateesh Kumar Peddoju. Container-based microservice architecture for cloud applications. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 847–852, 2017.

[19] Shmuel Tyszberowicz, Robert Heinrich, Bo Liu, and Zhiming Liu . Identifying microservices using functional decomposition, 08 2018.

[20] Mario Villamizar, Oscar Garces, Harold E. Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. *2015 10th Computing Colombian Conference (10CCC)*, pages 583–590, 2015.

[21] Mario Villamizar, Oscar Garcés, Lina Ochoa, Harold Castro, Lorena Salamanca, Mauricio Verano Merino, Rubby Casallas, Santiago Gil, Carlos Valencia, Angee Zambrano, and Mery Lang. Cost comparison of running web applications in the cloud using monolithic, microservice, and aws lambda architectures. *Service Oriented Computing and Applications*, 11, 06 2017.

Docker Container Networking for Local-Network Applications

Je-Ruei Yang
je-ruei.yang@aalto.fi

Tutor: Tuomas Aura

Abstract

This paper explores the use of various Docker network drivers, particularly the MACVLAN and IPVLAN drivers, as alternatives to the commonly used Bridge and Host drivers. This study focuses on non-cloud applications of containers, such as deploying containerized servers in the local network. We analyze the connectivity, isolation, performance, and usability of each driver. Additionally, we propose application scenarios to which each driver is best suited and evaluate them through experiments. The experiments demonstrate that MACVLAN and IPVLAN drivers offer superior performance while maintaining isolation and simplicity. Our study suggests that the MACVLAN and IPVLAN drivers are preferable for small-to-medium-sized projects. These drivers have the potential to significantly improve the performance and flexibility of containerized applications.

KEYWORDS: Docker, container networking, MACVLAN, IPVLAN

1 Introduction

In recent years, the container has become one of the most prominent technologies for virtualizing and deploying applications. Containers provide virtualization that enables multiple applications on a single host to run safely

and securely. Furthermore, containers are more lightweight than traditional virtual machines (VMs) as they share the host operating system (OS). As a result, containers have characteristics such as rapid startup time and negligible overhead to the host [1].

Although containers offer advantages over VMs in resource usage, they present challenges in the communication network. Containers lack a complete network stack and individual network addresses since they are launched as processes [2]. However, it is crucial to facilitate communication between containers while maintaining isolation. Container network addresses this by providing abstraction models on connectivity and isolation [3]. These models enable software developers to implement network drivers or plugins to achieve traditional networking capabilities.

Docker [4], a widely used container runtime, offers network drivers out of the box to provide core networking functionality. These network drivers can be classified into single-host and multi-host settings. Single-host network drivers aim to provide containers with network interfaces for containerized applications, while multi-host network drivers focus on interconnecting containers on different hosts [5]. Despite the power of orchestration tools, such as Kubernetes [6], in multi-host settings, these tools can be intricate and require significant learning effort. Standalone Docker without an orchestration tool is suitable for small- to medium-sized projects. Understanding Docker networking solutions increases flexibility on such projects while keeping simplicity.

This paper overviews the state-of-the-art container networking models and dives deep into the Docker network drivers, focusing on the single-host and non-cloud settings. Specially, we explore the MACVLAN and IPVLAN drivers, which facilitate direct connection to the physical networks. By comparing the strengths and weaknesses, we propose and implement example use cases that illustrate the optimal scenarios for each driver.

The remainder of this paper is organized as follows. Section 2 and 3 introduce the container networking models and the Docker network drivers. Section 4 and 5 propose and implement use cases where each network driver is most suitable. Finally, Sections 6 and 7 evaluate and discuss the experiment results and summarize the paper.

2 Container Networking Models

This section briefly introduces the designs and features of the two mainstream specifications for container networking: *Container Network Model*

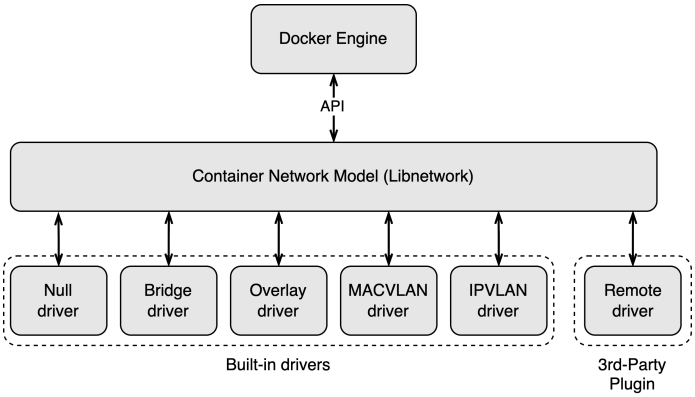


Figure 1. CNM architecture [3, 9, 10]

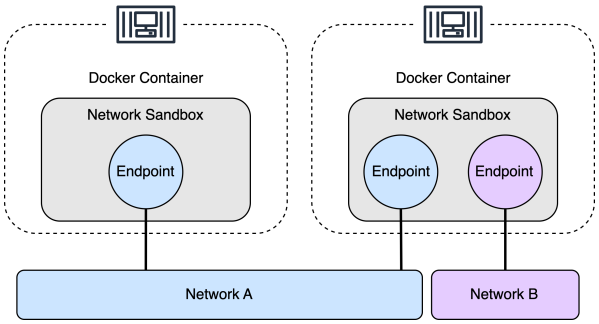


Figure 2. CNM components and their relationships [8]

(CNM) and *Container Network Interface (CNI)*.

2.1 Container Network Model (CNM)

CNM [7] is a standard proposed by Docker that defines the interface between a user, such as Docker Engine, and network drivers. Libnetwork [8] is an implementation of CNM and is used as the basis for Docker networking. Figure 1 shows the architecture of the CNM, and Figure 2 outlines the fundamental components of CNM and their relationships. As shown in Figure 2, CNM has three components: sandbox, endpoint, and network.

Sandbox is an isolated environment that contains the network specification of a container, such as network interfaces, IP addresses, MAC addresses, and routing and DNS configurations [8, 10].

Endpoint acts as a virtual network interface located in a sandbox and is responsible for joining the sandbox to a network. It should be noted that one endpoint belongs to one sandbox and one network, whereas one sandbox may have multiple endpoints.

Network is a group of endpoints that can communicate with one another. Each network is in a separate network namespace on the host OS [3]. This creates network isolation for the containers.

2.2 Container Network Interface (CNI)

CNI [11] is a specification proposed by CoreOS that acts as a lightweight contract between a container runtime and network plugins [3]. CNI is minimalistic and is able to intergrate with any container runtime [9]. As a result, CNI is adapted and used by container orchestration tools, such as Kubernetes, as their networking solution. Since the network solutions of orchestration tools are out of the scope of this paper, we will not dive deeper into this model.

3 Docker Network Drivers

As mentioned in Section 2.1, CNM forms an interface between Docker Engine and network drivers. Network drivers are network implementations that provide core networking functionality [12]. They handle connectivity and isolation between containers [10]. Docker ships with built-in network drivers on Linux, including Bridge, Host, Overlay, MACVLAN, and IPVLAN. The following sections introduce the main characteristics and features of each driver. In addition, we indicate the connectivity, isolation, and advantages and disadvantages of each driver.

3.1 Bridge

The Bridge driver is the default option for Docker containers on a single host [5]. When starting a Docker container without specifying the network type, the container will connect to the default Bridge driver. In addition to the default Bridge driver, users can create a user-defined Bridge for improved isolation and built-in DNS resolution.

In a Bridge network, each container is assigned a private IP address, and the containers connecting to this network are within the same subnet. The default Bridge network enables the containers to communicate using private IP addresses. Furthermore, containers in a user-defined Bridge network can utilize DNS resolution and communicate using the container names as domain names. Lastly, containers connecting to different Bridge networks cannot communicate, thereby ensuring isolation.

The Bridge driver is easy to use and is suitable for most scenarios. In conjunction with this, the DNS resolution functionality available within user-defined Bridges decouples the containers from relying on potentially changing private IP addresses. Consequently, applications involving multiple containers are more robust by leveraging such functionality.

On the negative side, the Bridge driver suffers from performance over-

head due to network address translation (NAT) while accessing external networks [5]. Moreover, ports must be explicitly published to make the container visible from the external network. This might create problems such as port conflicts. For example, a containerized DNS server might conflict with the host default DNS resolver on the standard DNS port 53.

3.2 Host

The Host driver uses the same network stack as the host machine. As a result, all containers connected to the host network have the IP address of the host. Containers connected to the host network can communicate with one another via the ports they expose.

The Host driver allows near bare metal performance [13] and is suitable for cases where the container needs to handle an extensive range of ports [14]. However, the Host driver has two main drawbacks. First, it has no isolation between containers using the host driver, which poses security risks. Second, the host driver also suffers from port conflict problems. This limits the number of containerized services on a single host [13].

3.3 Overlay

The Overlay driver enables the communication between multiple hosts. This driver is utilized by Docker's orchestration tool swarm to create multi-host networks spanning multiple Docker hosts. Since the Overlay driver is used mainly in multi-host settings, we will not discuss it further in this paper.

3.4 MACVLAN

The MACVLAN driver makes the containers appear as connecting directly to the physical network. It creates virtual network interfaces behind the physical interface of the host and assigns unique MAC and IP addresses to each container [15]. Therefore, the MACVLAN driver is suitable for tackling legacy applications that require separate MAC addresses [16]. Furthermore, the MACVLAN driver eliminates port mapping and provides better performance [10] than the Bridge driver.

However, MACVLAN requires networking devices to work in promiscuous mode, allowing multiple MAC addresses on an interface. The requirement is unachievable if the user lacks permission to access the host network settings. Moreover, the MACVLAN driver is unable to obtain an IP address from external DHCP, for instance, the DHCP server on the home router. This necessitates manually assigning IP addresses outside the DHCP range or

through the Docker compose configuration.

There are two modes for the MACVLAN driver: bridge mode and 802.1Q trunk bridge mode. In bridge mode, network traffic goes through a physical interface on the host. On the other hand, in the 802.1Q trunk bridge mode, network traffic goes through an 802.1Q sub-interface [16], which is useful when users have existing VLANs on their physical network.

3.5 IPVLAN

The IPVLAN driver is similar to the MACVLAN driver, except containers connecting to an IPVLAN network have the same MAC address. This solves the promiscuous mode issue while maintaining the physical appearance advantage. Moreover, since IEEE 802.11 wireless networks allow each wireless station to have only one MAC address, the MACVLAN driver is generally not supported by wireless networks [17]. The IPVLAN driver is a preferred alternative in this use case.

The IPVLAN driver has three modes: L2 mode, 802.1Q trunk L2 mode and L3 mode. The L2 mode is the default mode for IPVLAN. The IPVLAN L2 mode and IPVLAN 802.1Q trunk L2 mode are analogous to the MACVLAN bridge mode and the MACVLAN 802.1Q trunk bridge mode. They require the IP addresses to be in the same subnet as the physical network [15]. Containers within the same IPVLAN L2 network can communicate with one another either using an IP address or domain name.

In contrast, the IPVLAN L3 mode offers completely different functionalities. Instead of connecting to the physical network, containers in IPVLAN L3 mode create virtual subnets (different than the parent interface) within the host and make the parent interface acts as a router [18]. Containers in an IPVLAN L3 network can communicate even if they are in different subnets. This mode aims to give total control and finer granularity over the routing for network professionals. However, since the container in an L3 network does not act as a physical device on a physical network and is isolated from the external network. We will not discuss this mode further.

4 Scenarios

Connectivity, isolation, and performance are the key aspects to consider in container networking problems. This section compares local-network and non-cloud settings scenarios with these notions to find suitable network drivers for such use cases.

4.1 Applications with multiple containers

A common scenario is a web application, which usually comprises a web server and a database. Generally, exposing the database to the external network is undesirable. Therefore, it is prudent to employ the Bridge driver to establish connectivity between the web server and the database while maintaining segregation between the database and the external network. Nonetheless, the overhead of publishing ports for the web server still exists. We will propose a solution to this issue in the following section.

4.2 Applications with standard ports

Specific applications operate on standardized ports, such as port 53 for DNS and port 80 for HTTP. However, these ports may already be used by applications on the host, resulting in a port conflict when running a containerized version of the same application. Consequently, the Bridge and Host drivers are inadequate for this scenario and implementing MACVLAN or IPVLAN drivers is recommended to circumvent the issue.

In addition, when multiple containers of the same type are operated simultaneously, utilizing MACVLAN or IPVLAN drivers may be beneficial. A typical scenario for developers is the operation of multiple websites on the same host. In such cases, the MACVLAN or IPVLAN drivers simplify the process of managing a large number of published ports.

4.3 Performance-critical application

Particular applications, such as media and gaming servers, require high network performance to operate smoothly. The Host, MACVLAN, and IPVLAN drivers optimize the network performance for containerized applications of this nature. Among these options, the MACVLAN and IPVLAN drivers provide better isolation than the Host driver. This makes them more suitable for scenarios where enhanced network performance is essential while maintaining isolation between the application and the host.

4.4 Network-level security application

Applications that handle sensitive data may require network-level security to protect against security threats, such as unauthorized access. This could be, for instance, government applications that process personal identities and hospital applications that store patient data. However, these organizations may have a limited number of servers, restricting their ability to run all applications under distinct IP addresses. MACVLAN and IPVLAN drivers are instrumental in

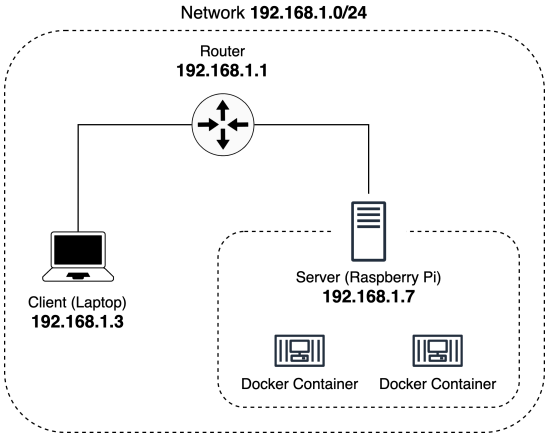


Figure 3. Experiment setup

this scenario. They provide applications with separate IP addresses and allow an external firewall to have individual filtering rules for the host and each containerized service. Moreover, MACVLAN and IPVLAN 802.1Q trunk mode also help the network administrator to isolate the services into different VLANs.

5 Experiment and Evaluation

This section demonstrates the advantages of MACVLAN and IPVLAN drivers by implementing the scenarios mentioned in Section 4.2 and 4.3. The experimental environment is composed of three devices within a private network 192.168.1.0/24: a router (192.168.1.1), a laptop acting as a client (192.168.1.3), and a Raspberry Pi acting as the server (192.168.1.7). Figure 3 shows the setups of the experiment.

5.1 Multiple web servers on the same host

This experiment shows how MACVLAN and IPVLAN drivers solve the port conflict issue. We run three containerized instances of Nginx [19] web servers on the Raspberry Pi host using the Bridge, MACVLAN, or IPVLAN driver, one at a time. The experiment involves measuring the port availability of each driver and verifying that multiple Nginx web server instances can run simultaneously without any port conflict issues.

5.1.1 Bridge driver

The Bridge driver restricts the visibility of the containers to the IP address of the server (192.168.1.7). Only one of the three containers can be published on standard HTTP port 80. This necessitates the allocation of non-standard ports. As a result, users must additionally specify a port number while accessing the application via a web browser.

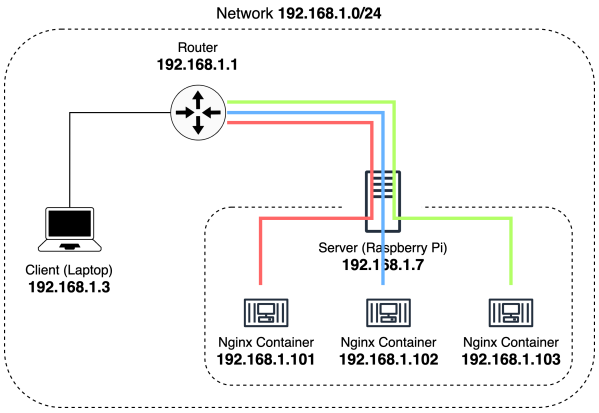


Figure 4. Multiple containerized web servers connected with MACVLAN/IPVLAN driver

5.1.2 MACVLAN/IPVLAN driver

Adapting the MACVLAN/IPVLAN driver allocates IP addresses for the containers, therefore, removing the need for non-standard port allocation. In this experiment, the containers are assigned to IP addresses ranging from 192.168.1.101 to 192.168.1.103. Figure 4 shows the outcome of this experiment. These separate IP addresses facilitate access via the web browser from clients in the same local network without specifying a port number. Furthermore, when combined with a nameserver, these IP addresses can be mapped to human-friendly domain names.

5.2 Performance test on a web server

This experiment shows how the MACVLAN and IPVLAN drivers provide high network performance while maintaining isolation between the container and the host. We run an Nginx container on the Raspberry Pi using Bridge, Host, MACVLAN, or IPVLAN driver, one at a time. The network performance of each network driver is measured by the client using Grafana k6 [20]. Figure 5 shows the experiment results. The Host, MACVLAN, and IPVLAN drivers statistically have better performance in terms of round-trip time for a request. The maximum is not shown since the first round-trip is typically an outlier. Especially for the IPVLAN driver, the first round-trip is in the range of 1000 to 2000 ms.

6 Discussion

In the previous section, we showed that MACVLAN and IPVLAN drivers effectively solve the port conflict problem, provide better network performance than the Bridge driver, and offer better isolation than the Host driver. They can be a great fit for developers' home lab projects for their flexibility, performance,

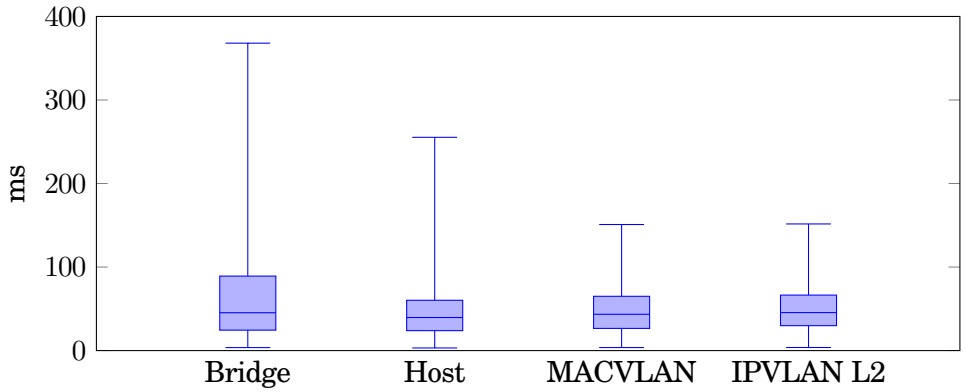


Figure 5. The round-trip time for a request over a one-minute period with a hundred parallel virtual users. The figure shows the minimum (3–4ms), 25th, 50th, 75th, and 99th percentiles.

and simplicity. The remaining decision between MACVLAN and IPVLAN depends on whether the user has permission to enable the promiscuous mode on network devices, whether the application needs a separate MAC address, and whether the parent interface uses a wireless connection.

Although the MACVLAN/IPVLAN drivers have some handy features, some places still require improvements. One example is the IP allocation for newly created containers. Currently, there is no official support to communicate with an external DHCP server. As a result, the simplest but still cumbersome approach is to assign an IP address for each container in the container configuration. There are open-source projects [21, 22] aiming to solve this problem, but they are neither mature nor production-ready.

7 Conclusion

This paper explores various container network models and Docker network drivers, focusing on the MACVLAN and IPVLAN drivers in a non-cloud local setup. By comparing the drivers, we propose several use cases where each driver is most suitable, such as multi-container and performance-critical applications. Moreover, we implement and evaluate the scenarios we proposed through various experiments.

The experiments highlight the benefits of the MACVLAN and the IPVLAN drivers, including having a direct connection to the physical network and ensuring isolation while maintaining high performance. Our study suggests that the MACVLAN and IPVLAN are preferable for small-to-medium-size projects. These drivers have significant potential for enhancing the performance and flexibility of containerized applications.

References

[1] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. An updated performance comparison of virtual machines and Linux containers. In *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pages 171–172. IEEE, 2015.

[2] Lucas Litter Mentz, Wilton Jaciel Loch, and Guilherme Piegas Koslovski. Comparative experimental analysis of Docker container networking drivers. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–7. IEEE, 2020.

[3] Hao Zeng, Baosheng Wang, Wenping Deng, and Weiqi Zhang. Measurement and evaluation for Docker container networking. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 105–108. IEEE, 2017.

[4] Docker. https://www.docker.com/ (Accessed Feb. 02, 2023).

[5] Kun Suo, Yong Zhao, Wei Chen, and Jia Rao. An analysis and empirical study of container networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 189–197. IEEE, 2018.

[6] Kubernetes. https://kubernetes.io/ (Accessed Feb. 02, 2023).

[7] Container network model. https://github.com/moby/libnetwork/blob/master/docs/design.md (Accessed Feb. 02, 2023).

[8] libnetwork - networking for containers. https://github.com/moby/libnetwork/ (Accessed Feb. 02, 2023).

[9] Lee Calcote. The container networking landscape: CNI from CoreOS and CNM from Docker. https://thenewstack.io/container-networking-landscape-cni-coreos-cnm-docker/ (Accessed Feb. 14, 2023).

[10] Nigel Poulton. *Docker Deep Dive*. Packt Publishing, 2020.

[11] CNI – the container network interface. https://github.com/containernetworking/cni (Accessed Feb. 14 2023).

[12] Networking overview | Docker documentation. https://docs.docker.com/network/ (Accessed Feb. 16, 2023).

[13] Ubaid Abbasi, El Houssine Bourhim, Mouhamad Dieye, and Halima Elbiaze. A performance comparison of container networking alternatives. *IEEE Network*, 33:178–185, 2019.

[14] Host networking | Docker documentation. https://docs.docker.com/network/ (Accessed Feb. 16, 2023).

[15] Lee Calcote. Container networking: A breakdown, explanation and analysis. https://thenewstack.io/container-networking-breakdown-explanation-analysis/ (Accessed Feb. 20, 2023).

[16] Macvlan networks | Docker documentation. https://docs.docker.com/network/macvlan/ (Accessed Feb. 20, 2023).

[17] Peini Liu and Jordi Guitart. Performance characterization of containerization for HPC workloads on InfiniBand clusters: an empirical study. *Cluster Computing*, pages 1–22, 2022.

[18] Ipvlan networks | Docker documentation. https://docs.docker.com/network/ipvlan/ (Accessed Feb. 20, 2023).

[19] Nginx: Advanced load balancer, web server, & reverse proxy. https://www.nginx.com/ (Accessed Mar. 5, 2023).

[20] Grafana k6: Load testing for engineering teams. https://k6.io/ (Accessed Mar. 18, 2023).

[21] Jack O'Sullivan. docker-net-dhcp. https://github.com/devplayer0/docker-net-dhcp (Accessed Mar. 5, 2023).

[22] Brent Salisbury. Experimental Docker libnetwork DHCP driver. https://gist.github.com/nerdalert/3d2b891d41e0fa8d688c (Accessed Mar. 5, 2023).

Audio-Visual Speaker Recognition using Deep Learning: A Survey

Jiehong Mo
jiehong.mo@aalto.fi

Tutor: Abduljalil Saif

Abstract

The confluence of visual and auditory data has emerged as a promising solution for speaker recognition (SR) applications, where it has given rise to the concept of audio-visual speaker recognition (AVSR). The implementation of deep learning methods into this domain has contributed to the acceleration of its advancement. However, AVSR shows a paucity of review articles, especially in the area of deep learning, hindering researchers from gaining insight into the field. This paper presents a survey on AVSR systems in perspective of performing thorough analysis to find the gaps for future. AVSR systems were branched into three major types, which are audio-visual speaker identification, audio-visual speaker verification, and audio-visual speaker diarization. Distinctive aspects and main shortages for various types of AVSR systems are provided. Despite of several promising AVSR methods in the literature, it is still unsolved research topic. The integration of the visual and audio systems poses a challenge to AVSR.

KEYWORDS: Audio-visual Recognition, Speaker Recognition, Deep Learning

1 Introduction

Voice is one of the physical characteristics of human beings can be used to identify speakers. Based on this logic, automatic speaker recognition research (ASR) has been developed. This research has piqued the interest of researchers since it can be used in a wide range of applications, such as surveillance [11].

In the last two decades, with the advancement of deep learning algorithms, deep learning has brought great developments in SR. For deep learning ASR, a considerable amount of research has been conducted to exploit a single modality, which is the audio or visual, mostly based on audio only [16]. Recently, due to the complementary nature of audio-visual (AV) biometrics, its research has attracted attention [2]. Researchers have found that visual cues carry complementary information on the audio cues, and its integration into an ASR models significantly improves the performance [18]. Furthermore, the National Institute of Standards and Technology (NIST) has proved that AV fusion significantly improves performance in comparison to audio-only or visual-only systems [21]. Recently, the major research challenge in AVSR is related to the best methods of integrating visual and auditory modalities.

As a new direction of research, AVSR has few review articles and almost none related to deep learning. The main goal of this paper is to fill this gap by reviewing existing AVSR systems published in the last five years. This paper will provide the advantages and disadvantages of each type of AVSR and discuss potential research directions for each.

This paper is organized as follows. Section 2 presents the categories in AVSR. Sections 3, 4 and 5 review the current approaches of the three taxonomies, respectively, while Section 6 introduces the datasets for AVSR. Finally, Section 7 concludes this work.

2 Audio-Visual Speaker Recognition

This section presents the most prominent way of classification of speaker recognition [13]. In general, speaker recognition systems can be divided into three types: speaker identification (SI), speaker verification (SV), and speaker diarization (SD). Similarly, as a branch of speaker recognition, AVSR also follows this classification [27]. The three types of speaker recognition will be briefly described below, and Fig. 1 depicts the taxon-

omy of SR systems.

Speaker identification (SI) uses the speech information to determine the identity of a speaker. This requires comparison with multiple known speakers in the dataset to obtain the identity of the recorded speaker.

Speaker verification (SV) utilizes the voice information to determine if a recording is belong to an individual. In other words, it verifies the claimed identity of a speaker.

Speaker diarization (SD) segments the audio signal into distinct segments based on the speaker identity, to determine when and who is speaking. Active speaker detection (ASD), which refers to the process of identifying who is currently speaking in a multi-party audio recording or video conference, is a critical prior step and can be broadly classified as SD [31].

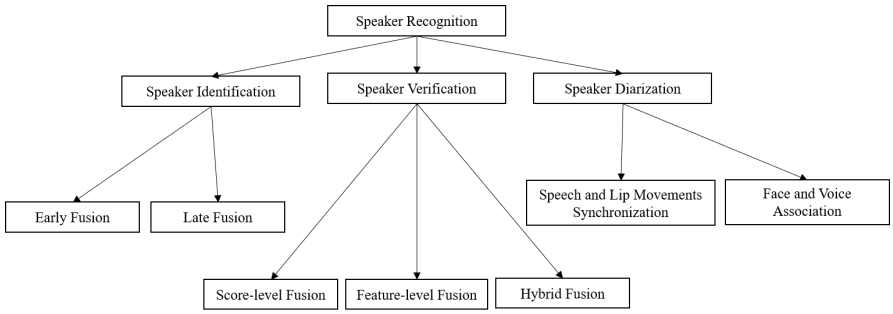


Figure 1. The taxonomy of SR systems

3 Audio-visual Speaker Identification (AVSI)

In 2019, the NIST Speaker Recognition Evaluation (SRE), a benchmark test for many systems, investigated a new direction on AVSR, causing this direction to gain more attention from academic researchers [22]. Several mechanisms of AV fusion have been introduced for AVSR models, including early fusion and late fusion.

3.1 Early Fusion

Early fusion is the processing of audio and visual embedding as general identity features as input to the system. For instance, a two-branch network was proposed to learn joint representations of faces and voices in a multimodal system before extracted features are used to train a classifier for speaker recognition [25].

3.2 Late Fusion

Late fusion feeds audio and visual inputs to the speaker recognition and face recognition systems, respectively, and then combines the results of both systems. Late fusion is preferred by many researchers because of its simplicity [29]. Late fusion only merges results of two systems without altering their components. While purely visual deep learning speaker recognition has generally not existed in the past, many mature systems exist for both speaker recognition and facial recognition tasks [25]. Researchers only need to combine the two systems to solve the AVSI task. Accordingly, the researchers manage to enhance the performance of the two subsystems separately to improve the overall performance.

Different strategies were used in the late fusion, such as the score-level strategy [9] and the greedy strategy [29]. Score-level strategy aggregates the scores from each model into a single score, which is then used as the final decision. In greedy strategy, the individual systems are calibrated and evaluated, and the best system is selected based on its performance. Then, all possible combinations that include the best system are evaluated, and the best two-system fusion is selected based on its actual cost. Although the score-level strategy is simpler and easier to implement, it may not be as effective as the greedy strategy. On the contrary, the greedy strategy is more likely to fall into the overfitting.

From our viewpoint, as early fusion has only one input, it can lead to simple computations. However, early fusion does not take into account the reliability of either modality. if a modality has much noise, the AV feature vector will be compromised and catastrophic fusion may occur. In late fusion, each modality can be processed independently, allowing more specialized models to be built for each modality, which enhances the performance. Nevertheless, the audio and visual modalities are disjoint and one does not take into account the knowledge of the other, which can lead to some loss of information.

4 Audio-visual Speaker Verification (AVSV)

A common method of SV is to extract a representative speaker embedding from a given corpus and compare pairs of embeddings with the distance metric to identify whether the given corpus belongs to the same person

[23]. For AVSV, it is essential to incorporate visual information into this process, and this section classifies the network according to the different levels of fusion of visual and audio information.

4.1 Score-level Fusion

Score-level fusion strategy is to assign the best weights to model scores from multiple biometric sources and then combine these scores [14].

In the 2019 NIST SRE Challenge, Grigory et al. [3] explored automated quality assessment that automatically estimates the quality of representations produced by the single-modality system, resulting in enhanced fractional-level fusion of multimodality.

Although score-level fusion is simple, it is still competitive among other strategies. For the three-model system, Madin et al. [1] compared embedding-level fusion with fractional-level fusion and demonstrated that the latter achieved better results. This may be because for three subsystems, including thermal facial system, visual system and audio system, coarse-grained fusion is a better representation of the overall performance of the system by achieving a more balanced fusion than the uneven fusion that results from fine-grained fusion.

4.2 Feature-level Fusion

The feature-level fusion represents the fusion of the extracted audio features with the visual features [24]. In AVSV, most of the time, the features are represented as embedding, hence they can also be called embedding-level fusion. In this strategy, the attention mechanism is frequently applied to feature fusion, since it dynamically identifies salient features as needed, without compressing the overall message into vague abstractions, and this combination of audio-visual salient features leads to better results.

In 2019, a feature-level fusion approach based on a neural network model was proposed that uses an attention mechanism to evaluate the face and speaker representation contributions extracted from each subsystem and combine them to obtain a joint representation [26]. This attention mechanism can learn to evaluate the salient modality of input data to produce a strong fusion representation. In addition to the attention mechanism, Chen et al. [4] proposed more embedding fusion methods, including simple soft attention fusion, multi-modal compact bilinear

pooling fusion, and gated multi-modal, and then conducted experimental comparisons (more information can be found in [4]). The three feature fusion strategies showed significant improvements over the single-modal system, with the gated multimodal fusion architecture performing the best. However, score-level fusion outperformed all three strategies. The likely reason is the presence of a more powerful modality in this work.

4.3 Hybrid Fusion

In addition to using the score-level and feature-level fusion methods individually, it is also possible to use a mixture of them. A multi-view approach was introduced by Facebook which generates high-level representations for audio and video modalities in a space shared across the two modalities with feature-level fusion [23]. It also showed that the best performance was obtained by combing two fusion strategies.

From our perspective, for feature-level fusion, the learned features of the modalities share a compatible space, resulting in a fused representation. It performs better than traditional fractional-level fusion and has advantages in severe cases with corrupted or missing modalities.

In terms of the advantages of score-level fusion, score-level fusion simply needs the results of modalities, making it easier to apply to the model. Moreover, it is more likely to perform better when a single modality is overpowered. Score level fusion can also help avoid overfitting. As it only uses the final decision or score, which has already integrated information from all modalities, this makes it more representative of the overall performance and less likely to lead to overfitting. In contrast, feature-level fusion may lead to overfitting due to an overly fine-grained feature extraction process.

Currently, feature-level fusion be further differentiated into multiple fusions characterized by varying granularity. This approach has garnered superior outcomes in comparison to the traditional score-level method on many models. As a result, one of the ongoing research objectives involves optimizing the extraction of AV features and effecting a more astute fusion. However, score-level is sometimes comparable to surpasses feature-level. Therefore, researchers need to consider which is the appropriate fusion strategy. In addition, some researchers are considering hybrid fusion to compensate for shortcomings and improve overall results.

5 Audio-visual Speaker Diarization (AVSD)

For AVSD, visual information can help with speech overlap and predicting active speakers among candidate speakers [31]. In particular, lips and facial features are highly correlated [32]. Consequently, researchers have studied the association of AV information in AVSD based on both lip visual information and facial visual information. Therefore, this section divides AVSD into two categories based on the association between talking faces and voice tracks and on synchronization between utterances and lip movements.

5.1 Speech and Lip Movements Synchronization for SD

Lip synchronization refers to determining the movement of the mouth and tongue during speech [17]. Lip information is often used as visual information in the field of speech synchronization [8], which is an important prior step in AVSD and is sometimes used by AD tasks only as part of the network [10]. This has led researchers to improve overall performance with lip synchronization.

Recently, an end-to-end multimodal model was proposed to distinguish speech from non-speech regions in an audio segment [12]. It is based on audio features, multi-speaker regions of interest (ROI) of lips and multi-speaker i-vector embeddings. This model also explores the impact of different levels of lip absence on speaker diarization through experiments where fragments of lip ROIs were manually removed. Similarly, Abudukelimu et al. [30] proposed a Dynamic Vision-Guided Speaker Embedding (DyViSE), which uses dynamic lip motion information to denoise the audio in latent space and then fuses the denoised audio and facial features together to obtain an embedding of each segment's identity. Unlike [12], this network does not fuse the lip information directly with the facial and audio information as feature embeddings into the system, instead it leverages the lip information as supplementary dynamic information for audio denoising.

5.2 Face and Voice Association for SD

As speech rhythm and word pronunciation are closely related to facial movements, a number of researchers have built their solutions for AVSD based on facial and audio information. Face-based AVSD can be consid-

ered as different tasks according to different assumptions, including assignment task and classification task. Juan et al. [15] proposed a multimodal assignment technique, which is a straightforward strategy to directly establish a correspondence between the audio and the facial features of all possible speakers in the scene. However, this assumption does not hold when there is an off-screen speaker or only background sound. Considering the assumption of the assignment task does not always hold, the researchers performed the AVSD as a classification task through evaluating the on-screen visual faces one by one. A new model based on a 3D convolutional neural networks (CNN)s and LSTMs was introduced to predict an active speaker among candidate speakers [5]. The model is based on temporal frame processing, but it does not consider temporal dependencies between frames. To address this problem, TalkNet was invented, which makes decisions by considering both short-term and long-term features [28].

In our viewpoint, vertical displacement of the face (movement of the lips and chin) is the most correlated visual feature to speech production and can be used to increase the discriminative power of audio features in SD tasks. However, detecting a speaker's lips from visual data is more challenging than detecting the face due to the smaller area it occupies. Moreover, lips are only better detected in close up images from the person front, while lip reading is difficult when subjects are not facing the camera. In contrast, face information is less correlated with verbal information though, which may lead to lower accuracy in SD task. However, the data for face information is easier to obtain, and face information can generally be obtained correctly in the case of non-frontal shots. Apart from using a single visual information, lips or face, some researchers have also tried to combine all three to get a better performance. In general, the selection of visual information is a point that needs to be carefully considered in AVSD. Furthermore, successful extraction of visual information depends on the speaker facing the camera. Nevertheless, if the speaker is back to the camera or off-screen, the model needs to be more robust and fault-tolerant. This is a challenge for future researchers.

6 Datasets for Audio-Visual Speaker Recognition

This section introduces some available datasets for AV speaker recognition based on three categories in Section 2. VoxCeleb1 [19] and VoxCeleb2 [7] are currently the most popular datasets on SI and SV. AVA-ActiveSpeaker [20], AVA-AVD Dataset [31] and VoxConverse [6] are widely used for AVSD tasks, the first two of which are primarily to solve active speaker recognition tasks. Table 1 shows the statistics of these datasets.

6.1 VoxCeleb1

VoxCeleb1 [19] is a large scale audio-visual speaker identification dataset. The dataset includes a total of 22,496 YouTube videos containing real-world noise, including background chatter, overlapping speech, laughter and recording devices, with 1,211 speakers and 148,642 utterances. Each speaker has both a video and corresponding audio. This dataset is gender balanced, with 45% of speakers being female.

6.2 VoxCeleb2

VoxCeleb2 uses a similar approach as VoxCeleb1 to collect data on YouTube, but on a larger scale, with 150,480 videos containing 5,994 speakers and 1,092,009 utterances, and is also multilingual, with speeches from 145 speakers of different nationalities, covering a range of accents, ages, ethnicities and languages [7].

Compared with VoxCeleb1, VoxCeleb2 is more than 5 times larger. Moreover, VoxCeleb2 addresses the problems of the VoxCeleb1 dataset still being an order of magnitude smaller than the popular face dataset and the lack of ethnic diversity.

6.3 AVA-ActiveSpeaker

AVA-ActiveSpeaker is the first publicly available, large-scale benchmark for the ASD task [20]. AVA-ActiveSpeaker is based on 160 videos from YouTube movies. The dataset contains about 3.65 million human labelled frames or about 38.5 hours of face tracks, and the corresponding audio. It consists of three types of labels, including Not Speaking, Speaking and Audible, as well as Speaking but Not Audible, which account for 21.8 hours, 9.46 hours and 0.35hours respectively.

Dataset	Videos	Speakers	Hours	Scenario	Language	Aim
VoxCeleb1	22,496	1,211	352	multi	En	SI, SR
VoxCeleb2	150,480	5,994	2,442	multi	multi	SI, SR
AVA-ActiveSpeaker	160	-	38.5	movies	multi	ASD (AV)
AVA-AVD Dataset	351	1,500	20.25	movies	multi	ASD (AV)
VoxConverse	448	8,268	64	debate, news	En	SV

Table 1. Datasets for AVSR systems

AV datasets are still far inferior to image datasets in quantity and variety. The reason for this may be that auditory datasets are more difficult to collect and require higher conditions than image datasets.

In terms of the AV datasets for SR, the characteristics of the datasets differ depending on the classification of recognition. Table 1 shows larger datasets for SI and SV compared to SD with more videos, diverse scenarios and languages. Regarding the datasets for SD, the datasets for ASD contain more scenes and languages, although the number of videos, speakers and duration are not as large as those for general AD.

With regard to the trend of AVSR datasets, AVSR datasets are evolving to employ automated video data collection pipelines, resulting in a larger dataset. Moreover, videos will tend to comprise multifaceted scenarios and encompass diverse forms of noise to improve model stability. Language will be multilingual, not just English.

7 Conclusion

This paper has provided a comprehensive overview of the deep learning based AV speaker recognition. The paper begins with an introduction to the classification of AVSR. With the focus on these classifications, existing AVSR systems are reviewed. The classification, strengths as well as weaknesses and challenges of these systems are explored. Prominent AV datasets are also detailed, compared and classified.

References

[1] Madina Abdrakhmanova, Saniya Abushakimova, Yerbolat Khassanov, and Huseyin Atakan Varol. A study of multimodal person verification using audio-visual-thermal data. In *The Speaker and Language Recognition Workshop (Odyssey 2021)*, 2021.

[2] Petar S. Aleksic and Aggelos K. Katsaggelos. Audio-visual biometrics. *Proceedings of the IEEE*, 94(11):2025–2044, 2006.

[3] Grigory Antipov, Nicolas Gengembre, Olivier Le Blouch, and Gaël Le Lan. Automatic quality assessment for audio-visual verification systems. the love submission to nist sre challenge 2019. In *INTERSPEECH 2020*, pages 2237 – 2241, 2020.

[4] Zhengyang Chen, Shuai Wang, and Yanmin Qian. Multi-modality matters: A performance leap on voxceleb. In *INTERSPEECH 2020*, pages 2252–2256, 2020.

[5] Joon Son Chung. Naver at activitynet challenge 2019–task b active speaker detection (ava). *ArXiv*, 2019.

[6] Joon Son Chung, Jaesung Huh, Arsha Nagrani, Triantafyllos Afouras, and Andrew Zisserman. Spot the conversation: Speaker diarisation in the wild. pages 299 – 303, 2020.

[7] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH 2018*, 2018.

[8] Joon Son Chung and Andrew Zisserman. Out of time: automated lip sync in the wild. In *ACCV Workshops*, pages 251–263, 2017.

[9] Rohan Kumar Das, Ruijie Tao, Jichen Yang, Wei Rao, Cheng Yu, and Haizhou Li. Hlt-nus submission for 2019 nist multimedia speaker recognition evaluation. *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2020)*, pages 605–609, 2020.

[10] Yifan Ding, Yong Xu, Shi-Xiong Zhang, Yahuan Cong, and Liqiang Wang. Self-supervised learning for audio-visual speaker diarization. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, pages 4367–4371, 2020.

[11] Rafizah Mohd Hanifa, Khalid Isa, and Shamsul Mohamad. A review on speaker recognition: Technology and challenges. *Computers & Electrical Engineering*, 90:107005, 2021.

[12] Mao-Kui He, Jun Du, and Chin-Hui Lee. End-to-end audio-visual neural speaker diarization. In *INTERSPEECH 2022*, page 1461 – 1465, 2022.

[13] Muhammad Mohsin Kabir, M. F. Mridha, Jungpil Shin, Israt Jahan, and Abu Quwsar Ohi. A survey of speaker recognition: Fundamental theories, recognition methods and opportunities. *IEEE Access*, 9:79236–79263, 2021.

[14] Takuhiro Kimura, Yasushi Makihara, Daigo Muramatsu, and Yasushi Yagi. Quality-dependent score-level fusion of face, gait, and the height biometrics. *Information and Media Technologies*, 9(3):346–350, 2014.

[15] Juan León-Alcázar, Fabian Caba Heilbron, Ali K. Thabet, and Bernard Ghanem. Maas: Multi-modal assignation for active speaker detection. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 265–274, 2021.

[16] Lantian Li, Ruiqi Liu, Jiawen Kang, Yue Fan, Hao Cui, Yunqi Cai, Ravichander Vipperla, Thomas Fang Zheng, and Dong Wang. Cn-celeb: multi-genre speaker recognition. *Speech Communication*, 137:77–91, 2022.

[17] David F McAllister, Robert D Rodman, Donald L Bitzer, and Andrew S Freeman. Lip synchronization of speech. In *Audio-Visual Speech Processing: Computational & Cognitive Science Approaches*, 1997.

[18] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. VoxCeleb: A Large-Scale Speaker Identification Dataset. In *INTERSPEECH 2017*, pages 2616–2620, 2017.

[19] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: A large-scale speaker identification dataset. In *INTERSPEECH 2017*, 2017.

[20] Joseph Roth, Sourish Chaudhuri, Ondrej Klejch, Radhika Marvin, Andrew Gallagher, Liat Kaver, Sharadh Ramaswamy, Arkadiusz Stopczynski, Cordelia Schmid, Zhonghua Xi, and Caroline Pantofaru. Ava active speaker: An audio-visual dataset for active speaker detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, pages 4492–4496, 2020.

[21] Seyed Omid Sadjadi, Craig Greenberg, Elliot Singer, Lisa Mason, and Douglas Reynolds. The 2021 nist speaker recognition evaluation. *ArXiv*, 2022.

[22] Seyed Omid Sadjadi, Craig S Greenberg, Elliot Singer, Douglas A Reynolds, Lisa P Mason, Jaime Hernandez-Cordero, et al. The 2019 nist audio-visual speaker recognition evaluation. In *The Speaker and Language Recognition Workshop (Odyssey 2020)*, pages 259–265, 2020.

[23] Leda Sarı, Kritika Singh, Jiatong Zhou, Lorenzo Torresani, Nayan Singhal, and Yatharth Saraf. A multi-view approach to audio-visual speaker verification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*, pages 6194–6198, 2021.

[24] Milton Sarria-Paja, Mohammed Senoussaoui, Douglas O'Shaughnessy, and Tiago H Falk. Feature mapping, score-, and feature-level fusion for improved normal and whispered speech speaker verification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*, pages 5480–5484, 2016.

[25] Saqlain Hussain Shah, Muhammad Saad Saeed, Shah Nawaz, and Muhammad Haroon Yousaf. Speaker recognition in realistic scenario using multimodal data. *ArXiv*, 2023.

[26] Suwon Shon, Tae-Hyun Oh, and James Glass. Noise-tolerant audio-visual online person verification using an attention-based neural network fusion. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, pages 3995–3999, 2019.

[27] Ruijie Tao, Rohan Kumar Das, and Haizhou Li. Audio-visual speaker recognition with a cross-modal discriminative network. In *INTERSPEECH 2020*, 2020.

[28] Ruijie Tao, Zexu Pan, Rohan Kumar Das, Xinyuan Qian, Mike Zheng Shou, and Haizhou Li. Is someone speaking? exploring long-term temporal features for audio-visual active speaker detection. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3927–3935, 2021.

[29] Jesús Villalba, Bengt J Borgstrom, Saurabh Kataria, Magdalena Rybicka, Carlos D Castillo, Jaejin Cho, L Paola Garcıa-Perera, Pedro A Torres-Carrasquillo, and Najim Dehak. Advances in cross-lingual and cross-source audio-visual speaker recognition: The jhu-mit system for nist sre21. In *The Speaker and Language Recognition Workshop (Odyssey 2022)*, pages 213–220, 2022.

[30] Abudukelimu Wuerkaixi, Kunda Yan, You Zhang, Zhiyao Duan, and Changshui Zhang. Dyvise: Dynamic vision-guided speaker embedding for audio-visual speaker diarization. In *IEEE 24th International Workshop on Multimedia Signal Processing (MMSP 2022)*, pages 1–6, 2022.

[31] Eric Z. Xu, Zeyang Song, Chao Feng, Mang Ye, and Mike Zheng Shou. Ava-avd: Audio-visual speaker diarization in the wild. *Proceedings of the 30th ACM International Conference on Multimedia*, 2021.

[32] Hani Yehia, Philip Rubin, and Eric Vatikiotis-Bateson. Quantitative association of vocal-tract and facial behavior. *Speech Communication*, 26(1-2):23–43, 1998.

Service Mesh Technical Details

Jinjia Zhang
jinjia.zhang@aalto.fi

Tutor: Tuomas Aura

Abstract

Moving from monolithic approaches to microservice-based architectures has been increasingly common in cloud-based software development and delivery. Service meshes are the latest infrastructure development to support microservices even better. It provides many significant features for microservices infrastructures, including routing, resilience, and monitoring. This paper reviews the state-of-the-art service mesh implementation Istio and conducts experiments on how the service mesh impacts application performance. Finally, it discusses future opportunities in service mesh technology.

KEYWORDS: Service Mesh, Istio, Kubernetes

1 Introduction

The microservices software architecture has gained considerable popularity by enabling rapid, reliable delivery of complex applications. This architecture separates an application into multiple small and lightweight services, which are independently deployable and loosely coupled [1]. However, the main disadvantage of microservices is that the integration of microservice systems faces particular difficulties due to the complexity

and dynamism of the interfaces and communication patterns between the services [19]. Therefore, a promising approach, known as *service mesh*, has been developed and widely adopted in the industry. A service mesh is a dedicated infrastructure component that facilitates traffic management observing and securing communication between services without code changes to the service implementations. It only focuses on service-to-service communication rather than on business concerns.

Many open-source service mesh solutions exist, including Istio [10], Linkerd [12], and Cilium Mesh [2]. This paper aims to review the key technologies behind the service mesh based on the Istio implementation, evaluate its performance impact through experiments and discuss the future directions of service mesh technology.

The rest of this paper is organized as follows. Section 2 elaborates on the definition of Kubernetes as well as the service mesh. Section 3 analyzes the Istio architecture while Section 4 conducts experiments and discusses the results. Section 5 surveys some future possibilities of service mesh. Finally, Section 6 concludes the paper.

2 Background

This section overviews Kubernetes and the service mesh concept.

2.1 Kubernetes

Kubernetes is an open-source container orchestration framework for automatically managing containerized applications through simple configurations [11]. It manages multiple distributed machines as a cluster, and each working machine is referred to as a node. The commonly used concepts in the framework are as follows:

- *Pods* are the smallest management and deployment unit in Kubernetes. A pod might contain multiple containers sharing the same network stack and storage resources. One pod typically implements one microservice instance in the microservice architecture.

- *Deployments* control the behavior of pods by defining the desired number and types of pods in the system. The deployment can be scaled up or down by changing the number of replicas of the pods in it.

- *Services* define a virtual static IP address that can be accessed by other cluster services or outside services. When pods communicate with dynamic IP addresses, pods have to restart and connect new IP addresses since they are ephemeral. In addition, a solution is needed for discovering the pods and balancing the load among them.

2.2 Service Mesh

A service mesh is a dedicated infrastructure component to improve the convenience of microservice development and to reduce its operational complexity. The service mesh layer is located between the application layer and the orchestration layer. It consists of the data plane and the control plane. The data plane includes a series of proxies, also known as sidecars. The sidecar function is to proxy ingress and egress (i.e., inbound and outbound) traffic to and from a microservice and to collect as well as report mesh telemetry data. The control plane manages the sidecars through configuration and provides a visual interface to view the traffic of the whole system. In particular, the service mesh layer consists of four principal features, as follows:

- *Traffic management* ensures the connectivity of the various services and provides functions to ensure high availability and resiliency, including fault injection and circuit breaking. It allows fine-grained control of the mesh traffic.

- *Observability* provides data and metrics on all service behaviors. Furthermore, it improves the developers' troubleshooting capabilities by integrating visualization tools.

- *Security* protects communication that traverses the physical network between pods by encryption and authentication. Additionally, it isolates the microservices and their communication from each other on the cluster network. Imposing access control policies prevents unauthorized access. Therefore, security aims to mitigate some threats. For example, attackers can compromise microservices within the service mesh, gaining access to the service mesh components and other microservices.

- *Extensibility* improves the flexibility of the service mesh and facilitates

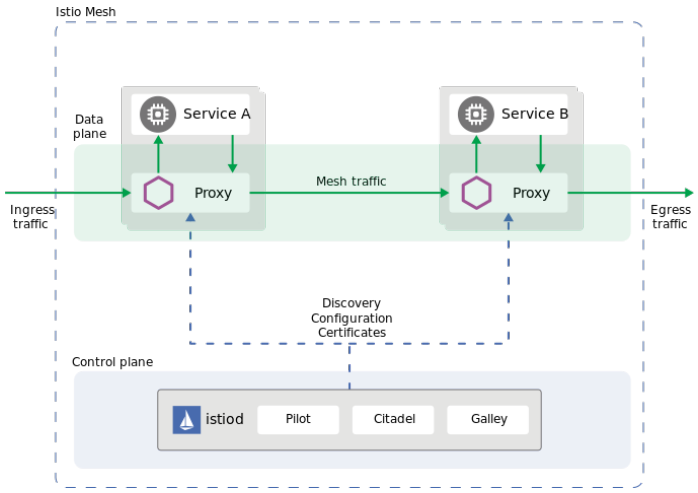


Figure 1. Service mesh architecture adapted from [10]

the expansion of its capabilities.

3 Example: Istio

As mentioned before, Istio is an open-source implementation of the service mesh, and it was released to the public in 2017 with the support of Google, Lyft, and IBM [14].

3.1 Architecture

The architecture of Istio follows the core principle of the service mesh, which involves separating the control plane from the data plane. Istio 1.5 changed the architecture of the control plane from microservices to a monolith, which greatly simplifies the service mesh management for administrators. The pod name is istiod that consists of Mixer, Citadel, Galley, and Pilot. In terms of the data plane, each pod comprises two containers. One is the microservice, and the other is a proxy, which acts as an agent to intercept inbound and outbound traffic of the microservice. In Istio, Envoy [3], a high-performance proxy implemented in C++, is the only supported sidecar. The latest Istio 1.17 architecture is shown in Figure 1.

3.2 Traffic Management

In traditional microservice application development, the microservices establish connections, such as HTTP connections, through domain names or IP addresses to communicate with each other. In service mesh applica-

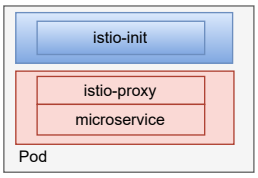


Figure 2. Pod architecture

tion development, the application code and development process require no changes because the sidecars are transparently added to the microservices. The sidecars will intercept the incoming requests and redirect them to the microservices.

Istio provides three methods to inject a sidecar into a pod: automatic injection, manual injection, and custom injection. The automatic injection is mainly analyzed here. Istio takes advantage of Kubernetes mutating webhook to complete this task. Mutating webhook allows requests to invoke the webhook server before reaching the Kubernetes API server. The webhook server attaches two additional containers, istio-init and istio-proxy, to the pod. The pod architecture is shown in Figure 2. All containers in the same pod share a network stack. The istio-init container is responsible for initializing the network environment of the pod. In addition, it configures iptables rules, enabling sidecars to intercept inbound and outbound traffic. The container runs as a privileged user and exits after performing these tasks. The istio-proxy container executes two processes. The first one is the pilot-agent, and the second one is the Envoy proxy bootstrapped by the pilot-agent. The pilot-agent process receives control plane instructions and Envoy-related configurations as an xDS server. The xDS refers to a series of discovery service resources. For example, endpoint discovery service (EDS) provides service addresses.

Additionally, Istio offers two methods to intercept traffic. It is based on the istio-init container or the Istio Container Network Interface (CNI) plugin. Both of them complete the traffic interception by writing iptables rules. However, the istio-init container needs the *NET_ADMIN* and *NET_RAW* Linux capabilities, which is the default method. The Istio CNI plugin implements the same networking functionality but without requiring privileges. It is added to the existing CNI plugin configuration and only writes iptables rules in the Kubernetes pod lifecycle's network setup phase.

Figure 3 analyzes how Istio traffic is routed in the two scenarios. The numbers in the figure indicate the order of the processing steps. Appendix

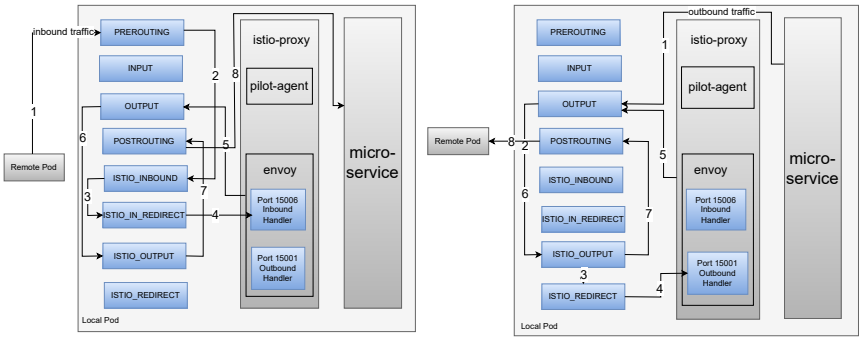


Figure 3. Traffic flow adapted from [18]

1 shows detailed iptables rules.

3.3 Observability

Istio generates three types of data to provide observability of the service mesh. The first data type is metrics such as latency and response times for requests. These metrics are organized into three classes, namely proxy level, service level, and control-plane level. Prometheus is the database that collects metrics by default and scrapes metrics from the specific endpoint /**metrics** at intervals [17]. Specifically, the control-plane level metrics are collected on port 15014 by the Istio control plane (i.e., the istiod pod). and the proxy level and service level metrics are collected on port 15020 of the sidecar.

The second data type is distributed traces, which provide a method to monitor the dependencies of the services and interactions between different services. Istio leverages Envoy's distributed tracing feature to implement the tracing integration. Although the sidecars automatically generate trace spans on behalf of the applications they proxy, the applications need to forward the appropriate request context to ensure that the spans in the same request are in a single trace. For example, application developers must collect some HTTP headers from each incoming request and forward these headers to the next application.

The third data type is access logs, which provide an approach to record the behavior of services, including request source and destination metadata. Microservices output log data to a file in the pod to help debug applications or sends data to an OpenTelemetry Collector, which can receive, process, and export telemetry data [16].

3.4 Security

Istio ensures the security of the service mesh by a Certificate Authority (CA), authentication policies, and authorization policies. It securely provisions strong identities to every pod with X.509 certificates. In Kubernetes, the ServiceAccount resource serves as the Istio identity model. Each pod is assigned a service account to run. By default, Istio provides certificate management out of the box. When Istio starts, it will create a self-signed certificate for the istiod pod as the root CA. However, Istio can also integrate an external CA, thus securing the communication in multiple service meshes with the same trusted root CA. When a pod starts, pilot-agent running in the data plane creates a key pair and certificate signing request (CSR), and sends the CSR to the istiod pod. The CA verifies the credentials in the CSR. If the validation succeeds, the CA generates a certificate and returns it to the pilot-agent, which sends the certificate and private key to the Envoy via the Envoy secret discovery service (SDS) API. After this, the sidecar proxy has credentials to communicate with the control plane and other proxies.

Peer authentication and request authentication are provided by Istio. Peer authentication is used for service-to-service communication and supports three modes: permissive mode, in which a service accepts both plaintext traffic and mutual TLS traffic at the same time, strict mTLS mode, and disabled mTLS mode. Request authentication adopts JSON Web Token (JWT) validation for end-user authentication. Furthermore, an AuthorizationPolicy resource is introduced to define authorization policies, which offers mesh-wide, namespace-wide, and workload-wide access control. The authorization policies impose access control to the inbound traffic in the server-side Envoy sidecar based on the runtime authorization engine running in the Envoy process, which verifies the request context against the current authorization policies and returns the authorization result, either ALLOW or DENY. In Kubernetes, the access control is based on the labels to restrict policies to apply to specific workloads and the service accounts to restrict policies to apply to specific identities.

Istio can be extended to support joining virtual machines into the Kubernetes mesh. Various infrastructures identify workloads differently, for example, identifying a workload by its IP address and port in virtual machines or by service names in Kubernetes. Therefore, in a multi-cloud environment, it is important to have a unified identity authentication

standard. Istio 1.14 supports the Secure Production Identity Framework for Everyone (SPIFFE) standard by enabling SPIFFE Runtime Environment (SPIRE), which is a production-ready implementation to identify workloads in dynamic and heterogeneous environments. The heart of the SPIFFE specification is defining short-lived cryptographic identity documents, also known as SPIFFE Verifiable Identity Documents (SVIDs) by simple APIs. SPIRE consists of the SPIRE server and the SPIRE agent. The server is responsible for issuing SVIDs and registering workloads. The agent running on every node requests SVIDs from the server and offers Workload APIs the identity documents in the X.509 or JWT format. The SPIRE agent communicates with pods on the same node through the shared UNIX Domain Socket (UDS) and supports the Envoy SDS APIs, i.e., each Envoy sidecar can fetch the SVID through the SDS API.

3.5 Extensibility

Istio utilizes the Envoy WebAssembly to provide extensibility. WebAssembly is a portable binary instruction format for a stack-based virtual machine [7]. In previous versions of Envoy, extending its functions (e.g., adding custom filters) required a specific language, C++, and these programs needed to be compiled into the Envoy binary, i.e., the developers were responsible for maintaining a fork of Envoy [13]. After the introduction of WebAssembly, extensions can be loaded into Envoy dynamically at runtime.

4 Experiments And Results

We set up the service mesh cluster using Minikube in one node [15]. It was allocated 2 CPUs and 2 GB of memory. We utilized Istio's load testing tool Fortio [5]. Fortio can run at a specified number of queries per second (QPS) and calculates percentiles for the response times. Our test scenario is to simulate 10 users concurrently requesting the Fortio server for 20 seconds with or without sidecar proxies in one cluster. The files used for this experiment are shared on our GitHub page[1] for community access. The performance impact of introducing sidecars is the focus of this experiment. The experimental results are shown in Figure 4. The 99th percentile is a crucial metric to know microservices latencies [6].

According to the experimental results, as the expected QPS increases,

[1]https://github.com/jinjiaKarl/Istio-experiment

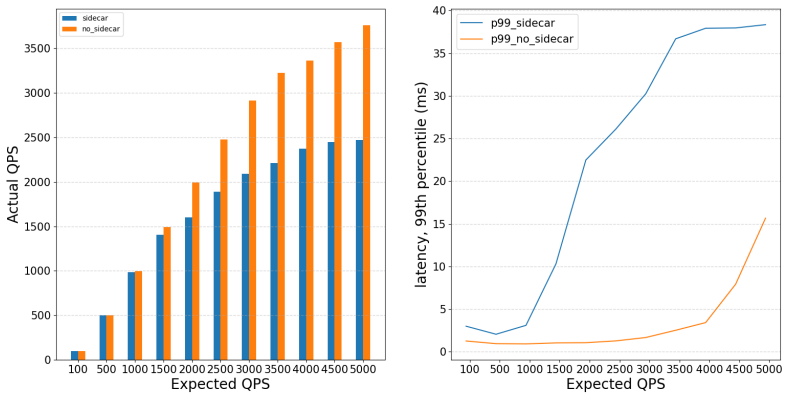


Figure 4. Experiments

the actual QPS fastly reaches the maximum QPS with sidecars and the maximum QPS with sidecars is lower than without sidecars. The latencies increase faster with sidecars than without sidecars. The introduction of sidecar proxies causes some performance degradation because the traffic between services transits the proxies' network protocol stack, resulting in an increased delay. Furthermore, Istio collects some additional telemetry data, such as, tracing data. The more services a user's request passes, the more obvious the application performance impact will be.

5 Future Directions

The sidecar model requires a large number of proxies, many additional network connections, and a complicated redirection logic to feed network traffic into the proxies, which leads to performance degradation. Therefore, the sidecar-less model has attracted the attention of the community. Cilium Service Mesh, which utilized Extended Berkeley Packet Filter (eBPF), is the advocate of this architecture. eBPF programs run directly on the Linux kernel instead of calling syscalls and passing packets back and forth between kernel space and user space. However, it is difficult to write and debug eBPF programs, and the model has many limitations. For example, the size of the program is limited. Although eBPF powers forwarding, filtering, and monitoring of IP layer (L3) and transport layer (L4) packets, it is difficult to process application layer (L7) data such as HTTP retries. Therefore, Cilium Service Mesh currently adopts the following method. If eBPF programs are not capable of processing the request, the Envoy proxy running per node handles the request. In a multi-tenant

data center with shared hosts, Envoy needs to be multi-tenant aware since it handles all connections in this node. Although this architecture reduces the number of proxies and network delays, it also causes other problems. First, the failure of the proxy has a wider impact on the service mesh. Second, all traffic on the host is handled by the same proxy. If one application has extremely high traffic and consumes all of the proxy's resources, the other applications on the host risk being starved. Therefore, choosing the sidecar model or eBPF model is similar to choosing virtual machines or containers. The sidecar model provides stricter isolation, whereas the eBPF-based model can share resources more effectively.

Currently, several directions in the service mesh deployment model have been developing. The first direction is to improve the ability of eBPF in managing L7 traffic in order to substitute sidecar proxies. The second direction is the introduction of the Proxyless Service Mesh based on Google Remote Procedure Call (gRPC) [8]. The development team only needs to maintain gRPC versions in different languages, and the service mesh has no sidecar injections. The third direction is Istio launched sidecarless Ambient Mesh [9], which splits Istio's functionality into two distinct layers: the L7 processing layer and the secure overlay layer. When the complicated L7 functions are not required, it is not necessary to deploy sidecars. Finally, the fourth direction is to persist in the sidecar model and continuously optimize the performance of the control plane and data plane.

In terms of security, Role-based access control (RBAC) is used by default in Kubernetes for access control. It results in some problems such as the explosion of the number of roles. Therefore, Next Generation Access Control (NGAC) is proposed as a service mesh access control with fine-grained granting or denying of user management capabilities [4].

6 Conclusion

This paper has reviewed the key technologies implemented by service meshes, including traffic management, observability, security, and extensibility. We used Istio as the example of a state-of-the-art service mesh. Our experimental results show that the sidecar injection causes some performance degradation due to more network hops and runtime overhead. Finally, we discuss the future directions of service mesh.

References

[1] Washington Henrique Carvalho Almeida, Luciano de Aguiar Monteiro, Raphael Rodrigues Hazin, Anderson Cavalcanti de Lima, and Felipe Silva Ferraz. Survey on microservice architecture-security, privacy and standardization on cloud computing environment. *ICSEA 2017*, page 210, 2017.

[2] Cilium. Online. https://cilium.io/. Accessed: 2023-01-29.

[3] Envoy. Online. https://www.envoyproxy.io/. Accessed: 2023-01-29.

[4] David Ferraiolo, Ramaswamy Chandramouli, Rick Kuhn, and Vincent Hu. Extensible access control markup language (XACML) and next generation access control (NGAC). In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, ABAC '16, page 13–24, New York, NY, USA, 2016. Association for Computing Machinery.

[5] Fortio. Online. https://fortio.org/. Accessed: 2023-01-29.

[6] Mrittika Ganguli, Sunku Ranganath, Subhiksha Ravisundar, Abhirupa Layek, Dakshina Ilangovan, and Edwin Verplanke. Challenges and opportunities in performance benchmarking of service mesh for the edge. In *2021 IEEE international conference on edge computing (EDGE)*, pages 78–85. IEEE, 2021.

[7] Andreas Haas, Andreas Rossberg, Derek L Schuff, Ben L Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. Bringing the web up to speed with webassembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 185–200, 2017.

[8] Isito. Online. https://istio.io/latest/blog/2021/proxyless-grpc/. Accessed: 2023-01-29.

[9] Isito. Online. https://istio.io/latest/blog/2022/introducing-ambient-mesh/. Accessed: 2023-01-29.

[10] Istio. Online. https://istio.io/latest/. Accessed: 2023-01-29.

[11] Kubernetes. Online. https://kubernetes.io/docs/home/. Accessed: 2023-01-29.

[12] Linkerd. Online. https://linkerd.io/. Accessed: 2023-01-29.

[13] Antonio Lioy, Dott Ignazio Pedone, and Matteo Pace. Zero trust networks with istio. *Ph. D. dissertation*, 2021.

[14] Nabor C Mendonça, Craig Box, Costin Manolache, and Louis Ryan. The monolith strikes back: Why Istio migrated from microservices to a monolithic architecture. *IEEE Software*, 38(05):17–22, 2021.

[15] Minikube. [online]. https://minikube.sigs.k8s.io/docs/. Accessed: 2023-01-29.

[16] Opentelemetry. Online. https://opentelemetry.io/docs/concepts/components/. Accessed: 2023-01-29.

[17] Prometheus. Online. https://prometheus.io/. Accessed: 2023-01-29.

[18] Jimmy Song. Online. https://jimmysong.io/en/blog/istio-sidecar-traffic-types/. Accessed: 2023-01-29.

[19] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Wenhai Li, Chao Ji, and Dan Ding. Delta debugging microservice systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 802–807, 2018.

1 Appendix: Iptables Rules

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
ISTIO_INBOUND  tcp  --  anywhere             anywhere

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

target     prot opt source               destination
ISTIO_OUTPUT  tcp  --  anywhere            anywhere

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain ISTIO_INBOUND (1 references)
target     prot opt source               destination
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15008
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15090
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15021
RETURN     tcp  --  anywhere             anywhere             tcp dpt:15020
ISTIO_IN_REDIRECT  tcp  --  anywhere             anywhere

Chain ISTIO_IN_REDIRECT (3 references)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             redir ports 15006

Chain ISTIO_OUTPUT (1 references)
target     prot opt source               destination
RETURN     all  --  127.0.0.6            anywhere
ISTIO_IN_REDIRECT  all  --  anywhere  !localhost   owner UID match 1337
```

```
RETURN     all  --  anywhere             anywhere   ! owner UID match 1337
RETURN     all  --  anywhere             anywhere      owner UID match 1337
ISTIO_IN_REDIRECT  all  --  anywhere     !localhost  owner GID match 1337
RETURN     all  --  anywhere             anywhere    ! owner GID match 1337
RETURN     all  --  anywhere             anywhere      owner GID match 1337
RETURN     all  --  anywhere             localhost
ISTIO_REDIRECT  all  --  anywhere             anywhere

Chain ISTIO_REDIRECT (1 references)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere        redir ports 15001
```

Machine learning for fog and edge service placement

Kwan Li
kwan.li@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

In recent years, technology is evolving in a rapid manner. In order to provide the best user experience, edge and fog computing are often used in developing devices and applications. However, the environment for the devices can be highly dynamic. As a result, optimal time and location placement of fog services has been an issue that needs to be solved. This study examines the utility of reinforcement learning in solving fog/edge service placement problems by examining various reinforcement learning algorithms that were proposed.

KEYWORDS: Reinforcement Learning, Deep Reinforcement Learning, Markov Decision Process, service placement, fog computing, edge computing

1 Introduction

With the constant evolution of technology, many new applications and devices are constantly developed. These applications and devices often requires minimal latency to provide an excellent user experience. Some of these latency issues can be solved with the services provided by fog and edge computing. Yu et al. [1] describes edge computing as a paradigm that is used to migrate storage and computation of data to the edge of the

network where it is closest to the end-users. This provides benefits such as lower transmission latency between servers and application response time improvements. On the other hand, fog computing can be seen as a computing model that provides services such as networking and control in addition to the services provided by edge computing [2]. However, despite the benefits of fog and edge computing, there are challenges regarding both computing models. The key issue being finding a way to determine the optimal time and location to place to the services of fog and edge computing. There are multiple methods proposed to solve this problem. One of the approaches is to use reinforcement learning to accurately determine the most optimized configurations for the fog and edge service placement. This can be done with machine learning techniques such as reinforcement learning (RL) and supervised learning (SL). This study will mainly focus on the reinforcement learning approach.

2 Background

This chapter will introduce the fundamentals of fog and edge computing, as well as the challenges of service placement of the two computing models. Additionally, the basics of reinforcement learning and the applications of it in fog and edge computing will be explained.

2.1 Fog and Edge computing

As a relatively new technology, the definition of fog computing may vary drastically depending on the source. Many associates fog computing as a way to provide services such as computing, storage and networking to nodes in the network that is closer to the end-users [2]. However, as stated by Bonomi et al. [3], the services provided by fog computing are not necessarily only placed on the edges of the network and instead could be positioned across the network.

Unlike fog computing, the services provided by edge computing such as data computation and storages are only provided on the edge of the network that is closest to the end-users. This allows edge computing to handle common performance issues such as latency and connectivity much better than the other computing models [2]. This does not imply that the other computing models are to be replaced by edge computing. Instead, according to Cao et al. [4], edge computing should be implemented in a

way that compliments other computing models.

2.2 Service placement

In order for the two previous computing models to work efficiently, one major concern needs to be addressed. “How to optimize the deployment of services?” This service placement problem can be quite challenging, as unlike cloud computing services, the fog and edge services are resource-constrained and highly dynamic [5]. Additionally, as opposed to the cloud computing, the services provided by fog and edge are generally spread out in the network which further complicates the service placement problem as there may be geographical complications that must be taken into account as well. However, if the service problems are addressed and optimized properly, fog and edge computing provides great benefits to the end-users.

2.3 Reinforcement learning

Reinforcement learning (RL) is a subarea of machine learning, where an agent is placed in an environment for it to perform actions independently, in hopes to finding the best actions to reach a specified goal [6]. The environment are often characterized by few factors. First, a state space is defined, which is a set of states that typically depicts locations in which actions can be taken. These actions also form another set called action space. The goal of the agent is then defined as a policy which maps the states to actions that would reach the specified goal. In order to find the policy, each action is assigned a reward and the agent is then required to perform the actions to find a goal while also maximizing the cumulated reward. This is a lengthy process as the agent must explore multiple different sequences of actions in order to optimize the reward obtained. Another key aspect of reinforcement learning is that it differs from other subareas of machine learning by not requiring data sets to be provided to train the algorithm. This allows reinforcement learning a better ability to explore more freely with less bias towards the data set used to train [7].

3 Reinforcement learning for service placement

Service placement problems are generally very complicated and would require significant amount of effort only to find a sub optimal solution.

Therefore, reinforcement learning is used as an approach in hopes to produce better solutions more efficiently.

In order to utilize reinforcement learning, the service placement problem is formulated as a Markov Decision process (MDP). The MDP is generally characterized as tuples with different designs depending on the author. Zhang et al. [8] formulated their MDP as a 4-tuple (S, A, P, R), where S is defined as the finite state space and A as the finite action space. P is the state transition strategy and R represents the reward function associated with the mappings of the states and actions. On the other hand, Eyckerman et al. [9] also formulated their MDP as Multi-Objective Markov Decision Process (MOMDP) with similar values but with few additions. Their MOMDP is represented as 6-tuple ($S, A, P, R, \omega, f_\Omega$), where ω is the space of preference and f_Ω is the preference function that takes preference ω as input and outputs the scalar utility for each action in a given state [9].

In addition to formulating the MDP, there are other factors that must be taken into consideration when developing a reinforcement learning algorithm to solve service placement problems. This chapter will first explore the different optimizations that forms the service placement problem. Afterwards, the training process of RL algorithms for service placement is discussed. Lastly, the relationship of deep learning with reinforcement learning techniques and the usage of deep reinforcement learning in service placement problems is examined thoroughly.

3.1 Defining the optimization

One of the core aspect of the service placement problem is to define the value to optimize. Poltronieri et al. [6] explores the applications of deep reinforcement learning by optimizing value of information (VoI) for service placement. VoI is a form of utility that information objects (IO) brings to end-users [10]. Using VoI as the resource management criterion, it can be optimized to produce a resource management system that prioritizes assigning resources to the services that provides the highest amount of value to end-users. Consequently, the results of optimizing VoI shows that it is an effective method for addressing resource management issues of fog computing.

On the other hand, Liu et al. [11] prioritizes on minimizing total latency of tasks in a long term. This is achieved by optimizing the service placement and allocation. As a consequence, they hope to solve the issue

of low-latency requirements of computation-intensive applications in 5G environments. Similarly, Eyckerman et al. [9] aims to solve the service placement problem of optimizing energy efficiency while also minimizing the total impact on the network. As a result, they have proposed a Multi-Objective Reinforcement Learning (MORL) model to solve these conflicting objectives.

3.2 Training the reinforcement learning model

As mentioned in chapter 2, RL algorithms does not require any data sets to train the algorithm to produce results. With reinforcement learning, there are other approaches which can be used to train the algorithms. For example, Poltronieri et al. [6] proposed an algorithm FogReinForce which is trained using a simulator. The simulator reenacts the properties of fog services which interacts with the algorithm through a HTTP REST interface. The algorithm is then trained using the simulator in 1000 episodes of iterations in order to find the optimal policy. On the other hand, Mohammadi et al. [12] explores the possibilities of training the algorithm with data sets while still utilizing the deep reinforcement learning model. Both of the model training methods produced results indicating the benefits of the algorithms. However there is still an issue regarding the training time of the RL algorithms. The training process for RL models generally requires huge time investment, which is one of the key concerns of developing a RL algorithm. Figure 1 describes an example of the training process of RL model using simulation. An algorithm is executed to interact with a simulation. The simulation then returns a certain amount of reward based on the action taken to the algorithm. This process is repeated for a finite amount of times, and the RL model is then trained based on the results of this training component.

3.3 Utilizing deep learning with reinforcement learning

Deep learning (DL) is a subsection of machine learning techniques which suited to solve high dimensional problems. Additionally, by making use of deep neural networks (DNN), deep learning can model high-level abstractions in data [13]. This property of deep neural network can also be used in reinforcement learning. Li et al. [14] defines deep reinforcement learning (DRL) as methods which utilizes deep neural networks to approximate the values of reinforcement learning components, such as the

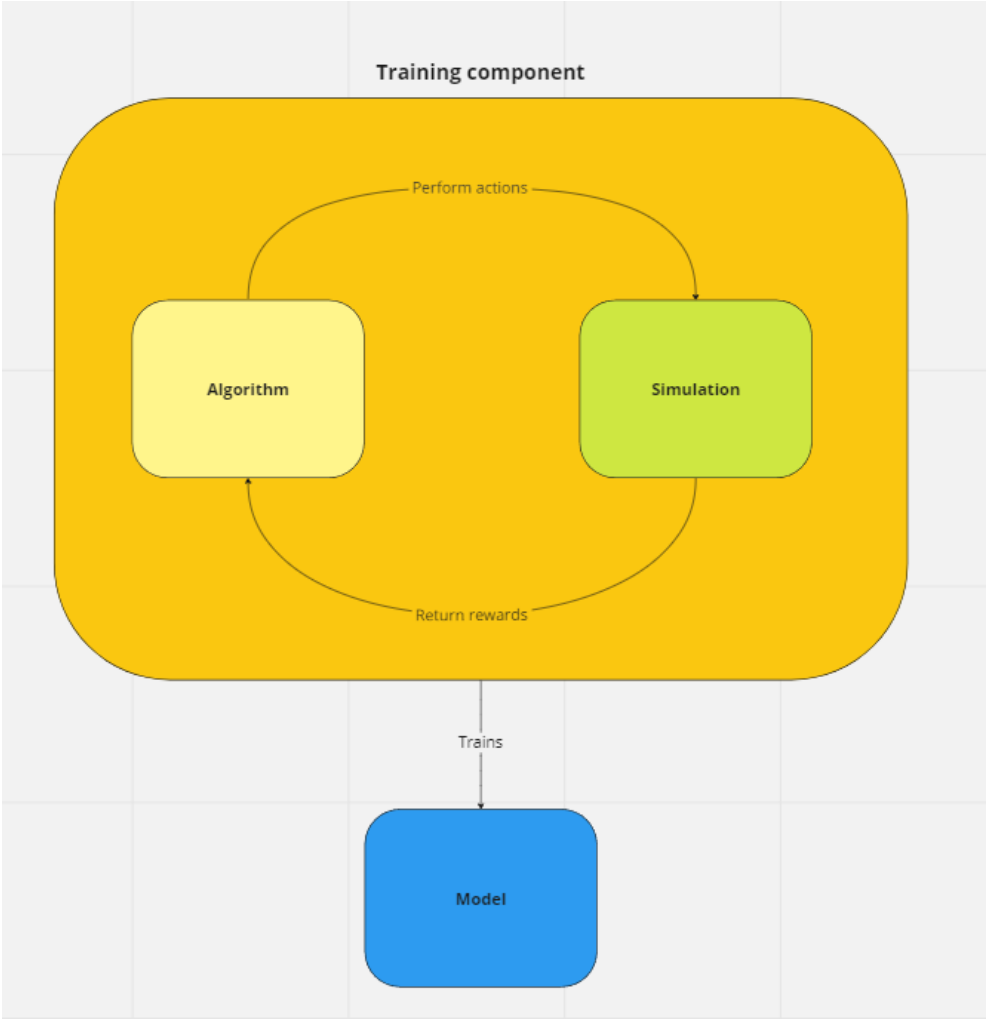


Figure 1. An example of the training process for a model using simulation.

policy.

Deep reinforcement learning has been integrated in many of approaches to solve the service placement problem. Poltronieri et al. [6] employs a DLR based algorithm FogReinForce, to solve the VoI optimization problem. Specifically, the Deep Q-Network (DQN) algorithm was used to implement the learning process for the FogReinForce algorithm. The algorithm proved to be able to find a good-rewarding allocation policy that optimizes VoI efficiently. This was evaluated by using a realistic fog computing scenario, which represents a city where the citizens exploits the functionalities of different Fog services.

DLR was also employed by Liu et al. [11] in their approach to solve the online service placement and computation resource allocation problem. They proposed an algorithm that is similar to the Deep Q-Network that was used in the FogReinForce, namely the Parameterized Deep Q Network (PDQN). PDQN is an algorithm which is used to evaluate the values

of service placement and computation resource allocation decisions. The evaluation of the decisions is mainly handled by the two deep neural networks that were used in the algorithm. As opposed to the DQN algorithm, PDQN is capable of handing discrete-continuous hybrid action spaces effectively, which was one of the reasons for adopting it in the approach to solve the service placement problem. The performance of the PDQN algorithm is evaluated using different simulation setups with various service placement approaches, such as DQN-based approach. The results of the evaluation indicate that PDQN performs better than other service placement approaches by having significantly lower total latency [11].

Similarly, Eyckerman et al.[9] explored the DQN methodologies to solve their Multi-Objective Optimization (MOO) problem. In their studies, Eyckerman et al. [9] examined two different Multi-Objective Reinforcement Learning algorithms, which were the Deep Optimistic Linear Support (Deep-OLS) and the Conditioned Network. Deep-OLS is an algorithm proposed by Mossalem et al. [15] that utilizes two Deep Neural Networks for optimization. Conditioned network (CN) on the other hand is an algorithm proposed by Abels et al. [16] which is trained based on the DQN algorithm. These algorithms were further expanded by Eyckerman et al. [9] using Double Dueling DQN. Lastly, the algorithms are then evaluated by comparing their performance to other algorithms, such as Random Search and Non-dominated Sorting Genetic Algorithm II (NSGA-II) that previously proposed by Eyckerman et al. [9]. While the results show that all of the algorithms were able to find solutions that fulfills the goal, the MORL algorithms were shown to be more efficient by requiring less execution time. Specifically, the CN algorithm is able to produce similar results as the NSGA-II while requiring much less execution time.

The previously mentioned approaches to solve the service placement problem all uses deep reinforcement learning in their solution. This is due to dimensions of the state and actions spaces growing exponentially [6, 11]. As a consequence, the traditional model free RL methods, such as Q-learning are not as suitable for the problem [11]. On the other hand, Zhang et al. [8] proposed the Q-placement algorithm, which is a model free reinforcement learning technique. By utilizing Q-placement, they have managed to achieve service cost reductions by optimizing the service placement. The results were evaluated by comparing Q-placement to other state-of-art service/data placement algorithms such as Least recent used (LCU). Based on the results, the service cost reductions provided

by Q-placement were shown to be more significant compared to the other service/data placement algorithms.

Comparision between Reinforcement Learning algorithms			
Algorithm	FogReinForce	Parameterized Deep Q Network	Conditioned Network
MDP model	(S, A, P, R, π)	(S, A, P, R, γ)	$(S, A, P, R, \omega, f_\Omega)$
Optimization	Value of information	Latency	Energy efficiency, total impact on network
Training method	Simulation	Simulation	Simulation
DRL usage	Yes	Yes	Yes

Table 1. Comparison of different reinforcement learning algorithms that were discussed in this study.

4 Discussion

The various reinforcement learning models that were examined in this study all had a different strategy on solving the service placement problem. Despite the different approaches to solve the problem, there were many similarities between them. The similarities can be seen in Table 1, which compares the different reinforcement learning algorithms and their properties. The three main RL algorithm that were discussed in this study all utilized deep reinforcement learning in their approaches. Furthermore, the reinforcement learning models were trained using a simulation. However, there are still key differences between the RL algorithms, namely how the Markov Decision Process is defined and the value to optimize.

Defining the value to optimize is one of the core aspects of designing a RL algorithm, as value formulates the optimization problem that the algorithm aims to solve. The different approaches in this study all chose

a different value to be optimized. The FogReinForce prioritizes on optimizing VoI, PDQN algorithm focuses on minimizing the total latency and Conditioned Network was used to improve energy efficiency while minimizing the total impact on the network. The difference between the optimizations has a huge influence on the techniques and technologies used on solve the service placement problem. For example, in the case of optimizing energy efficiency and total impact on the network, Eyckerman et al. [9] had to specifically formulate an extension of MDP to propose a solution to the problem. On the other hand, FogReinForce had to use a specific simulator to train model to evaluate VoI service components.

For the FogReinForce algorithm, the MDP was defined as the following tuple: (S, A, P, R, π). This closely resembles a typical MDP model defined by Zhang et al. [8] with the addition of π which describes the optimal policy that would maximize the amount of reward for each action taken. On the other hand, the MDP model of PDQN algorithm differentiates from the MDP model defined in FogReinForce. Instead of having the optimal policy as one of the 5 elements of the tuple, γ was used to describe a discount factor which is used to indicate the degree of future rewards looked into. In contrast, the MDP model defined in Conditioned Network has another additional element in the tuple. This is to be expected as Eyckerman et al. [9] defined it as Multi-Objective MDP, which an extension of the traditional MDP model, in order to solve the Multi-Objective Optimization problem. It is also important to note that while all of the MDP models shares the same reward function element R, the function itself is different.

In addition to the algorithms discussed in this study, there are also other algorithms that can be used for solving issues of fog computing. For example, Heuristic algorithms are often simple to implement while also providing a decent solution to a given problem in a reasonable amount of time. On the other hand, Fuzzy-based algorithms are the best suited for solving problems that is involved with vagueness or uncertainty [17]. While these algorithms can be used as an approach to solve the service placement problem, the demerits of the algorithms outweighs the benefits gained from them. Heuristic algorithms are not as flexible as DLR algorithms, which limits their potential in larger networks. On the other hand, fuzzy-based algorithms does perform well in dynamic environment, but DLR algorithms are able to solve complex problems efficiently in addition to the flexibility that fuzzy-based algorithms provides.

Based on the approaches that were reviewed in this study, reinforcement learning methodologies is proven to be effective in solving service placement problems. Specifically, deep reinforcement learning methods is often used over traditional model free reinforcement learning methods as DRL methods are better at handling high dimension problems more effectively. Furthermore, simulations were used as the main training methods for the models. This allows the models the freedom to explore with less bias towards a given data set. This is especially important as the models may produce unexpected results that is optimal towards the goal. The different algorithms also displays the flexibility of DRL, by showcasing the various approaches to the service placement and allocation problem using different optimizations. There wasn't a specific best optimization that could be used to solve the service placement problem, however single objective optimization should be prioritized as Multi-objective optimization is much more challenging.

5 Conclusion

This study explored the fog and edge service placement problem and reinforcement learning as a potential solution. The service placement problem is proved to be challenging since the environment are highly dynamic. To present the possible solutions to the service placement problem, three main reinforcement learning approaches were examined. The various RL approaches all had a different value to be optimized, which had huge influence on the design on the algorithms. Deep reinforcement learning in particular has proven to be effective at producing results to optimization problems in the different approaches. Furthermore, simulations were also used in all three approaches to train the model instead of using data sets. As a consequence, the training process are generally very long. While there was not a clear method that surpasses all the other approaches, the methods displays the advantages of RL in solving the service placement problems.

References

[1] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.

[2] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[4] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020.

[5] Farah Aït Salaht, Frédéric Desprez, and Adrien Lebre. An overview of service placement problem in fog and edge computing. *ACM Comput. Surv.*, 53(3), jun 2020.

[6] Filippo Poltronieri, Mauro Tortonesi, Cesare Stefanelli, and Niranjan Suri. Reinforcement learning for value-based placement of fog services. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 466–472, 2021.

[7] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

[8] Ziyao Zhang, Liang Ma, Kin K. Leung, Leandros Tassiulas, and Jeremy Tucker. Q-placement: Reinforcement-learning-based service placement in software-defined networks. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1527–1532, 2018.

[9] Reinout Eyckerman, Phil Reiter, Steven Latré, Johann Marquez-Barja, and Peter Hellinckx. Application placement in fog environments using multi-objective reinforcement learning with maximum reward formulation. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2022.

[10] Filippo Poltronieri, Mauro Tortonesi, Alessandro Morelli, Cesare Stefanelli, and Niranjan Suri. Value of information based optimal service fabric management for fog computing. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2020.

[11] Tong Liu, Shenggang Ni, Xiaoqiang Li, Yanmin Zhu, Linghe Kong, and Yuanyuan Yang. Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing. *IEEE Transactions on Mobile Computing*, pages 1–1, 2022.

[12] Mehdi Mohammadi, Ala Al-Fuqaha, Mohsen Guizani, and Jun-Seok Oh. Semisupervised deep reinforcement learning in support of iot and smart city services. *IEEE Internet of Things Journal*, 5(2):624–635, 2018.

[13] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*, pages 426–440. Springer, 2018.

[14] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[15] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

[16] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*, pages 11–20. PMLR, 2019.

[17] Bushra Jamil, Humaira Ijaz, Mohammad Shojafar, Kashif Munir, and Rajkumar Buyya. Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. *ACM Comput. Surv.*, 54(11s), sep 2022.

Digital Scent in Mulsemedia applications

Leonardo Pasquarelli
leonardo.pasquarelli@aalto.fi

Tutor: Nassim Sehad

Abstract

Multisensory media applications have gained a lot of traction in the past years. Scent is one of the senses that has yet to be incorporated into commercial products. Much research has been conducted on digital scent in Mulsemedia applications, for which we aim to review the most relevant ones. In this literature review, we investigate the application areas and present state-of-the-art technologies and devices. Furthermore, we will discuss the future of scent in mulsemedia applications and how likely it is to be a key component with the announcing date of 6G in 2030.

KEYWORDS: internet of senses, mulsemedia, metaverse, virtual reality, digital scent

1 Introduction

The term Internet of Senses (IoS) was first introduced by Ericsson in their Consumer Lab [1]. While most companies focus on delivering audio and visual senses in their Virtual Reality applications [2], IoS looks beyond these two senses. It can cover olfactory senses, and bodily senses, but also a change of the environment.

Stimulating a wider array of senses than just audio and visual can lead

to a more realistic perception of the environment and overall to an improvement of the Quality of Experience (QoE), without increasing cybersickness. On top of that, it was found that the stimulation of multiple senses allow for a decrease of the quality of the VR video, without significantly damaging the QoE [3]. Other reasons to develop IoS applications, is that there are numerous application fields including but not limited to Healthcare, Military, Work and Education. Such types of Multimedia applications are often referred to, as Multi-sensory multimedia (Mulsemedia) applications.

Mulsemedia applications face numerous technical challenges such as Latency and QoE [2], later presented in Section 5. Such challenges can be addressed via modern technologies. This paper reviews the technologies that enable Mulsemedia applications, the application areas as well as the open research and challenges, with a special focus on digital scent and devices stimulating the olfactory system.

This paper is organized as follows. The most prominent enabling technologies, namely Digital Scent and Networking technologies, are presented in Section 2. Next, application areas of Mulsemedia are discussed in Section 3. Furthermore, the state-of-the-art digital scent technology as well as devices that are being developed are exhibited in closer detail in Section 4. Section 5 peeks at the Open Research, including the challenges that Mulsemedia applications bring along and whether digital scent could be commonly used in Mulsemedia applications by 2030. Lastly, we give a final conclusion in Section 6.

2 Enabling Technologies

Many technologies and devices are already in place in commercial VR-applications, such as audio devices and tracking devices. Such devices focus on audiovisual senses.

In order to stimulate additional senses, additional technologies are needed. We will first talk about Networking technologies, as they become relevant in Mulsemedia applications in order to address latency and communication challenges. Afterwards, we will discuss scent technologies. More research is required on some missing enabling technologies, such as the synchronization of multiple senses [2].

2.1 Networking

In order to provide a great user experience for Mulsemedia applications, it is crucial to minimize the latency so that the virtual world's interactions feel smooth. To achieve an immersive user experience, a latency of at most 20 ms is needed, to provide the feeling of being in a different location [4]. For other applications, such as digital twins, this constraint might be more tight. In order to approximate this goal, it is desirable to make efficient use of fast networking technologies.

The technologies listed in the following are based on a case study of a remote UAV, deployed for a Mulsemedia application. If it is needed to connect to a non-stationary remote device, 5G is typically used in order to connect to the device. Such applications are often deployed on the Edge, in order to leverage the computation to the cloud while keeping the latency at its minimum. Lastly, the use of modern protocols, such as WebRTC and QUIC help minimize the latency for video streaming and sending commands from the local user to the remote machine. [5].

While video streaming mainly requires high-throughput, another challenge is to synchronize all incoming sense streams [6]. The transmission of smells does not require high throughput, as little data about the scent has to be transferred. We didn't furhter look into existing protocols and formats for transferring scnent, however, research on the topic has been conducted [7].

2.2 Scent

In the following, the technologies related to digital scent are addressed. Broadly speaking, digital scent can be split into scent recognition and scent synthesis. Recognition works through electrochemical sensors and classifying detected scents based on an existing data set, oftentimes in combination with machine learning. [8]. Devices for both categories for both chemical [9] and electrical stimulation [10] have been developed already.

Digital scent is synthesized through chemistry or electrical stimulation. When producing a scent through chemistry, cartridges are used, typically to release molecules, which evaporate in the air and create a certain smell. However, using up and refilling the cartridges can be quite costly. On top of that, the produced scents are limited by the variety of cartridges [8].

In the case of electrical stimulation, studies were conducted where par-

ticipants were exposed to electrical stimulation at the trigeminal nerve and the olfactory bulb. This would lead to a sensation of smell. Using electrical stimulation, participants weren't able to reliably feel a smell and the electricity would sometimes lead to pain [8]. However, research on finding more innovative scent synthesis technologies using electrical signals is conducted [10].

Scent will be the main focus of this paper, and existing scent devices will be discussed more thoroughly in Section 4.

3 Application areas

Mulsemedia serves a lot of different use cases. Following the theme of this paper, we will focus on the applications which especially make use of the olfactory sense. The application areas that we will focus on are on *Health and well-being* and on the *Industry*.

There exist other use cases, where scent can be used, but where a different sense has a greater impact on the application.

For instance, mulsemedia enables the healthcare industry to perform surgeries remotely, by communicating Haptics [2]. While scent could be used to communicate some data between the patient and the practitioner, it's by far not as relevant as the communication of haptics.

3.1 Health and well-being

Mulsemedia can be used to improve someone's health and well-being. By analysing the scent and air quality inside someone's home, a remote practitioner can make a diagnosis of certain medical conditions [8].

Since our olfactory system is linked to our emotions and memories, the use of digital scents can have an impact on our well-being. Certain scents can be used in order to create a relaxing environment. This holds for scents that users associate with positive memories, as well as scents that are relaxing, such as lavender or the smell of grass. This is especially useful for moments in which a person faces anxiety. In combination with digital scent, immersing oneself in a mulsemedia application of a walk in nature can be utilized to reduce anxiety and stress [2].

3.2 Industry

Tourism mulsemedia applications greatly benefit from enhancing the user experience using mulsemedia. [8]. A common strategy would be to use scents in virtual tours. Our olfactory system is linked to our memories, and therefore smelling scents that are associated with different locations can evoke an immersive feeling. As an example, woody smells could be used at historical buildings, incense and myrrh could be used inside religious buildings and marine, citrusy and aquatic smells could be used for beach resorts.

The same goes for art and entertainment. Digital art content can make use of scent as an additional sense, to portray the piece of art differently. Video games and movies can augment the perception as well. Similarly to the tourism industry, scents can be attached to different scenes and locations, to augment the experience.

Another branch benefiting from digital scent is marketing [8]. Likeable and fitting smells can be attached to a brand in order to influence customer behaviour unconsciously.

3.3 Food

Furthermore, scent cannot only be used to evoke more realistic experiences, but also to simulate the taste of food [11]. The applications described in the paper range from online taste sampling, research and potentially reducing obesity such as by fighting flavour addiction using digital taste, without the user needing to eat.

4 Scent devices

In this section, we'll talk about scent devices developed as commercial and scientific devices. Our focus will be on scent-producing devices, rather than scent-recognising devices, because scent-producing devices have more use cases in Mulsemedia applications, than scent-recognising devices and because the focus of mulsemedia research has been on scent production devices [21]. Additionally, based on our findings, within the mulsemedia sector, companies have developed more scent-producing devices than scent-recognising devices.

The devices were found on our own through search engines and through

Table 1. Overview of scent production devices

Ref	Name	Type	Format	Platform	Use Cases	# Cartridges
Commercial devices						
[12]	OVR	C	WRBL	Unity, Unreal	WELL, ENT	1
[13]	Olorama	C	DESK	Unity, Unreal, API	ENT	10-12
[14]	Aromajoin	C	DESK	Unity, SDKs	MAR	6
[15]	Inhalio	C	DESK	SDKs	WELL, MAR	4
[16]	ScentRealm	C	DIFF	App	WELL, ENT	12
Research Devices						
[17]	OSpace	C	DESK	Arduino	WELL	6
[9]	inScent	C	WRBL	FRA	WELL	8
[18]	Essence	C	WRBL	API	WELL	1
[19]	BioEssence	C	WRBL	API	WELL	3
[20]	Season Traveller	C	WRBL	BLE Nano	ENT	4
[10]	Nose pin	E	WRBL	-	ENH	-

Includes: Name (Either the company name, or the device name, or a conceptual name if neither exist), Type (C=Chemical, E=Electrical), Format (WRBL=Wearable, DESK=Desktop, DIFF=Diffuser), Platforms Disclosed (Unity, Unreal Engine, API for Programming language, SDK for programming language, App, Arduino, FRA=Framework, BLE Nano), Use Cases (WELL=Wellbeing, ENT=Entertainment, MAR=Marketing, ENH=Enhancement of Smell), Number of Cartridges

literature surveys [22, 23, 24]. We did not include all devices, as some of them were outdated or not unique enough, compared to other devices. Some of the devices, listed in [24], were not necessarily intended for Mulsemedia.

All the commercial scent production devices that we looked at produce scent chemically and not electronically. A possible reason for this, is that research has focused much more on chemical production, rather than the electrical stimulation of the nose.

For an overview of the described devices, please look at Table 1.

4.1 Commercial Devices

The listed devices have their own unique features. The devices that OVR develops combine different aromas from a single scent cartridge, to create thousands of scents [12]. Olorama, Inhalio and ScentRealm all focus on a wide array of use cases and have separate devices available [13, 15, 16]. Aromajoin supports a change of scents within 0.1 seconds, and it supports blending scents granularly [14].

4.2 Research Devices

After OSpace was presented, it was reused in other studies [17]. In the cited study, scents were released, to notify the driver about car-related information.

inScent, Essence and Bioessence are wearable necklaces. With inScent, mobile notifications could be paired with a scent [9]. For example, if the user received messages from important contacts, or calendar notifications, different scents could be released. Essence and its modified version, Bioessence rely on contextual and biometric data, to release scent in order to improve wellbeing [18, 19].

The Season Traveller is a VR application, in which the user goes on a journey through four seasons. On its journey, the olfactory, as well as haptic senses are stimulated, in order to create a more realistic user experience.

Finally, the nose pin is a device developed to enhance the smell of gas and thus locate gas leaks [10]. While the application itself is not related to Mulsemedia, the device showed an improvement compared to previous devices using electrical stimulation, as the device would work reliably and not cause pain [8].

5 Discussion

In the following, we will look at the challenges that research is facing, and we will consider whether scent can become a key component by 2030, the announcing date of 6G. We will also look at the negative aspects of digital scent and take them into account in the discussion.

In terms of networking, mulsemedia applications face some challenges. When it comes to scent, low latency is arguably one of the most important components. Given that 6G networks aim to provide latencies of 1ms or less, this requirement would be satisfied. [25]

Another aspect regarding quality of experience is to synchronize the streams of senses, which requires more research.

High throughput is not required for scent-intensive applications, but rather for applications which are heavy on video transmission, such as holographic communication.

Overall, the opportunities that digital scent brings in Mulsemedia applications are great, as laid out in Section 3. The scent devices themselves are capable enough already, to be used in mulsemedia applications.

Although there still exist some challenges, regarding Quality of Experience and networking, it doesn't seem unrealistic that those challenges will be solved by 2030.

A potential problem with distributing digital scent to consumers is the cost of digital scent. Olorama's *Scented VR Pack* that comes with a device and 10 scents retails for €2,449 [13]. A more affordable alternative is to build a scent device on its own. However, it is questionable, whether the average consumer would be willing to invest the time into this process.

Moreover, based on our findings, it is unknown what the user retention is, that is how long the users keep using a scent device. Users might stop being interested in using scent devices for various reasons. Scent may cause headaches for some subjects. Finally, when using chemically produced scents, which is the current norm of the industry, the scent needs to be restocked, which increases the price point.

6 Conclusion

As presented in this literature review, digital scent has gained a lot of attention in research and in the industry. Due to its large amount of ap-

plications, some companies also decided to develop digital scent devices. Based on our discussion, the technology behind digital scent is likely to be mature enough to be widely used in multimedia applications by 2030, the announcing date of 6G. On the other hand, it's hard to predict how many users of mulsemedia applications would acquire scent devices and regularly restock cartridges. Further research is required for some challenges presented earlier, such as the synchronization of scents.

References

[1] "Internet of touch - Future technologies," https://www.ericsson.com/en/6g/internet-of-senses. [Online]. Available: https://www.ericsson.com/en/6g/internet-of-senses (Accessed 2023-01-21).

[2] T. H. Falk, L. B. Le, and R. Morandotti, "The Internet of Senses: A Position Paper on the Challenges and Opportunities of Multisensory Immersive Experiences for the Metaverse," in *2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*. Rome, Italy: IEEE, Oct. 2022, pp. 139–144. [Online]. Available: https://ieeexplore.ieee.org/document/9967586/ (Accessed 2023-01-20).

[3] B. De Jesus Jr, M. Lopes, M.-A. Moinnereau, R. A. Gougeh, O. M. Rosanne, W. Schubert, A. A. Oliveira, and T. H. Falk, "Quantifying Multisensory Immersive Experiences using Wearables: Is (Stimulating) More (Senses) Always Merrier?" in *Proceedings of the 2nd Workshop on Multisensory Experiences-SensoryX'22*. SBC, 2022.

[4] R. Vargic, M. Medvecký, J. Londák, and P. Podhradský, "Advanced Interactive Multimedia Delivery in 5G Networks," in *Interactive Mobile Communication Technologies and Learning*, ser. Advances in Intelligent Systems and Computing, M. E. Auer and T. Tsiatsos, Eds. Cham: Springer International Publishing, 2018, pp. 421–430.

[5] T. Taleb, N. Sehad, Z. Nadir, and J. Song, "VR-based Immersive Service Management in B5G Mobile Systems: A UAV Command and Control Use Case," *IEEE Internet of Things Journal*, pp. 1–1, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9951155/ (Accessed 2023-01-20).

[6] R. Joda, M. Elsayed, H. Abou-zeid, R. Atawia, A. B. Sediq, G. Boudreau, M. Erol-Kantarci, and L. Hanzo, "The Internet of Senses: Building on Semantic Communications and Edge Intelligence," *IEEE Network*, pp. 1–9, 2022, conference Name: IEEE Network.

[7] P. Isokoski, K. Salminen, P. Müller, J. Rantala, V. Nieminen, M. Karjalainen, J. Väliaho, A. Kontunen, M. Savia, J. Leivo, A. Telembeci, J. Lekkala, P. Kallio, and V. Surakka, "Transferring scents over a communication network," in *Proceedings of the 23rd International Conference on Academic Mindtrek*, ser. AcademicMindtrek '20. New York, NY, USA: Association for Computing Machinery, Feb. 2020, pp. 126–133. [Online]. Available: https://doi.org/10.1145/3377290.3377301 (Accessed 2023-04-04).

[8] D. Panagiotakopoulos, G. Marentakis, R. Metzitakos, I. Deliyannis, and F. Dedes, "Digital Scent Technology: Toward the Internet of Senses and the Metaverse," *IT Professional*, vol. 24, no. 3, pp. 52–59, May 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9811514/ (Accessed 2023-01-20).

[9] D. Dobbelstein, S. Herrdum, and E. Rukzio, "inScent: a wearable olfactory display as an amplification for mobile notifications," in *Proceedings of the 2017 ACM International Symposium on Wearable Computers*. Maui Hawaii: ACM, Sep. 2017, pp. 130–137. [Online]. Available: https://dl.acm.org/doi/10.1145/3123021.3123035 (Accessed 2023-01-31).

[10] J. Brooks, S.-Y. Teng, J. Wen, R. Nith, J. Nishida, and P. Lopes, "Stereo-Smell via Electrical Trigeminal Stimulation," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Yokohama Japan: ACM, May 2021, pp. 1–13. [Online]. Available: https://dl.acm.org/doi/10.1145/3411764.3445300 (Accessed 2023-01-23).

[11] A. S. Duggal, R. Singh, A. Gehlot, M. Rashid, S. S. Alshamrani, and A. S. AlGhamdi, "Digital Taste in Mulsemedia Augmented Reality: Perspective on Developments and Challenges," *Electronics*, vol. 11, no. 9, p. 1315, Apr. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/9/1315 (Accessed 2023-02-28).

[12] "OVR Technology," https://ovrtechnology.com/. [Online]. Available: https://ovrtechnology.com/ (Accessed 2023-02-13).

[13] "Olorama," https://www.olorama.com/. [Online]. Available: https://www.olorama.com/ (Accessed 2023-02-13).

[14] "Aromajoin: Digital Scent Technology for Modern Scent Marketing, Aroma Therapy and Olfactory Multimedia," https://aromajoin.com. [Online]. Available: https://aromajoin.com (Accessed 2023-02-20).

[15] "Inhalio," https://inhalio.com/. [Online]. Available: https://inhalio.com/ (Accessed 2023-02-20).

[16] "ScentRealm," https://www.qiweiwangguo.com/. [Online]. Available: https://www.qiweiwangguo.com/ (Accessed 2023-02-28).

[17] D. Dmitrenko, E. Maggioni, and M. Obrist, "I Smell Trouble: Using Multiple Scents To Convey Driving-Relevant Information," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*. Boulder CO USA: ACM, Oct. 2018, pp. 234–238. [Online]. Available: https://dl.acm.org/doi/10.1145/3242969.3243015 (Accessed 2023-02-28).

[18] J. Amores and P. Maes, "Essence: Olfactory Interfaces for Unconscious Influence of Mood and Cognitive Performance," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Denver Colorado USA: ACM, May 2017, pp. 28–34. [Online]. Available: https://dl.acm.org/doi/10.1145/3025453.3026004 (Accessed 2023-02-20).

[19] J. Amores, J. Hernandez, A. Dementyev, X. Wang, and P. Maes, "BioEssence: A Wearable Olfactory Display that Monitors Cardio-respiratory Information to Support Mental Wellbeing," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and*

Biology Society (EMBC). Honolulu, HI: IEEE, Jul. 2018, pp. 5131–5134. [Online]. Available: https://ieeexplore.ieee.org/document/8513221/ (Accessed 2023-02-28).

[20] N. Ranasinghe, P. Jain, N. Thi Ngoc Tram, D. Tolley, Y. Liangkun, C. Eason Wai Tung, C. C. Yen, E. Y.-L. Do, K. C. R. Koh, and K. Shamaiah, "A Demonstration of Season Traveller: Multisensory Narration for Enhancing the Virtual Reality Experience," in *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '18. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–4. [Online]. Available: http://doi.org/10.1145/3170427.3186513 (Accessed 2023-02-28).

[21] A. Covaci, L. Zou, I. Tal, G.-M. Muntean, and G. Ghinea, "Is Multimedia Multisensorial? - A Review of Mulsemedia Systems," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–35, Sep. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3233774 (Accessed 2023-01-20).

[22] Y. Pan, K. Ono, J. Huang, and M. Watanabe, "A Review of Olfactory Display Devices from a Habit-Forming Perspective," *Journal of the Science of Design*, vol. 5, no. 1, pp. 1_27–1_36, 2021.

[23] E. B. Saleme, A. Covaci, G. Mesfin, C. A. S. Santos, and G. Ghinea, "Mulsemedia DIY: A Survey of Devices and a Tutorial for Building Your Own Mulsemedia Environment," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–29, May 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3319853 (Accessed 2023-02-28).

[24] A. K. Holloman and C. S. Crawford, "Defining Scents: A Systematic Literature Review of Olfactory-based Computing Systems," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 18, no. 1, pp. 1–22, Jan. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3470975 (Accessed 2023-02-16).

[25] S. A. Abdel Hakeem, H. H. Hussein, and H. Kim, "Vision and research directions of 6G technologies and applications," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2419–2442, Jun. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1319157822001033 (Accessed 2023-03-02).

Algorithmic Power

Markus Kähkönen
markus.kahkonen@aalto.fi

Tutor: Antti Rannisto

Abstract

This paper investigates the concept of algorithmic power and its impact on individuals, while also exploring various strategies for resistance against it. The study examines academic literature on algorithmic power, highlighting the disconnect between research that portrays users as passive victims and research that emphasizes user agency and resistance. By analyzing these two perspectives, the paper proposes a comprehensive approach to better understand the complex dynamics of algorithmic power. Furthermore, it highlights the need for increased transparency from companies in the context of algorithmic power. Ultimately, the paper calls for a more balanced and nuanced analysis that considers both the power of algorithms and the potential for individuals to resist or adapt to these influences.

KEYWORDS: algorithms, algorithmic power, sociology

1 Introduction

In recent years, algorithms have become an intrinsic part of our daily lives, shaping and influencing our experiences and decisions in ways that we often do not understand [1, 2]. The increasing influence of algorithms has made the concept of algorithmic power a pressing concern. The term

algorithmic power refers to the control and influence that algorithms and the companies that develop them have over individuals, society, and the economy [1,3]. This power is derived from the ability of algorithms to process and analyze vast amounts of data, automate decision-making processes, and shape our experiences, beliefs and behavior [4]. Some challenges associated with algorithmic power include biased algorithms, lack of transparency, and privacy concerns.

This paper examines the literature surrounding the concept of algorithmic power from two distinct perspectives. The first approach focuses on the analysis of utilization of algorithmic power, which is examined through the lens of critical sociology of algorithms. The second approach addresses resistance to the use of algorithmic power and the efforts to counteract its effects, studied within the framework of sociology of algorithmic critique. The scope of this literature review is mainly limited to peer reviewed articles and books relevant to algorithmic power and associated concepts. Due to the field's vast and rapidly changing nature, this paper highlights a select few foundational sources, predominantly from Western societies, which may not fully represent global experiences.

This paper is organized as follows: Section 2 provides an overview of the critical sociology of algorithms, highlighting the key themes. Section 3 examines resistance to algorithmic power, exploring the strategies and tactics that individuals and groups use to challenge or undermine the influence of algorithms. Section 4 analyzes the main findings of the paper, reflects on the implications of these findings, and suggests avenues for future research in this field. Finally, section 5 draws conclusions from the analysis and discusses the significance of algorithmic power in the future.

2 Utilization of Algorithmic Power

Algorithmic power is a relatively new type of power that has emerged from the widespread use of digital technologies and big data. It refers to the power that is exerted by algorithms, which are sets of rules or instructions that are used to solve a problem or perform a task [3].

Companies are among the most prominent users of algorithmic power, as they use algorithms as a tool to achieve their goals. Profit-oriented companies in particular are likely to choose algorithms that maximize their profits, rather than settling for neutral options. Thus, it is important to analyze algorithmic power. This section analyzes the literature on the

overall power dynamics of algorithmic power, big data, and surveillance capitalism in the context of algorithmic power.

2.1 The power of algorithms

Firstly, it is crucial to examine how algorithms can possess power. Unlike traditional forms of power, such as economic or political power, algorithmic power usually functions on a subtler level [1,5]. Algorithms typically exert power by shaping individuals' or groups' choices, rather than using overt coercion or persuasion [6]. As a result, this form of power is less noticeable to users compared to, for example, legislative power, since algorithms often operate behind the scenes [7].

This does not mean that algorithms lack power, as Schwartz [1] argues in his book that content recommendations, for instance, exert a form of power since they can result in tangible changes in our lives. An example he provides is that a navigation software cannot directly dictate which route a user will take, but it significantly influences their decision-making process. Furthermore, these decisions may also have unintended consequences that are not immediately apparent. For instance, if a navigation software consistently promotes a particular road to its users, it can lead to increased traffic on that road, which may cause other issues.

The power of algorithms extends beyond recommendations, as individuals and companies are becoming increasingly reliant on completely automated decisions made by algorithms [2,4]. This increase in automation is advantageous for companies, as algorithms can analyze vast amounts of data in milliseconds at a fraction of the cost instead of using humans for the same task [7]. One major concern is that these algorithms are not only used for trivial decisions, but can also decide who gets insurance coverage, loans, and even job opportunities [4]. This raises significant concerns about the implications of algorithmic power becoming too vast and unregulated.

A key challenge in analyzing algorithmic power lies in understanding its true impact. Olhede and Wolfe [8] suggest that public discourse tends to swing between "unrealistic expectations" and "overblown fears," leading to significant uncertainty about the real implications of algorithmic power. This can be attributed, in part, to science fiction films like Terminator, which emphasize existential fears surrounding autonomous weapon systems, even though such concerns may not be currently warranted [8]. As a result, the actual capabilities and limitations of algorithms might

not be clear to the average person. Furthermore, attempts to simplify algorithmic capabilities often result in oversimplification, which can distort their potential and functionality [9]. By recognizing the multifaceted nature of algorithms, we can develop a more nuanced understanding of their roles, applications, and potential impacts on various aspects of society and human life [9].

2.2 Big data and Surveillance capitalism

One important term in relation to algorithmic power is big data. Big data refers to the massive amount of data generated by individuals, organizations, and machines that cannot be analyzed via traditional means [7, 10]. The extent to which data is collected is often not understood by individuals, since companies can effectively collect everything and use that information to infer even minute details about a person [7,11]. The immense value of big data incentivizes companies to gather as much information as possible, often without meaningful consent or fair compensation [11]. While big data itself is not inherently a form of algorithmic power, it serves as an essential foundation for algorithmic power to function effectively.

The rise of algorithms and availability of data in our daily lives has given way to a new type of socio-economic structure called surveillance capitalism [4, 12]. This structure is characterized by the collection of vast amounts of data about individuals, which is not done for their benefit, but rather for the benefit of large corporations[4]. As Zuboff writes in her book "The Age of Surveillance Capitalism" [4] algorithmic power has led to a new type of power structure that revolves around the use of data to monitor, track, and manipulate individuals.

The model of surveillance capitalism as described by Zuboff [4] is a really harsh depiction of algorithmic power where human experience is used as “raw material” for behavioral data. This data is fabricated into “prediction products” used to create behavioral futures markets, in which surveillance capitalists trade on predictions of human behavior. Furthermore, Zuboff [4] claims that the power of surveillance capitalism is it can shape and modify human behavior at scale, subordinating the means of production to an increasingly complex means of behavioral modification. She goes even as far as calling surveillance capitalism a parasitic force, which threatens “to cost us our humanity“.

This portrayal of surveillance capitalism is rather dystopian, as it im-

plies that nothing good can come from algorithms. However, it is crucial to recognize that Shoshana Zuboff [4] does not label algorithms as inherently negative or positive. Instead, she highlights the exploitative nature of surveillance capitalism and its associated practices, which often involve the misuse of algorithmic power.

While surveillance capitalism's abuses of algorithmic power are concerning, algorithms can also be harnessed for positive outcomes, such as improving healthcare, education, and environmental sustainability [8, 10]. Furthermore, it's essential to acknowledge that algorithms are the only realistic method for analyzing vast data amounts, making research into them important [10]. Thus, it is crucial to carefully consider the ethical implications of algorithmic decision-making and ensure that they align with human values and rights.

2.3 Ethical considerations of Algorithmic power

Algorithmic power raises several ethical considerations that need to be addressed [13]. A major concern is the potential for discrimination and bias in algorithmic decision-making, as algorithms are only as objective as the data they are trained on [13]. If the data contains biases, these biases will be reflected in the decisions made by the algorithm. Moreover, models can become outdated or flawed over time. For example, if an algorithm is trained on historical data that reflects biased hiring practices, the algorithm may continue to perpetuate these biases by favoring certain groups of people over others [13].

Cathy O'Neil [13] in her book, as an example, relates problems of bad data in algorithmic power to racism. She argues that racism, at an individual level, can be seen as a predictive model operating in the minds of billions of people around the world. This model is built from faulty, incomplete, or generalized data that indicates certain types of people have behaved badly, leading to the binary prediction that all people of that race will behave in the same way. Racists rarely spend time searching for reliable data to train their models, and once their model becomes a belief, it is hardwired and generates poisonous assumptions. While O'Neil's example is not inaccurate, it shares similarities with Zuboff's portrayal of Surveillance capitalism, in that it lacks nuance and presents a predominantly negative perspective.

2.4 Algorithmic accountability

As reliance on automated decision-making grows for critical aspects of human life, such as employment and prison sentencing, biases in algorithmic power have become a pressing concern [1]. The use of algorithms in these contexts raises questions about accountability, as it is often difficult to determine who is responsible for the outcomes of algorithmic decisions [3]. This issue becomes even more significant when considering the fact that algorithms are susceptible to errors [3]. Consequently, addressing the matter of algorithmic accountability and determining who should be held responsible for decisions made by automated systems is crucial [3].

Diakopoulos [3] argues that algorithms should be viewed as "objects of human creation". As such, algorithmic accountability should take into account the human influences embedded in algorithms, including criteria choices, training data, semantics, and interpretation [3]. Additionally, the intentions of any group or institutional processes that may have influenced the algorithm's design and the agency of human actors in interpreting the output of algorithms must be considered when evaluating algorithmic accountability [3]. However, this approach can become problematic when attempting to pinpoint exactly who to blame for a mistake made by an algorithm, as bad data might be considered a mitigating factor. This challenge is akin to determining fault when a company's employee makes a mistake, since it can be difficult to discern whether responsibility lies solely with the individual or if the company should also be held accountable as well.

3 Resistance to algorithmic power

The analysis of algorithmic power often neglects the role of user agency, as the prevailing narrative tends to emphasize the negative impacts of algorithms on individuals [5]. However, it is essential to recognize that there are many forms of resistance to algorithmic power. These can range from individual actions, such as modifying privacy settings, data collection settings or disregarding recommendations, to collective efforts, such as advocating for algorithmic transparency and accountability.

The availability and use of data is a crucial aspect of algorithmic power, particularly in the context of big data [14]. The accuracy of algorithmic predictions heavily relies on the quality and quantity of accessible data

[14]. Without relevant data, algorithms cannot effectively perform their intended functions. Consequently, the power of algorithms is intrinsically connected to the availability and application of data. As a result, an effective method of resisting this power is by limiting data availability.

This section explores diverse strategies for resisting algorithmic power, including transparency in algorithmic systems, legislative restrictions such as GDPR, and individual-level tactics like selective information sharing and using ad blockers. By examining these methods, we can empower users, promote equitable and accountable algorithmic systems, and mitigate the negative consequences of algorithmic power on society.

3.1 Transparency

Achieving transparency is a key concept in resisting algorithmic power as it involves making information about AI systems and their decision-making processes accessible and understandable to individuals [3]. However, achieving transparency is not as easy for algorithms as it is for other forms of power, such as political power. The hidden motives of politicians can be analyzed by looking at who paid them, but algorithms are often hidden behind layers of technical complexity, making it difficult for individuals to understand how they work and the decisions they make [3]. Thus, it is important to define how to achieve transparency, as large algorithmic systems like neural networks are not human-readable, and the data collected is immense [3]. Furthermore, there is the question of whether companies should disclose their trade secrets.

Diakopoulos [3] emphasizes the importance of striking a balance between disclosure and secrecy when it comes to algorithmic systems. Requiring companies to disclose their entire source code, for example, would give away all of their trade secrets. Furthermore, creating a standard for what should be disclosed will be challenging since different companies collect and use data in completely different ways. Therefore, it is necessary to carefully consider what information should be made public and what can remain proprietary to ensure a fair and transparent system.

3.2 Legislative Restrictions

Legislative restrictions are another form of resisting algorithmic power. Governments worldwide have recognized its potential negative consequences and enacted laws to mitigate them [14]. These primarily focused on man-

aging data acquisition and usage [14]. One of the most prevalent legislative restrictions has been the European Union's General Data Protection Regulation (GDPR), which aims to protect individuals' personal data and regulate how it is collected, processed and stored by companies [15].

The GDPR provides individuals with greater control over their personal data, including the right to access, correct, and erase it [15]. It also requires companies to obtain explicit consent from individuals before collecting and processing their data [15]. However, merely granting individuals control over their data may not necessarily diminish algorithmic power, as many users might consent to data collection without much consideration. Additionally, consent forms, such as end-user licensing agreements (EULAs), can be one-sided, non-negotiated, and presented in lengthy, dense legal language, making it difficult for users to be fully informed of the terms and conditions they are agreeing to [11]. However, GDPR seeks to mitigate this issue by implementing a series of requirements that promote transparency and enhance user understanding.

While legislative restrictions can be effective in mitigating the negative consequences of algorithmic power, there are also challenges associated with their implementation. For example, there may be difficulties in enforcing laws across different countries with varying legal systems and cultural norms. Additionally, it is arguable that strict regulations may reduce innovation and hinder technological progress [3]. Another limitation of legislative restrictions is that they can be slow to adapt to rapidly changing technological advancements, which may require immediate responses. However, overarching legislative changes are crucial in combating algorithmic power, as changes at the European Union(EU) level are likely to have a significant impact on companies operating outside of the EU. This is because companies operating outside of the EU might find it challenging to maintain two different systems.

Overall, legislative restrictions can be an effective way to empower users to control their personal data and ensure that sensitive information, such as health and personal information, is protected from being sold or misused [15]. In this regard, legislative restrictions are crucial for safeguarding individuals' privacy and rights.

3.3 Individual-Level Resistancel

Although legislative measures have expanded users' control over their data, there are additional strategies for resisting algorithmic power at

the individual level. One such approach involves selectively sharing information with algorithms, providing only necessary data while withholding or falsifying other details [5]. In recent times, the use of virtual private networks has increased, making it increasingly challenging to obtain accurate location data. This effect can also extend to other users, since if a noticeable proportion of users falsifies their data, the overall dataset becomes less trustworthy since algorithms cannot distinguish between authentic and fake information. Users can also limit the data they generate by adjusting their online activity or declining data collection entirely. By employing these tactics, individuals can undermine the accuracy and effectiveness of algorithms, reducing their power over personal information.

A powerful tool for individuals to resist algorithmic power is the use of ad blockers. This method is significant as it effectively neutralizes one of the most prevalent forms of algorithmic power meaning manipulation through targeted advertising [4]. This is also important for resisting the power of larger tech companies such as Google and Meta as advertisement constitutes a large portion of their income [1].

4 Analysis

After reviewing the literature in this paper, it is evident that there is a disconnect between some academic writing and the reality of algorithmic power. While some academic works (e.g. [4], and [13]) depict algorithmic power as something exerted over unsuspecting victims who are unaware of its influence, writing from the perspective of resistance (e.g., [5] and [15]) highlights ways in which individuals can resist and undermine its power, demonstrating that users are not merely passive victims. Moreover, in academia, the focus on individual users' resistance to algorithmic power seems to be less prevalent, potentially contributing to the disconnect in perspectives within the literature.

For example, academic works focusing on the negative aspects of algorithmic power might discuss how algorithms can be used to manipulate user behavior and exploit personal information, often portraying users as helpless in the face of these forces. On the other hand, research on resistance to algorithmic power showcases the agency and resilience of users and governments, revealing their ability to selectively share information, alter their online behavior, and employ other tactics to counteract algorithmic power. This contrast between passive victimhood and active

resistance highlights the discrepancy found within the literature.

By combining the approaches discussed in this paper, a more comprehensive understanding of algorithmic power can be achieved. For example, a study could investigate a company's use of algorithms for data analysis and the resulting conclusions. It could also examine the impact of user resistance, such as selective information sharing or behavior modification, on algorithmic outcomes. This research would deepen our understanding of algorithmic power dynamics and potential resistance. However, effectiveness of such analysis would require a greater transparency from companies, as black box testing alone would be challenging due to limited access to what data is tracked.

One aspect not covered in this review is the impact of artificial intelligence on algorithmic power, which has received limited academic research. This is partly due to the capabilities of AI-generated content not being fully realized until the emergence of large language models like ChatGPT, which can generate text closely resembling human language, blurring the line between human-generated and AI-generated content. Given this development, it is crucial to demand greater transparency from companies regarding how they utilize this technology.

5 Conclusions

This paper analyzed the complex dynamics of algorithmic power, its potential negative consequences and resistance to it. A key finding is that combining perspectives from usage of algorithmic power and resistance to it can lead to a more comprehensive understanding of algorithmic power. However, the effectiveness of resistance to algorithmic power is often contingent on greater transparency from companies utilizing algorithms.

The growing impact of artificial intelligence on algorithmic power, especially through large language models like ChatGPT, underlines the importance of transparency in AI-generated content. Future research should focus on deepening our understanding of algorithmic power dynamics and exploring AI's role in this context. Regulatory bodies, such as the EU, should consider requiring companies to disclose when content is generated by AI models to maintain accountability and protect users from potential manipulation.

6 References

[1] O. Schwarz, "Sociological Theory for Digital Society: The Codes That Bind Us Together" Cambridge, Polity, 2021, pp. 114-160. ISBN 978-1509542970

[2] D. Beer, "The social power of algorithms," Inf. Commun. Soc., vol. 20, no. 1, pp. 1-13, 2017, doi: 10.1080/1369118X.2016.1216147.

[3] N. Diakopoulos, "Algorithmic Accountability," Digit. Journal., vol. 3, no. 3, pp. 398-415, 2015, doi: 10.1080/21670811.2014.976411.

[4] S. Zuboff, The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power. New York: PublicAffairs, 2019, pp.20-200 ISBN: 9781781256855.

[5] J. Velkova and A. Kaun, "Algorithmic resistance: media practices and the politics of repair," Inf. Commun. Soc., vol. 24, no. 4, pp. 523-540, 2021, doi: 10.1080/1369118X 2019.1657162.

[6] T. Gillespie, P. J. Boczkowski, and K. A. Foot, "The Relevance of Algorithms," in Media Technologies: Essays on Communication, Materiality, and Society, Cambridge, MA: MIT Press, 2013, pp. 167-193.

[7] V. Mayer-Schönberger and K. Cukier, Big Data: A Revolution That Will Transform How We Live, Work, and Think. Boston, MA: Houghton Mifflin Harcourt, 2013.

[8] S. C. Olhede and P. J. Wolfe, "The growing ubiquity of algorithms in society: implications, impacts and innovations," Philos. Trans. R. Soc. A, vol. 376, no. 2128, Sep. 2018, doi: 10.1098/rsta.2017.0364.

[9] T. Bucher, If...Then: Algorithmic Power and Politics, New York: Oxford University Press, 2018, doi: 10.1093/oso/9780190493028.001.0001.

[10] Y. Wang, L. Kung, and T. A. Byrd, "Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations," Technol. Forecast. Soc. Change, vol. 126, pp. 3-13, Jan. 2018.

[11] J. Sadowski, "When data is capital: Datafication, accumulation, and extraction," Big Data Soc., vol. 6, no. 1, 2019, doi: 10.1177/2053951718820549.

[12] S. Zuboff, "Big other: Surveillance Capitalism and the Prospects of an Information Civilization," J. Inf. Technol., vol. 30, no. 1, pp. 75-89, 2015, doi: 10.1057/jit.2015.5

[13] C. O'Neil, Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy, New York: Crown, 2016, ISBN: 978-0553418811

[14] S. Barocas and A. D. Selbst, "Big data's disparate impact," Calif. Law Rev., vol. 104, pp. 671-732, 2016.

[15] European Commission, "General Data Protection Regulation (GDPR)," 2018. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protectionen.

Secret Management in Infrastructure as Code

Meri Lemponen
meri.lemponen@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

This seminar paper explores the issues and challenges related to secret management in infrastructure as code (IaC) practices. IaC is a software development practice that configures system dependencies and provisions remote and local instances. The paper analyzes the common issues and practices found in the grey literature, discussing and analyzing two main state-of-the-art resources in this area. The aim of the paper is to provide a good overview of the topic and identify possible gaps in the two main sources. The paper concludes that it is essential to develop common practices and spread awareness of the secret management issues in IaC for practitioners to improve the quality of IaC scripts and ensure overall implementation safety.

KEYWORDS: Secret Management, Infrastructure as Code

1 Introduction

Infrastructure as code (IaC) is a practice in software development that automatically configures system dependencies and provisions remote and local instances [1]. One of the main benefits of using IaC practices is the increased deployment frequency which saves a significant amount of

time in software production. Despite the growing interest in IaC among both the practitioners and the researchers, the state of research regarding secret management in IaC has a lot of room for improvement in the future.

The lack of research on security and secret management in IaC is alarming since the lack of secure practices in IaC can cause significant consequences [1]. One of the biggest issues in secret management in IaC is that IaC scripts are vulnerable by nature to include secrets e.g., hard-coded passwords, usernames, and private cryptographic keys [2]. Therefore, it is vital to develop common practices and spread awareness of the secret management issues of IaC for practitioners so that the quality of IaC scripts will increase and the overall implementation of IaC will become safer.

This seminar paper discusses issues and challenges regarding secret management in IaC. The aim of this seminar paper is to analyze the common issues and practices found in the grey literature. The seminar paper discusses and analyzes two main state-of-the-art resources in this area and the aim of this paper is to provide a good overview of the topic and identify possible gaps in the two main sources.

This paper is organized into 5 sections. Section 2 will provide the background information for understanding the discussion in the later sections. Section 3 discusses the issues and common practices of secret management in IaC. Section 4 will discuss the gaps and challenges found in the literature. Lastly, section 5 will provide a summary of the main findings.

2 Background

The two main topics in this paper are secret management and infrastructure as code. This section defines both of the topics and provides the background that is needed for following the deeper discussion.

2.1 Secret Management

Most common secrets during software development are related to authentication and they can be passwords, credentials, and keys [3]. In some cases, secrets can be considered as key-value pairs that contain secret information and the key-value pair can be used to authenticate or authorize access to services or APIs, e.g., databases and cloud services [4]. The unique key in a key-value pair is used to identify an item in a data

structure and the value is the data associated with the key. To keep the secret safe, the key-value pair must be stored in a secure location and the transition of the key-value must be protected.

Secret management is the management of multiple secrets, which might be used in multiple projects in software development [4]. Secret management stores the secrets safely and ensures that the transit of the secret is protected. The creation, rotation, and revocation of secrets are also managed by a secret management system. The same secrets might be needed in different projects and phases in the software development cycle and therefore secret management also aims to make the secrets available in all phases of the cycle.

2.2 Infrastructure as Code

Infrastructure as code manages and provisions the infrastructure for computing environment and the configuration of the source code [2]. One of the main benefits of IaC is that it automates the process of provisioning large infrastructure at scale. IaC uses dedicated programming languages, such as Terraform [5], Puppet [6], Chef [7], and Ansible [8], to implement the IaC system.

The biggest challenge for IT organizations to use IaC in their systems is that IaC scripts contain secrets, such as hard-coded passwords and private SSH keys [2]. It has been found that over 9000 hard-coded passwords have been found in a collection of over 60 000 Open Source Software (OSS) IaC scripts [9], which means that secret management in the OSS domain for IaC scripts is predominant. The existence of secrets in Open Source Software IaC scripts is harmful to the security of the computing infrastructures and, therefore the integration of proper secret management is important in infrastructure as code development [2]. However, the lack of crucial practices related to secret management for IaC makes it hard to implement secure secret management integration into IaC systems.

Although secret management tools like Ansible Vault, Chef Vault, and Hiera exist for storing secrets in IaC scripts, there is a lack of standard secret management practices in the field of IaC which can limit the effective use of the secret management tools [2]. Thus, it is crucial to establish a comprehensive set of best practices for secret management in IaC. This can help practitioners better understand how to leverage secret management tools and implement secure IaC scripts.

3 Secret Management in Infrastructure as Code

There are many issues relating to secret management in IaC because not only that the IaC scripts store secrets but the management of creation, storage, rotation, and revocation of the secrets is also difficult. As a result, there is a pressing need for effective secret management practices that can enable secure deployment of IaC scripts.

3.1 Security Threats in Secret Management in Infrastructure as Code

This subsection will identify 4 security smells in IaC scripts that are related to secret management in IaC. A security smell is a recurring coding pattern that is an indication of a security weakness that can potentially lead to a security breach [10].

Empty Password
The first security smell refers to a situation where an empty string is used as a password. An empty password is different from using no password at all since 'Empty Password' occurs when an empty string is assigned to an attribute or variable that is related to passwords. An empty password is not considered a hard-coded secret since the value must be a string of length of one or more to be considered a hard-coded secret [10].

Hard-Coded Secret
Hard-coded secrets reveal sensitive or secret information and the three most common types of hard-coded secrets in IaC scripts are usernames, passwords, and private cryptographic keys [9].

Use of HTTP Without TLS
This smell is about using the HTTP protocol without using Transport Layer Security (TLS). TLS provides protection against various attacks, such as man-in-the-middle attacks, and is therefore essential for maintaining a secure system [10].

Use of Weak Cryptography Algorithms
Using weak cryptographic algorithms to encrypt sensitive information can pose a security threat as they are vulnerable to attacks like the collision attack [10].

3.2 Recommended practices for Secret Management in Infrastructure as Code

This subsection will present and discuss the recommended practices for addressing the four security smells related to secret management in IaC.

Hard-Coded Secret

One of the main concerns in secret management for IaC is the creation, rotation, and revocation of hard-coded secrets. To address this issue, secret management tools like Ansible Vault, Hashicorp Vault, and Hiera are commonly recommended [2]. It is important for developers to have knowledge on how to effectively implement the security features offered by these tools. Each tool also has its own set of recommended best practices for configuring them in a safe and efficient manner.

Ansible Vault is a secret management tool that can be used for password management in IaC scripts [2]. Ansible Vault one can encrypt variables and files with a password which will create a file called 'vault' and therefore the IaC script does not include secrets anymore as plaintext. The first recommended practice for Ansible Vault is that the vault files must not be committed to a version control system (VCS) and therefore it is recommended to implement a system that will notify the developer if a vault file is being committed to a VCS. Ansible Vault also has command line utilities that help the programmers to provide the Ansible Vault passwords for the vault file, otherwise the password needs to be given as a file. By using the command line tool there are no risks that the file containing the Ansible Vault password will end up in a VCS.

Hashicorp Vault is a tool that uses encrypts and decrypts data with an encryption key [2]. The process of decrypting data is called unsealing in Hashicorp Vault and there are two practices that are recommended to use in order to accomplish appropriate unsealing. Hashicorp Vault uses Shamir's Secret Sharing principle for unsealing, which means that the master key is split into multiple pieces. The developer needs to provide the pieces manually, which is an error-prone and lengthy process, and therefore it is advised to use the 'autoseal' tool that is provided by Hashicorp Vault. The 'autoseal' automates the process of providing the information needed to unseal the data. It is also recommended to use multiple vault servers so that the unsealing process is not susceptible to the single point of failure.

Hiera is a secret management tool that can be used to manage secrets in

Puppet scripts [2]. However, the downside of using Hiera is that it affects the readability of Puppet scripts and therefore it is recommended to place secret data at the appropriate hierarchy: the secrets that are used in a few Puppet scripts should be at the top of the hierarchy, and secrets that are used in many Puppet scripts should be at the bottom of the hierarchy.

There are also recommended practices for managing the secret management tools that manage hard-coded secrets [2]. Adequate directory structure and dedicated naming conventions are important for the maintainability and readability of managing secrets with the secret management tools. By applying access control policies the developers can restrict access to storing and reading secrets. Excessive encryption can lead to maintainability issues and therefore it is recommended to prioritize the secrets that need to be encrypted instead of encrypting all the data in IaC scripts. It is also important to separate the secret management environments in a way that the separation is based on grouping the secrets by a certain characteristic. Lastly, limiting the authentication attempts for logging into the secret management tools and decreasing the authentication duration minimizes the impact of unauthenticated access to the hard-coded secrets.

Empty Password

The use of an empty password is related to secret creation. Empty passwords are strongly advised against and strong passwords should be used instead [10]. In addition, the passwords need to be rotated to gain more security against unauthorized users [2]. Password rotation helps to minimize the exposure if a password is leaked or reused.

Use of HTTP Without TLS

The use of HTTP without TLS is related to managing certificates in secret management. The transit of the data between the IaC compiler and the secret management tool should be secured in end-to-end encryption [2]. It is recommended to use HTTP with TLS [2], and some tool vendors offer resources that help developers to set up HTTP with TLS [10].

Use of Weak Cryptographic Algorithms

The use of weak cryptographic algorithms is related to the secret creation in secret management. To address this problem, developers can refer to the list of recommended cryptographic algorithms provided by the National Institute of Standards and Technology [11] [2].

4 Discussion

Many secret management tools discussed in this seminar paper encrypt the data in IaC scripts with a key or a password, While this approach effectively addresses the issue of hard-coded secrets in IaC scripts, it creates a new challenge of managing the keys and passwords required for the secret management tools in IaC. Thus, to achieve complete management of hard-coded secrets in IaC scripts, some practices should address the password and key management of the secret management tools in IaC scripts. Considering that IaC aims to automate manual-intensive tasks of system administrators, it would be reasonable to automate the password and key management for secret management tools at some level in IaC scripts.

Evaluating the effectiveness of recommended best practices for managing hard-coded secrets in IaC scripts can be challenging without prior experience using the specific tools discussed in this paper. While each tool has its own set of recommended practices, utilizing them can bring about various management issues, such as reduced readability, complex decryption processes, and the need to separate secret management files from version control systems. Despite these challenges, secret management tools effectively addresses the issue of having hard-coded secrets in IaC scripts.

Other recommended practices related to managing secret management tools can greatly benefit the maintainability and security of IaC scripts. For example, namespace collision is a known issue in IaC scripts [2] [12], and by following good folder organization practices and consistent naming conventions code maintainability and debugging can be improved. Additionally, excessive encryption can cause maintainability issues, which is important to be aware of when designing a secret management system for IaC. Other recommended practices, such as access control, separation of secret management environments, limiting authentication attempts, and reducing authentication duration, are also beneficial practices for managing the secret management tools. As tool vendors continue to develop their products, it's likely that they will include additional features to address secret management issues in IaC.

The issue of having an empty password in IaC scripts can be solved with a strong password. However, there are other secret management practices that are considered as good practices that relate to secret creation, for

example, password rotation is a highly recommended practice for secret management in IaC since it can minimize the exposure of secrets in case passwords are leaked or reused in large-scale systems.

The issue of using HTTP without TLS is addressed by utilizing resources that the tool vendors provide regarding this issue but otherwise, there are not any other recommended practices relating to this issue. The proposed best practice does not solve the issue of using HTTP without TLS completely and therefore other practices are needed in order to achieve end-to-end TLS to the secret transit in IaC. One possible approach is to investigate the security functions and resources offered by tool vendors to ensure secure secret transit before selecting a vendor for a project. It is recommended to use end-to-end TLS in production and development environments in IaC, although it is unclear from the two main sources reviewed in this paper how this can be accomplished in IaC.

Using weak cryptographic algorithms in secret creation in IaC can be avoided by employing encryption algorithms that are recommended by the National Institute of Standards and Technology [11] or other trusted source. Additionally, other recommended best practices discussed in the literature that relate to secret creation and cryptographic algorithms include prioritized encryption and secure cryptographic key management practices.

Overall, the 12 practices for secret management in IaC [2] covered the 4 security smells identified in this seminar paper, but the quality and quantity were not consistent across the 4 security smells related to secret management in IaC. These 12 practices touched upon various aspects such as access control, folder organization, and secret rotation. It is worth noting that while hard-coded secrets were addressed in detail, issues such as the use of HTTP without TLS received less attention in scientific papers, indicating an opportunity for further research in this area. An another opportunity for further research in this area would be the effectiveness of the recommended practices. Lastly, this seminar paper focused on the recommended best practices for secret management in IaC and therefore it did not cover all of the best practices recommended for IaC in the literature. Consequently, there are other practices that can enhance the quality and security of IaC scripts.

5 Conclusion

To conclude, this seminar paper aimed to explore the issues and challenges associated with secret management in IaC practices. Through an analysis of the literature, this paper discussed the common practices and issues related to secret management in IaC and reviewed two main state-of-the-art resources in this area. The findings of this study include the identification of four security smells related to secret management in IaC, along with recommended best practices and tools for addressing these issues. The paper also identified some gaps in the literature, such as the lack of specific recommended practices for addressing the issue of using HTTP without TLS. Overall, the 12 recommended practices for secret management in IaC covered the four security smells identified in this seminar paper. Future research could focus on further developing common practices for secret management in IaC and investigate the effectiveness of the recommended practices.

References

[1] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A Systematic Mapping Study of Infrastructure as Code Research," *Information and Software Technology*, vol. 108, pp. 65–77, 2019.

[2] A. Rahman, F. L. Barsha, and P. Morrison, "Shhh!: 12 Practices for Secret Management in Infrastructure as Code," in *2021 IEEE Secure Development Conference (SecDev)*, pp. 56–62, 2021.

[3] S. K. Basak, L. Neil, B. Reaves, and L. Williams, "What Are the Practices for Secret Management in Software Artifacts?," in *2022 IEEE Secure Development Conference (SecDev)*, pp. 69–76, IEEE, 2022.

[4] M. Blomqvist, *Secret Management in a Multi-Cloud Kubernetes Environment*. PhD thesis, University of Turku, 2021.

[5] Terraform, "Terraform language documentation," 2021. https://www.terraform.io/docs/language/state/index.html.

[6] Puppet, "Puppet language documentation," 2021. https://puppet.com/docs/.

[7] Chef, "About chef workstation," 2021. https://docs.chef.io/workstation/.

[8] Ansible, "Ansible language documentation," 2021. https://docs.ansible.com/.

[9] A. Rahman and L. Williams, "Different Kind of Smells: Security Smells in Infrastructure as Code Scripts," *IEEE Security Privacy*, vol. 19, no. 3, pp. 33–41, 2021.

[10] A. Rahman, C. Parnin, and L. Williams, "The Seven Sins: Security Smells in Infrastructure as Code Scripts," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 164–175, 2019.

[11] E. Barker *et al.*, "Guideline for using cryptographic standards in the federal government: Cryptographic mechanisms," *NIST special publication*, pp. 800–175B, 2016.

[12] I. Kumara, M. Garriga, A. U. Romeu, D. Di Nucci, F. Palomba, D. A. Tamburri, and W.-J. van den Heuvel, "The do's and don'ts of infrastructure code: A systematic gray literature review," *Information and Software Technology*, vol. 137, p. 106593, 2021.

The Biases of Algorithms

Murali Abinaov
muraliamudha.abinaov@aalto.fi

Tutor: Rannisto Antti

Abstract

This article provides a review of literature surrounding algorithmic bias, its origins, effects and influence, and strategies for mitigating it. The main emphasis is on assessing the social power of algorithms and the harm that arises from bias, as well as how human-centered algorithm design can minimize biases.

KEYWORDS: ALGORITHMIC BIAS, SOCIAL HARM, SOCIAL POWER, ALGORITHMIC POWER

1 Introduction

Algorithms have become a highly influential aspect of modern technology. Although not a new phenomenon, their widespread use by a plethora of websites and social media platforms, including Amazon and YouTube, has led to extensive research on the topic. Recent years have even brought the role that algorithms play in the lives of people, into the public sphere of consciousness. For example, YouTube's video recommendation algorithm has been the focal point of a multitude of discussions. These ranging from the need for creators to change the format, style, or even subject matter of content to be promoted and avoid being suppressed, to the algorithm's

potential to guide viewers towards increasingly extremist content [3, 11]. A common concern associated with algorithms is their potential to have biases.

Algorithms can be understood as a series of steps used to solve a mathematical problem, consequently it is not inherently clear how an algorithm could develop a bias [8]. The most prominent source of bias for an algorithm originates at the intersection of big data input, and the black-boxed internal logic of the algorithm itself [9]. Due to the sheer scale that many of the major social media algorithms operate at, vast amounts of data must be processed to provide the optimal solution within the capability of the algorithm. Within this data, correlations may be found that society collectively recognizes as morally problematic. These correlations may be caused by poor training data with an inherent bias itself, or data that was irrationally interpreted when taken out of context [5]. Nevertheless, these correlations may be incorporated into the decision-making logic and obfuscated by a lack of transparency derived from proprietary knowledge and the complexity and scale of the systems themselves [4].

This paper reviews recent literature on the dangers of algorithmic bias and some proposed solutions for alleviating the potential risks.

Algorithms can alter the perception of individuals and groups in society via the intentional and unintentional promotion certain narratives. Therefore, it is crucial to examine algorithms through a sociological framework. Similarly, several prominent instances of harm resulting from algorithms can be analyzed through the lens of digital zemiology and the social harm framework, given the frequency of cited inequalities in these cases. Finally, the integration of human elements during the design process can significantly diminish algorithmic bias and the harm it can cause in the resulting products.

The paper is structured as follows. Section 2 focuses on the overarching sociological issues due to the bias seen in algorithms. Section 3 examines the social harm framework to see how algorithmic bias detracts from society as a whole. Section 4 details methods to mitigate its propagation and effects. Finally, section 5 concludes the paper with a brief overview of the points indicated throughout with some further analysis.

2 The Sociology of Algorithms and Power

David Beer, a professor of Sociology at the University of York, listed in his paper, "The social power of algorithms", four key areas which are primary vectors that enable algorithms to effect power on society [2]. The areas are: questions on agency, how algorithms alter decision-making, "the politics of algorithmic sorting, ordering and prediction" which in part refers to what information is presented and what is obscured by the algorithm, and last the sense of trust people tend to have in algorithms due to their perceived neutrality, objectivity, or rationality. It should be noted that the author himself clarified that this short list is not all encompassing, rather it represents some of the most significant factors for algorithmic social power.

2.1 Algorithms and Agency

Algorithms are often attributed with their own type of agency as they make decisions in situations where there is no one definitive 'correct' answer [13]. One example of such a scenario is when a news feed algorithm acts as an editor, gate-keeping which articles are displayed to users by determining the most relevant news. Since relevance is largely subjective, yet the non-sentient algorithm's decision-making process still leaves an impact on users, the agency of the algorithm is demonstrated. However, despite having their own agency, algorithms do not simply supplant all human agency in interactions, instead they intertwine with it [2]. Aspects of the human side appear in the creators of the algorithm and their designs, which greatly influence the functionality and results of the algorithm. Furthermore, users can assert their own agency by utilizing multiple news feeds or using the feed as a starting point to search for further information.

2.2 Decision-making

While algorithms are entirely capable of autonomous decision-making, as previously mentioned, they are also frequently utilized by various actors to assist in their decision-making processes [2]. Companies often rely on algorithms to filter resumes for job positions. Despite the fact that the algorithm does not have the final say, it still impacts the decision-making process and may even alter the outcome if a highly qualified candidate has

a resume that the algorithm cannot read. Institutions, organizations, and governments all employ algorithmic input to make influential decisions, such as for the allocation of public resources, determining which schools receive funding, and selecting communities for reinvestment [10, 6]. An increasing number of decisions, previously made entirely by humans, are now recipient to algorithmic aid. This can conceal, entrench, and systematize possible bias in algorithms, making it near impossible to evade or even identify its existence [2].

2.3 'Politics', What is Normal or Abnormal?

The 'Politics' of algorithms are closely tied to their ability to rank and categorize information [2]. For instance, the same example that was used previously to illustrate the agency of algorithms could be employed here. Alternatively, another example could be the propensity for grouping users in like-minded digital communities that exists among some social media algorithms. The act of connecting individuals to others who predominantly share the same beliefs and values creates an echo chamber, which may intensify people's commitment to their convictions. A case in point is the Facebook Myanmar crisis, where the Burmese population became increasingly incensed at the Rohingya minority of the country. On Facebook, the principal internet platform in Myanmar, the rhetoric became more and more extreme, culminating in the outbreak of violence [14]. Algorithms may also contribute to the perception that certain circumstances are more abnormal than they are in actuality via censorship [12]. Search engines provide an additional example, as they sort the numerous web pages of the internet, rank the results, and channel traffic to sites that cater to them best through search engine optimization.

2.4 The Power of the word, 'Algorithm'

The latter third of Beer's article discusses the evolution of the term 'Algorithm' within the social consciousness. It has come to represent more than its definition as a problem-solving series of steps [8], but instead as something beyond human logic and capability - rational, efficient, precise, and neutral [2]. The perceived objectivity of algorithms and their reduced potential for error compared to human action lends them a certain trust, closer to some subjective 'ideal'. This perceived superiority and reduced infallibility when compared to the labours of man, can be used to perpet-

uate certain truths or to lend weight to an argument. Furthermore, the idea of algorithms as objects greater than the sum of their code can lead to the spread of algorithmic systems in and of itself, as systems are deemed better for their incorporation, irrelevant of any actual substantive gain [2].

3 Social Harm

Social harm, as written by the authors of, "Dynamics of social harms in an algorithmic context", can be embedded, disseminated, and magnified through the application of algorithms and their biases [6]. Zemiology, or the study of social harms, provides a scientific approach to documenting and researching the collective negative effects of an action on society, regardless of intent as is key in criminology. This is a significant distinction to note, as algorithms lack intent, yet they nevertheless can severely harm people at the macro level of structural societal pillars. The aforementioned article presented three case studies which exemplify the harm caused to economic and political pillars, as well as the degradation of trust in institutions. These situations include the Michigan Integrated Data Automated System (MDAS), which attempted to automate the state's unemployment insurance claim review process; the 2010 financial market flash crash, which was largely caused by automated stock trading algorithms; and the Cambridge Analytica (CA) scandal, in which the company purchased user data from Facebook to target advertising for multiple major elections worldwide [6].

3.1 Political Harm

Algorithms have the potential to increase political polarization, in addition to manipulate elections, as was the case with CA [6]. One source of political harm was demonstrated in the previous section, as the ability of some algorithms to define the normal and abnormal. This is done through the silent grouping of users into echo-chambers or the misguided censoring of potentially problematic content which may limit legitimate political discourse [14, 12]. The Cambridge Analytica situation attempted to nudge elections in order to orchestrate an outcome that was deemed materially or culturally beneficial to the clients who hired the firm. This interference provides external and internal actors a platform to support political tides

that may have remained minority opinions without such aid. Elections such as Brexit or the 2016 Trump Presidential Campaign could have had similar results regardless of the involvement of CA, but simply the fact its interference occurred creates reverberations that eliminate trust and spark apathy amongst the populace [6]. MDAS, likewise served as evidence of the failures of government, which manufactured distrust in the government and potentially ideological shifts to those affected.

3.2 Loss of Trust

Trust in institutions is a vital component of social cohesion in developed countries. Without trust, social disengagement and tribalism can occur. Algorithms can contribute to the erosion of trust when they are implemented poorly into the systems that shape the lives of people. MDAS, for instance, had a higher percentage of false positives post switching from human-administered to automated due to an algorithmic bias, as far more unemployment insurance claims were falsely flagged as fraudulent [6]. Innocent individuals were suddenly deemed as criminals in the eyes of the system, with limited avenues to seek human oversight to resolve the error. The 2010 flash crash similarly reduced trust in the stock market due to the constant fluctuations and sudden shifts caused by algorithms trading with each other, resulting in a greater complexity and risk of financial losses for human investors. The CA situation can also be used to highlight this narrative. Election interference erodes trust in the outcomes of elections and increases tribalism, as supporters of opposite sides of an issue can proclaim the other side has been influenced by an outside party to disregard their arguments, while feeling as if the system is unfairly rigged against them.

3.3 Economic Harm

Algorithms possess tremendous potential to transform various segments of the economy; however, they also have the capacity to inflict financial losses upon both individuals and society at large [6]. The flash market crash of 2010 serves as a poignant illustration of this phenomenon, as the instability caused by algorithms resulted in the savings of numerous individuals being wiped out. In another instance, the MDAS system created significant but temporary savings for the state of Michigan by wrongly accusing unemployment insurance claimants of fraud, leading to devastat-

ing consequences for the affected individuals such as bankruptcy, rental agreement rejections, homelessness, and seizure of income by the state to recover non-existent debts [6]. In the long-term, whatever savings were achieved for the state by these false positives was negated several times over by the subsequent lawsuits.

4 Human-centered Algorithm Design

Human-centered algorithm design (HCAD) refers to the integration of social science techniques into the development of algorithms [1]. Concrete examples of HCAD include the utilization of multidisciplinary teams and the increased involvement of potential users during development [1]. The inclusion of experts from diverse fields among the developers can help mitigate the risk of unintended problems arising from the algorithm due to the wider outlook on potential consequences. Nonetheless, it may be exceedingly difficult for a team to identify all sources of bias or create safeguards against all possible misuse, as people will interact with an algorithm in ways never conceived of by its designers. The participation of various groups during the development process can assist in addressing many of these specific and obscure complications.

Value-sensitive design (VSD) modeling represents another approach for implementing HCAD, but in a more abstract manner. VSD centers on the values of various stakeholders, enabling the identification of specific aspects that demand special attention and preventing the algorithm from compromising those values [7]. Utilizing VSD can be particularly advantageous when the values of multiple stakeholders conflict. In such situations, some level of bias may be inevitable, and it is crucial to determine which bias is least harmful and most fair to stakeholders. Deciding on the extent and type of bias that is fair to incorporate into the algorithm necessitates a careful examination of the consequences through both the lenses of equality and equity. Equality refers to the even distribution of both benefits and drawbacks, regardless of circumstances [7]. In practice, this may entail counter-balancing a bias that provides a slight advantage to a stakeholder in one aspect of an algorithm, with a slight disadvantage offset in another. Conversely, equity differs from equality in that it primarily takes into account the surrounding circumstances of a stakeholder to determine the fairest decision [7]. Achieving the most equitable outcome may necessitate providing some favorable bias towards the party

with unfavourably circumstances, in order to prevent them from being entirely overshadowed.

5 Conclusion and Analysis

The present paper has showcased two frameworks through which the power of algorithms and their biases can be viewed, in addition to some approaches that may be employed to alleviate the negatives. The first framework, explored in section two, is an over-arching social power lens that scrutinizes algorithmic agency and its capacity to alter, blend, and create power. The second, illustrated in section three, is the social harm perspective, where algorithmic biases reinforce or diminish preexisting forms of social power to negatively affect individuals. Finally, techniques are provided to discover, model, and reduce biases and their ramifications during development.

Ensuring HCAD is employed in conjunction with multiple frameworks, including those not discussed here, to determine biases, consequences, and stakeholders will assist in the creation of high quality algorithms that have reduced detrimental impacts on society. To achieve this, some projects may adopt a process similar to an open beta, allowing for the identification and resolution of issues, as well as an exploration of potential user behaviors, both positive and negative. Any potential abuse can then be further analyzed with relevant frameworks to ascertain how stakeholders can be advantaged or disadvantaged, and limit those if deemed necessary.

The articles referenced throughout here are not representative of the totality of literature on algorithmic bias, but rather provide a sample that highlights some significant ideas in the space. Other related areas that have not been addressed include the possibility of biased big data leading to algorithmic biases, or how the recent advancements in AI, such as deep neural network art generators and large language model chat bots, are affected by and perpetuate biases. Avenues for further research include investigating the resistance that potential biases may generate towards algorithms in general, and examining how algorithmic bias can also reinforce bottom-up power dynamics as opposed to the top-down power mostly discussed in this paper and in literature.

References

[1] Eric PS Baumer. Toward human-centered algorithm design. *Big Data & Society*, 4(2):2053951717718854, 2017.

[2] David Beer. The social power of algorithms. *Information, Communication & Society*, 20(1):1–13, 2017.

[3] Lauren Valentino Bryant. The youtube algorithm and the alt-right filter bubble. *Open Information Science*, 4(1):85–90, 2020.

[4] Jenna Burrell. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1):2053951715622512, 2016.

[5] David Danks and Alex John London. Algorithmic bias in autonomous systems. In *Ijcai*, volume 17, pages 4691–4697, 2017.

[6] Malik HM, Viljanen M, Lepinkäinen N, and Alvesalo-Kuusi A. Dynamics of social harms in an algorithmic context. *International Journal for Crime, Justice and Social Democracy*, 11(1):182–195, 2022.

[7] Min Kyung Lee, Ji Tae Kim, and Leah Lizarondo. A human-centered approach to algorithmic services: Considerations for fair and motivating smart community service management that allocates donations to non-profit organizations. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 3365–3376, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Merriam-Webster. Algorithm. In *Merriam-Webster.com dictionary*. Encyclopædia Britannica, Inc., 2023.

[9] David Moats and Nick Seaver. "you social scientists love mind games": Experimenting in the "divide" between data science and critical algorithm studies. *Big Data & Society*, 6(1):2053951719833404, 2019.

[10] Nuno Mota, Negar Mohammadi, Palash Dey, Krishna P Gummadi, and Abhijnan Chakraborty. Fair partitioning of public resources: Redrawing district boundary to minimize spatial inequality in school funding. In *Proceedings of the Web Conference 2021*, pages 646–657, 2021.

[11] Emily Pedersen. "my videos are at the mercy of the youtube algorithm": How content creators craft algorithmic personas and perceive the algorithm that dictates their work. Technical report, University of California, Berkeley, 2019.

[12] Uwe Peters. Algorithmic political bias in artificial intelligence systems. *Philosophy & Technology*, 35(2):25, 2022.

[13] Zeynep Tufekci. Algorithmic harms beyond facebook and google: Emergent challenges of computational agency. *Colo. Tech. LJ*, 13:203, 2015.

[14] Neriah Yue. The" weaponization" of facebook in myanmar: A case for corporate criminal liability. *Hastings LJ*, 71:813, 2019.

Analysing the security properties of the APT package manager

Niko Vänttilä
niko.vanttila@aalto.fi

Tutor: Jacopo Bufalino

Abstract

Package managers have huge impact on the security of the clients individual machines and supply chain overall. Small bugs in packages and libraries can spread to wide use causing a lot of harm due to wide distribution of packages. This paper targets to analyse the security properties of the APT package manager.

KEYWORDS: APT, security, cryptography

1 Introduction

Package manager is a software that allows clients to easily manage packages and libraries on their machines. It automates tedious tasks such as install, update and configure of packages, typically softwares and libraries. Package managers are divided on two basic categories based on the intended use. APT is built for Debian-based Linux distributions and intended to install binaries. Other example of package manager is PIP for Python which manages library dependencies in development environment [1].

Package managers have huge impact to operating systems security since they are in control of many packages on the machine. They usually have

additional privileges such as superuser which increases the potential harm to machine. Package managers are vital part of the supply chain security, since they are used in many stages among the developers and consumers. Therefore, package managers should contain reliable security properties which ensure that the packages are taken to use in a way they are intended to.

This paper targets to analyse APT package managers security properties. Section 2. presents the basics of package manager and its main security properties. Third section analyses specifically the security properties of APT package manager including ways to attack and how APT protects against them. Section 4. discusses the conditions of an successful attack and open vulnerabilities among the APT. Last section provides a conclusion.

2 Background

2.1 Package managers

Package manager controls the packages on the operating machine. Such tasks include installing, updating and deleting of packages. These packages are downloaded and installed from the package repositories. Packages provided by the repositories come typically in similar formats consisting of necessary files and metadata. Metadata contains information about the package itself and dependencies that are needed for this package to operate. Most of packages formats contain signatures whereas others do not support this feature [2]. Signatures which are based on public key cryptography are the main security property of repositories and packages.

A package repository is the place from where the manager downloads necessary data and it is typically an HTTP or FTP server. Before downloading and installing actual packages, manager downloads the root metadata which contains location and signatures of the each package and their metadata. With the package metadata and the signatures, the package manager is able to verify the validity of the package and download the actual package metadata and performs dependency resolution [2]. Figure 1. shows the common layout of the package data in repository. Root metadata contains all the individual packages needed. In APT, root metadata

contains its own signature but it is not used in every package manager. As the figure shows, the package metadata gives information about the individual package, which is needed to perform dependency resolution. Hashes in the figure refer to checksums which are discussed later in the paper.

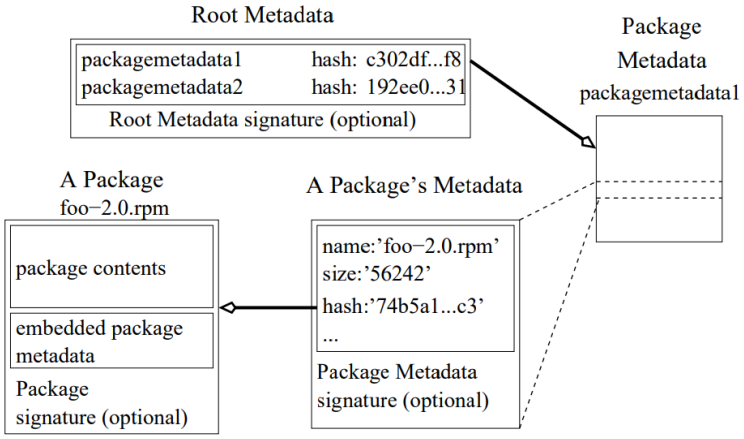


Figure 1. Figure 1. Repository Layout [2].

Purpose of dependency resolution process is to automatically install or locate all the necessary dependencies and the main software. Package manager searches the machine to locate all the dependencies that are needed before the main package files can be installed. Root metadata provides the repository which contains the needed dependency in case it is not in the machine already. [2]

The package repositories are divided into main repositories and mirrors. Purpose of mirrors is to distribute repositories to more than one server. Characteristics of main repository is that it is the one that is maintained by the administrators. This means that all the modifications to repository are done only to this main one. A mirror is a copy of the main repository, so it contains all the same data as the main one. When main repository is updated, the mirrors are synced via some tool [2]. Mirrors can be divided into public mirrors that are available for anyone, and private mirrors which are typically available to some specific organisations.

2.2 Security properties

One of the most significant factors in package manager security is how the client communicates with the repository. Package managers should use secure communication protocol such as HTTPS rather than basic HTTP. HTTPS is using HTTP but with a different default port (443) and an addi-

tional TLS encryption/authentication layer between HTTP and TCP [3]. Communication between the client and repository should be secured in a way that data flow cannot be tampered.

Package manager should check the integrity of the downloads to ensure that files haven't been modified [2]. One of the most used way to check this is to calculate checksum between the downloads and check that it matches. Cryptographic checksum is a way to determine whether changes have been made to some file. It calculates a secure hash from data and if changes have been made to the file, it produces different hash [4].

To support integrity checking, package manager should perform authenticity checking in case of modified checksum. This can be done via public-key cryptography. Each checksum is signed with private key by the administrators and the manager can verify that the downloaded file can be trusted [2].

3 Security of the APT package manager

3.1 Attacks against package managers

Successful attack against package manager may cause attacker to crash or control clients computer. Attacks can be conducted in all stages of package installation process including resolution, verification, fetching and installation. These attacks can be divided into families such as package manipulation, denial of service and code injection [5].

In package manipulation the process of package installation is corrupted in a way that the authenticity and integrity of artefacts are not verified or if the resolution of packages is done in an unpredictable manner [5]. This can happen when the transmission is not using SSL/TLS to secure communication or the validity of their certificate is not verified. Man in the middle attack (MITM) is a known package manipulation example which allows attacker to respond to requests made by a package manager [6]. Freeze attack is an example of MITM attack where the attacker is able to provide its own package in place of the correct one. In replay attack the package manager downloads the older version of the package. This way the attacker can exploit the possible vulnerabilities which no longer exist in the newest package version [2]. This only works when installing the package first time and not when updating. On the other hand, in freeze

attack the attacker provides metadata that is not up to date. In this way the client is not able to detect updates and might use old version of a package with vulnerabilities. With ability to rewrite metadata, attacker is able to provide extraneous dependencies. This means that the along with necessary dependencies, some additional dependencies are installed which could contain vulnerabilities. One of the different ways to exploit MITM attack is to cause package manager to download endless stream of data. This causes disk to fill up and possibly crash the clients computer [2].

In denial of service the attacker aims to disable the computational resources or access to the clients computer. Attacker may exploit the internet connection by flooding the client with multiple requests. Denial of service attack can be either a single-source attack, originating at only one host, or a multi-source, where multiple hosts coordinate to flood the victim with a barrage of attack packets [7]. Other way to exploit the package manager which do not limit the amount of data extracted from compressed package artifacts is a zip bomb. It is a malicious file which can overflow clients memory or disk space, or by putting excessive load on it's CPU and eventually crash a computer [5].

Code injection is an attack where attacker tries to inject malicious command executions and especially shell commands on clients computer. Package managers use typically git repositories which are accessed via command line. This way attacker may provide malicious URL for git repository which can lead to command injection [5].

3.2 How APT protects system against attacks

Various package managers have different focus areas when considering the security of the system. APT package manager focuses on securing the repository metadata rather than signing packages [8]. As mentioned before the repositories store packages, package metadata, and the root metadata. In different package managers the root metadata file is called in different names even though the content is similar. In APT it is called *Release* file [9].

Verification of signatures is based on public key cryptography which contain one public key and one private key. Signature is encrypted with public key, and it can be decrypted with private key. APT uses gpg as the OpenPGP implementation to verify signatures [10]. APT contains a program apt-key which manages the keyring of OpenPGP keys and it is used

to show, add and remove keys from the keyring. The keyrings are stored in files in directory */etc/apt/trusted.gpg.d* [10].

The *Release* file which is stored in archive is updated every time some package changes. It contain the checksums of other files in the archive. *Packages* file contain each checksum of individual package. This way APT can compare two checksums and verify that the package to be downloaded is the correct one. It can also compare the checksum of individual package against the content of the *Packages* file [10]. However, the APT cannot still verify that the *Release* file is valid and secure.

Secure-APT was created to overcome issues related to security of *Release* file and the properties of it are used in the latest versions of APT [8]. Alongside *Release* file, the repository ships OpenGPG signature for Release file and it is stored in file called *Release.gpg*. Security of the APT depends on a *Release.gpg* file, which signs a *Release* file, and of APT checking that signature using gpg. In order to check the signature, it has to know the public key of the person who signed the file. These keys are kept in APT's own keyring */etc/apt/trusted.gpg.d*, and managing the keys is where secure Secure-APT comes in [10].

```
/etc/apt/trusted.gpg.d/google-chrome.gpg
----------------------------------------
pub   rsa4096 2016-04-12 [SC]
      EB4C 1BFD 4F04 2F6D DDCC  EC91 7721 F63B D38B 4796
uid           [ unknown] Google Inc. (Linux Packages Signing Authority) <linux-packages-keymaster@google.com>
sub   rsa4096 2021-10-26 [S] [expires: 2024-10-25]
```

Figure 2. Figure 2. Example of key in keyring.

Figure 2. shows an example of key in the APT's keyring. It shows the location of the key in */etc/apt/trusted.gpg.d* directory. The field *pub* shows the key fingerprint and the expire date tells how long the key is valid [10].

4 Discussion

4.1 Conditions for a successful attack

Conditions required for an attacker can be divided to three levels which each can cause increasingly damaging attacks [8]. Firstly, the attacker is able to impersonate a repository and launch basic attack. It can be obtained with MITM attacks and control of repository or mirror. In MITM attack the attacker is able to interfere with communication between the client and repository [6]. When using insecure protocol such as HTTP

the client is vulnerable to such attacks. However, impersonating a repository does not necessary mean that the attacker is able to crate their own packages on repository.

In second condition attacker is able to sign metadata causing more damaging consequences. When attacker is in control of repository it may be able to tamper the metadata. This way attacker can cause client to add their own key to the clients keyring making it vulnerable [8]. The most dangerous way for attacker to cause harm is to sign packages. This way the attacker may be able to sign their own arbitary packages. However, since APT concentrates on signing the repository metadata, there is no package keys to sign. Therefore signing the metadata is more dangerous.

Attacker can also bypass security with uploading arbitary packages to repository in legit way. If attacker is able to compromise developers key, and the package is correctly formatted it can be uploaded to APT distribution. Reasons why this is doable is that the whole process is automated and any developer can upload updated packages. There are thousands of developer keys available and some of them are less secure (short keys) and even decades old [8].

4.2 Open vulnerabilities

APT assumes that the downloaded file size can fit in a C unsigned long. On 32 bit architectures it can download up to 4GB data but on 64 bit architecture it tries to download over 10TB data [8]. Because of this, the system is vulnerable to endless data attack. It could cause multiple problems such as not getting package updates, and consuming high amount of disk space and CPU which can result big issues on the machine [2].

APT is quite lazy in some situations relating to metadata. For example, it does not check the date of package which could refer to past or even future. Even though the metadata is signed to avoid tampering, there is no protection against replaying old metadata [8]. Additionally APT overwrites current metadata with the files it is downloading. Because of this, it cannot check the past state of the repository.

5 Conclusion

This paper has reviewed the security properties of a APT package manager. APT is a system-wide package manager that controls the packages

on Debian based Linux distributions. Security of APT is based on securing the repository metadata rather than signing individual packages. Signatures are based on public key cryptography. APT uses latest security properties from Secure APT, which as initially created to overcome issues related to *Release* file. There are multiple ways to attack package managers including package manipulation, denial of service and code injection.

References

[1] A. Athalye, R. Hristov, T. Nguyen, and Q. Nguyen, "Package manager security," tech. rep., 2014.

[2] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "A look in the mirror: Attacks on package managers," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 565–574, 2008.

[3] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 78–81, 2009.

[4] F. Cohen, "A cryptographic checksum for integrity protection," *Computers & Security*, vol. 6, no. 6, pp. 505–510, 1987.

[5] A. M. Bos, "A review of attacks against language-based package managers," *arXiv preprint arXiv:2302.08959*, 2023.

[6] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[7] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 99–110, 2003.

[8] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, "Package management security," *University of Arizona Technical Report*, pp. 08–02, 2008.

[9] "Debian repository format," *Debian Wiki*, 2022.

[10] "Secure apt," *Debian Wiki*, 2022.

Assessing Container Security: An Overview of Best Practices and Popular Tools

Nimer Amol Singh
nimer.singh@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

Containerization has transformed how software is developed and deployed, but the dynamic nature of containerized environments offers new security issues. This paper describes the capabilities, limitations, and use cases of four popular container security tools: Docker Bench, Clair, Falco, and Anchore. Docker Bench evaluates container security setup, Clair checks container images for known vulnerabilities, Falco detects suspicious activity in containerized environments, and Anchore detects vulnerabilities and policy violations in operating containers. To mitigate potential security concerns, effective container security necessitates a complete strategy that includes proactive security measures, continual monitoring, automated remediation capabilities, and regular assessments and training activities.

KEYWORDS: docker, security tools, static tools, dynamic tools, vulnerability analysis, Docker Bench, Clair, Falco, Anchore

1 Introduction

Docker has emerged from decades of evolution of microservices-based architecture. The benefits it presents, such as scalability, flexibility, and

cost-effectiveness, make it an attractive option for organizations of all sizes [1]. Despite these benefits, the shift to a cloud-based architecture has introduced new security challenges [2]. The annual survey of the Cloud Native Computing Foundation emphasizes the significance of security vulnerabilities in the context of container security. 33% of the 2063 participants questioned for the report expressed concern about these vulnerabilities [3] [4]. This stresses how crucial it is to put in place strong security mechanisms in containerized environments to reduce potential hazards. Given the crucial role Docker plays in managing sensitive data and critical systems in the cloud, it is important to ensure it's safety and effective deployment.

This paper focuses on the different approaches and tools utilized to test Docker development and deployment security. The paper explores various techniques, including manual and automated testing methods using static and dynamic analysis tools. By comparing different approaches to solve the problem, the paper will provide a guide for effective security testing of applications developed and deployed using Docker as well as safe development practices.

2 Background

It is important to consider the various components that can be selected for evaluation during the process of vulnerability assessment in Docker configurations and deployments. The various targets that can be studied, such as the Engine, Images, Containers, Hub, and Compose will be covered in this section [5]. For a Docker deployment to be secure and to lower the risk of security breaches, it is essential to understand the various targets and how they may be examined.

2.1 Docker Engine

The basic component of Docker is known as the Docker engine, which facilitates container management. It comprises of a server-client architecture that communicates via a representational state transfer (REST) application programming interface (API). The server (commonly known as the Docker daemon or dockerd) is responsible for building, running, and dispersing containers; in contrast, this functionality lies with the client referred to as the docker command line interface (CLI) which interacts

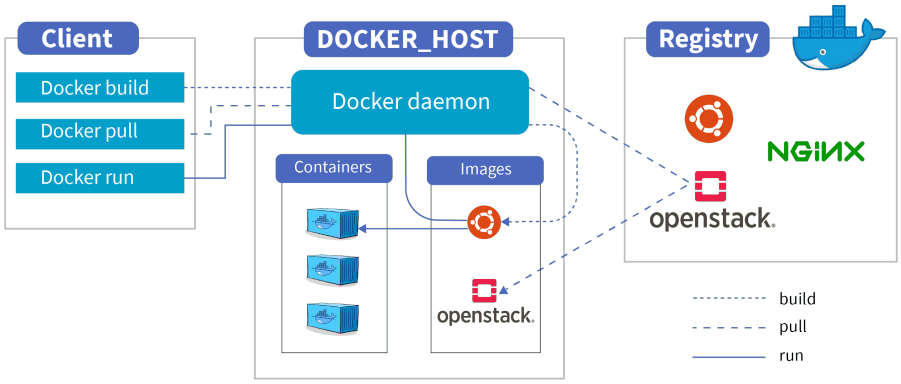


Figure 1. Docker Architecture

with the above mentioned.

2.2 Docker Images

As the foundation for containers, Docker images become significant. These are packages that have the ability to run code efficiently and contain all elements essential in running a program including libraries, and dependencies alongside codes. They are lightweight components, being able to stand entirely on their own without any support system. Images are built using Dockerfiles, which are a set of instructions that define how to build the image.

2.3 Docker Containers

Containers are lightweight, portable, and self-contained environments that have all the requirements to run an application. They package code and dependencies so that the application can run quickly and reliably in different environments. Containers are created using Docker images and can be easily shifted between different development, testing, and production setups.

2.4 Docker Hub

Docker Hub is a public registry and repository which hosts Docker Images. It is used for developing, storing, and distributing images with other developers. Similar to Github, it offers version control systems, frequent updates, and free sharing of images for unrestricted use. It is a crucial

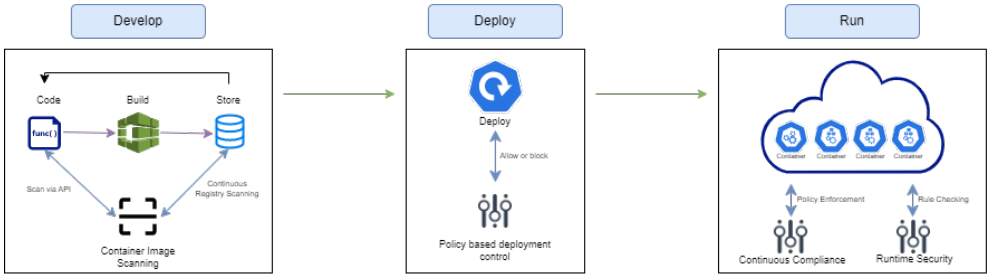


Figure 2. Docker Security Lifecycle

part of the Docker ecosystem for managing images and deploying applications as it provides the capabilities to authenticate users and push and pull images from the registry.

2.5 Docker Compose

The final component of Docker is the Docker Compose. It is a tool that enables a developer to define and deploy multi-container applications. A developer can specify the configuration of the containers in a single YAML file, for ease of use and development. These are mostly used to implement the declarative nature of Docker by describing the end result state of the container instead of writing procedural instructions for how to build and deploy an application.

3 Security Guidelines

With Docker, such as all tools, security guidelines must be followed in order to ensure that the developments and deployments transpire in a safe environment. This section mentions a few of them which must be followed for Docker deployments and the repercussions of not implementing the same [6] [7] [8].

3.1 Updates or Grenades

Every day, new vulnerabilities are detected in the existing software. To tackle these, companies release patches and updates. Docker Inc. [9] follows the same practices to prevent an attacker from using escape vulnerabilities and kernel exploits. Updates not only help with securing such vulnerabilities but also help in improving performance.

3.2 User not loser

When configuring a Docker container, it's crucial to ensure that the user has low privileges to avoid escalation attacks. This step is essential to reduce the significant risk in the event of a security breach.

3.3 Socket or Rocket

Networks act as an expressway for attackers, and as such, exposing the docker daemon socket can potentially expose unencrypted and unauthenticated direct access to the Docker daemon. This can lead to anyone from the public net having access to the daemon, which is not an ideal condition. Hence, sockets should not be exposed to the outside world.

3.4 Disable inter-container communication and limit resources

The default setting allows containers to communicate with each other, using which malicious containers can interfere with the correct execution of others. Moreover, malicious containers can also use or block resources for the same purpose. Ensuring limited communication between containers is a reliable way to avoid such mishaps.

To secure a container, tools like AppArmor can be used by configuring it with specific profiles that grant access only to necessary resources like network access, file permissions, and more. These profiles can be set to either block access to disallowed resources or only report violations [10].

4 Tools

In order to ensure secure and reliable containerized environments, it is essential to follow industry-standard guidelines and best practices for container security. To help achieve these goals, various container security tools can be used to assess, identify, and remediate security risks in containerized environments. By utilizing these tools and adhering to recommended security guidelines, organizations can reduce the likelihood of security breaches and better protect their containerized applications and data. This section details a brief overview of some of the tools used for such purposes. When it comes to security tools for container environments, there are two main types: static and dynamic. Static tools, like

Docker Bench [11] and Clair [12], assess containers and images for known vulnerabilities and misconfigurations without actually running them. Dynamic tools, such as Falco [13] and Anchore [14], monitor running containers and detect suspicious behavior in real time. While static tools are useful for preventing known vulnerabilities from being introduced, dynamic tools provide more comprehensive protection by identifying and responding to actual threats as they occur.
These have been chosen because of their popularity, the depth of analysis each does, and the importance to key factors such as Open Worldwide Application Security Project (OWASP) [15] guidelines and Center for Internet Security (CIS) [16] benchmarks.

4.1 Docker Bench

Docker Bench is an open-source project, which aims to check the Docker configurations against dozens of common best practices around deploying Docker containers in production [17]. The project consists of an automated script, which tests the given container(s) according to the latest CIS Docker benchmark.

The tool itself is meant for static analysis and is packaged as a docker container that runs with numerous privileges on the host, such that it shares the filesystem, process ID (pid), and network namespaces.

4.2 Clair

Clair is another open-source project used for static analysis of containerized applications including OCI and Docker [18] [19]. It parses the image contents and reports the vulnerabilities affecting the contents. It is mostly used for the official base containers of Ubuntu, Debian, RHEL, Suse, Oracle, Alpine, AWS Linux, VMWare Photon and Python but it can also be used to scan custom container images not hosted in these registries.

Clair uses ClairCore library as its base, and the application itself can be considered a service wrapper for the library. The application is divided into three parts, namely indexing, matching, and notifications. The tool utilizes libraries such as National Vulnerability Database (NVD), Common Vulnerabilities and Exposures (CVE) database, and other open-source databases to keep an updated list of threats.

4.3 Falco

Falco is an open-source dynamic analysis tool maintained by CNCF. It parses the system calls made at runtime inside the containers and cross-checks them against pre-defined as well as user-defined rules. The rules are designed to detect a wide range of security violations, including unauthorized file access, execution of known vulnerabilities, and breakout attempts. Upon a violation, the tool raises alerts for the developer to analyze [20].

Among many others, Falco is mainly used to detect privilege escalation, namespace changes, Read/Write to system directories, unexpected network activity, and system process calls such as sh, csh, bash, and ssh [21].

4.4 Anchore

Another open-source project, Anchore provides a centralized service for the inspection, analysis, and certification of docker images. It is capable of running in a standalone environment, or inside a container orchestration (e.g. Docker Swarm and Kubernetes) to automate and rectify the shortcomings of the application/images being analyzed.

The tool itself is hybrid, providing both static and dynamic analysis possibilities. The static analysis of the Anchore engine scans for vulnerabilities in software components, including OS packages, application dependencies, and libraries while in a dynamic setting, it monitors running containers in real-time to detect and prevent attacks such as privilege escalation, file tampering, and network scanning. It keeps an updated list by scanning NVD, CVE, Vendor advisories, and Open Policy Agents (OPA) as well as allowing users to create custom rules for detection.

5 Analysis

This section focuses on the features, advantages, and disadvantages of each tool.

5.1 Docker Bench Security

Advantages

1. It performs automated scanning of Docker images for known vulnerabilities in software components.
2. The tool provides a standardized CIS checklist of security best practices for Docker containers.

Limitations

1. Docker bench only checks security issues related to the configurations and installations. The vulnerabilities within images or containers are not identified by this tool.
2. While the tool does produce a report on the security risks, it fails to provide remediation of those risks. The user must manually address the issues, which can be both time-consuming and may require specialized knowledge.

5.2 Clair

Advantages

1. Clair provides detailed information about each vulnerability, including severity level and recommended fixes.
2. It can be easily integrated into existing container workflows and pipelines.

Limitations

1. Clair can only detect known vulnerabilities present in its database. Hence, new vulnerabilities may not be detected in some cases.
2. The tool is designed to work with Docker images and containers, and provides limited support for other container formats and languages.

5.3 Falco

Advantages

1. Falco performs real-time monitoring of running containers to detect malicious activities.
2. It uses a set of rules to detect suspicious activities and alert security teams to detect known and unknown vulnerabilities.
3. The tool can be integrated with various container orchestration platforms and security information and event management (SIEM) systems.

Limitations

1. Falco requires some tuning and customization to minimize false positives.
2. It does not perform static analysis, so it cannot detect vulnerabilities in container images.

5.4 Anchore

Advantages

1. Combines both static and dynamic analysis techniques to provide comprehensive security coverage as well as detailed information about vulnerabilities and recommendations.
2. Can be integrated with various container registries and continuous integration/continuous deployment (CI/CD) pipelines.

Limitations

1. Requires some tuning and customization to minimize false positives.
2. Can be more complex to set up and use compared to other tools.

Table 1 presents the analysis in a tabular format.

6 Conclusion

In conclusion, it is evident that Docker has gained immense admiration as a tool for developing, initiating, and shipping software. However, dismiss-

Tool	Type	Checklist-Database	Remediation
Docker Bench	Static	CIS Benchmark	Manual
Clair	Static+Dynamic	NVD, CVE, Open Source Databases	Manual
Falco	Dynamic	User defined rules	Manual
Anchore	Dynamic	NVD, CVE, Vendors, OPA, User Defined	Automated with externals

Table 1. Analysis of the four tools

ing the significance of upholding top-notch safety measures could jeopardize the protection and soundness of the Docker containers in addition to their likenesses. Key security practices include using tools, such as those mentioned above for static and dynamic analysis to check vulnerabilities in the design and implementation of an application. It is important to not only do these tests on the applications but also on the images, the system processes, and the CI/CD pipelines.

When it comes to container security tools, all four tools have their own benefits. However, Clair and Anchore beat the other options due to their comprehensive vulnerability scanning capabilities. Clair excels at scanning container images for known vulnerabilities and integrates well with Docker, Kubernetes, and other container orchestration platforms. Anchore, on the other hand, goes beyond image scanning and provides continuous security monitoring and policy enforcement for running containers. Additionally, Anchore integrates with external tools to provide automated remediation, making it a suitable choice for enterprise-scale container deployments. Specific use cases for these tools could include scanning images for vulnerabilities before they are deployed to production, monitoring running containers for security violations, and enforcing security policies across a large number of containers.

With these steps, deploying Docker applications becomes both secure and efficient. These methods cement a protective strategy against potential threats that could harm sensitive information or cause incidents within operations. By adopting such protocols, users ensure their systems have been set up with strong defenses in place before any mishaps occur.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53: 50–58, apr 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721672. URL https://doi.org/10.1145/1721654.1721672.

[2] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[3] CNCF. Cncf annual survey 2022. https://www.cncf.io/reports/cncf-annual-survey-2022/. Accessed: 2023-01-31.

[4] Vijay B Mahajan and Sunil B Mane. Detection, analysis and countermeasures for container based misconfiguration using docker and kubernetes. In *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, pages 1–6, 2022. doi: 10.1109/IC3SIS54991.2022.9885293.

[5] Thanh Bui. Analysis of docker security. *arXiv preprint arXiv:1501.02967*, 2015.

[6] Theo Combe, Antony Martin, and Roberto Di Pietro. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3(5):54–62, 2016. doi: 10.1109/MCC.2016.100.

[7] Aparna Tomar, Diksha Jeena, Preeti Mishra, and Rahul Bisht. Docker security: A threat model, attack taxonomy and real-time attack scenario of dos. In *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 150–155, 2020. doi: 10.1109/Confluence47617.2020.9058115.

[8] Kelly Brady, Seung Moon, Tuan Nguyen, and Joel Coffman. Docker container security in cloud computing. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0975–0980, 2020. doi: 10.1109/CCWC47524.2020.9031195.

[9] Docker Inc. Docker: Accelerated, containerized application development. https://www.docker.com/, 2011. Accessed: 2023-04-10.

[10] Kubernetes. Restrict a container's access to resources with apparmor. https://kubernetes.io/docs/tutorials/security/apparmor/. Accessed: 2023-04-10.

[11] Docker Inc. Docker bench for security. https://github.com/docker/docker-bench-security. Accessed: 2023-04-10.

[12] Red Hat. Vulnerability static analysis for container using clair. https://github.com/quay/clair. Accessed: 2023-04-10.

[13] Sysdig. The falco project. https://falco.org/docs/getting-started/running/. Accessed: 2023-04-10.

[14] Anchore. Anchore engine. https://github.com/anchore/anchore-engine. Accessed: 2023-04-10.

[15] Open Worldwide Application Security Project. Owasp top ten application security risks. https://owasp.org/www-project-top-ten/. Accessed: 2023-04-10.

[16] Center for Internet Security. Cis benchmarks list. https://www.cisecurity.org/cis-benchmarks. Accessed: 2023-04-10.

[17] Aakriti Sharma, Bright Keswani, and Anjana Sangwan. Optimization of docker container security and its performance evaluation. *Journal of Discrete Mathematical Sciences and Cryptography*, 24(8):2365–2375, 2021.

[18] Maximiliano Osorio, Carlos Buil Aranda, and Hernán Vargas. Dockerpedia: a knowledge graph of docker images. In *ISWC (P&D/Industry/BlueSky)*, 2018.

[19] Olufogorehan Tunde-Onadele, Jingzhu He, Ting Dai, and Xiaohui Gu. A study on container vulnerability exploit detection. In *2019 ieee international conference on cloud engineering (IC2E)*, pages 121–127. IEEE, 2019.

[20] Scott Surovich and Marc Boorshtein. *Kubernetes and Docker-An Enterprise Guide: Effectively containerize applications, integrate enterprise systems, and scale applications in your enterprise*. Packt Publishing Ltd, 2020.

[21] Guan-Yu Wang, Hung-Jui Ko, Min-Yi Tsai, and Wei-Jen Wang. Module architecture of docker image and container security. In *New Trends in Computer Technologies and Applications: 25th International Computer Symposium, ICS 2022, Taoyuan, Taiwan, December 15–17, 2022, Proceedings*, pages 661–669. Springer, 2023.

A Survey on Security of Microservices

Parsa Sadri Sinaki
parsa.sadrisinaki@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

This paper presents a survey on the security of microservices, examining both the challenges and solutions associated with this architecture. We categorize the security challenges into seven layers of Hardware, Virtualization, Network, Host, Service, Orchestration, and Configurations. We discuss the specific security challenges in each layer and provide possible solutions. Although microservices offer benefits, such as the isolation of vulnerabilities, they also introduce new challenges, such as increased attack vectors. This paper underscores the importance of robust security practices in a microservices environment and emphasizes the need for academia and industry to invest in research on modern attack vectors. Ultimately, both microservice users and cloud providers can benefit from the insights and solutions provided in this paper.

KEYWORDS: Microservice, Serverless, Cloud Computing, Security

1 Introduction

Microservices are popular software architecture comprising multiple independent modules with well-defined network interfaces [10]. Deployed in the cloud using containers, which can be easily scaled based on demand.

Microservices are increasingly used in critical domains, such as Fog Computing [21], Internet of Things [17], and 5G core network functions [2], necessitating enhanced security. Their inherent exposure to internal services, however, raises potential security concerns.

This paper reviews the latest security challenges faced in microservices hosted on cloud and serverless computing platforms, while also providing solutions for these challenges. The structure of the paper is as follows. Section 2 overviews microservices. Section 3 presents the significant security challenges faced in new domains, while Section 4 describes some solutions to these security issues. Section 5 provides a discussion and analysis of the topic. Finally, Section 6 provides concluding remarks.

2 Microservices

2.1 Definition

The phrase microservice was first used in 2011 [7]. It has various definitions, with Lewis and Fowler describing it as a development approach that creates small, independently deployable services running in separate processes and communicating through lightweight mechanisms [7]. These services, have minimal centralized management and support diverse programming languages and data storage technologies. While the microservice architecture resembles the older service-oriented architecture (SOA) pattern, it differs in that microservices are smaller, more independent, and loosely coupled, requiring only a lightweight API protocol, such as REST, rather than an enterprise service bus or simple object access protocol (SOAP) used in SOA [10].

2.2 Technologies related to Microservices

The success of microservices can be attributed to ten key technologies, some developed prior to the concept and some assisting its mainstream adoption [10]. Two of these technologies are containerization for easier deployment and container orchestration for management automation. Four other technologies are related to the need for distributed communication. Two of these are service discovery and fault-tolerant communication libraries, which provide the tools needed for reliable communication between services. The other two are sidecars that abstract these

communication-related features from developers and service mesh that provides secure communication and monitoring. The other four technologies are monitoring technologies, DevOps automation solutions, chaos engineering tools to test reliability and security, and serverless computing, which allows smaller and more fine-grained services.

Serverless computing is a cloud paradigm where the provider manages infrastructure on-demand and automatically executes code in response to events, such as hypertext transfer protocol (HTTP) requests or database changes.

3 Security Challenges

The security challenges in this paper are divided into seven categories: hardware, virtualization, network, host, service, orchestration, and configurations. This categorization is simply a logical decomposition of microservices into layers. The categories are each an abstraction layer based on a similar idea to the open systems interconnection (OSI) model. The base layer starts with the physical layer consisting of the hardware of the server. The next layer is the virtualization of each microservice; instead of accessing the actual hardware, they each access a virtual environment that simulates the hardware. The third layer is the communication layer which represents the network communication between the microservices. The next layer is the host. This layer consists of the operating system (OS) and the platform this host represents, which could be part of paradigms, such as a cloud computing node or a serverless function. The fifth layer describes the service and the application. The orchestration layer constitutes the orchestration and management of the microservices, such as relaunching new instances of each microservice or the service discovery that provides routing between these dynamically created microservices. The last layer, Configurations, represents the layer with which developers directly interact. Table 1 summarizes security issues and solutions mentioned in this paper.

3.1 Hardware

Sharing the same hardware with other cloud tenants poses a potential security threat. It is possible for adversaries to take advantage of side-channel exploits, such as Meltdown, Spectre [11] and the FLUSH+RELOAD

Category	Threat	Solution
Hardware	Sidechannel exploits, such as Meltdown, Spectre and the FLUSH+RELOAD technique and hardware backdoors [15]	Utilizing self Designed hardware [15]
Virtualization	Container breakouts, poisoned images and hypervisor compromise and shared memory attacks	Intel Software Guard Extensions
Network	Great attack surface, complexity to enforce security policies and Denial of Service	Zero-trust network and Circuit breaker pattern
Host	Proprietary nature of cloud infrastructures and denial-of-wallet	Monitoring of virtual networks using tools such as FlowTap
Service	Race condition, data at rest and other critical web application security threats, such as SQL injection and Cross-Site Scripting (XSS)	Encrypting data at rest and using static and dynamic code analysis
Orchestration	Integrity of service discovery and registering malicious services	Secure implementation of service discovery and registry components
Configurations	Privilege and deployment misconfiguration	Zero-trust network, regular audits, continuous monitoring and automated security checks

Table 1. A table summarizing the security challenges of microservices and their solutions

technique [19], to gain access to the data of other tenants or to send data to their virtual environment. Meltdown and Spectre are security vulnerabilities that exploit the speculative execution of modern central processing units (CPUs). In a simple example, the exploitation of speculative execution results in the CPU loading data from memory before the actual instruction by predicting the instruction and not checking the access privilege. The FLUSH+RELOAD technique exploits the shared data between an attacker and a victim. Some Operation Systems merge pages with the same data in the page tables. With this knowledge, the attacker can flush this page from the cache and check if, after some time, the victim will reload this page into the cache by accessing it. This scenario will result in the attacker inferring the private data of the victim. Compared to Meltdown, Spectre could pose a more significant threat to cloud providers.

While Meltdown enables unauthorized access to privileged memory, Spectre can possibly manipulate a hypervisor into sending data to a virtual machine (VM) [1].

3.2 Virtualization

Virtualization is the technique of creating a virtual version of a resource. By abstracting resources, multiple VMs can run on a single physical system. One popular virtualization approach uses VMs, which are created and managed by a hypervisor that creates a VM on the host machine. Another type of virtualization that has led to the popularity of microservices is containerization. Containerization is an OS level virtualization. Containerization uses the same kernel for different containers and two features of Linux kernel, namespaces and cgroups. With namespaces, system resources are isolated, while cgroups provide limit enforcement for those resources.

Containers have their own potential security issues, such as denial of service (DoS) attacks, container breakouts, poisoned images, and compromised credentials [21]. The attacks on containers can be categorized into direct and indirect. Direct attacks focus on the shared kernel of the containers, with the intention of modifying it. Indirect attacks have similar objectives to direct attacks, but instead, they aim to compromise the code and image repositories.

3.3 Network

The main challenge of the network layer is the greater surface attack area that results from making internal services accessible from the public internet [5]. An additional challenge in securing modern systems is network complexity [5]. As networks become more intricate, it becomes more challenging to enforce security policies, monitor the network, and conduct forensic analysis. These complexities make it difficult to detect and respond to attacks. Another main challenge in the communication of microservices is the trust between services. This challenge can be highlighted in an attack on Netflix, where a subdomain was compromised, allowing an attacker to abuse the cookies of the user to access or modify the data of the user since Netflix allowed the cookies of all users to be accessed from any subdomain [18].

3.4 Host

Depending on the environment in which the microservice is deployed there could be additional security challenges that have to be considered. Some microservices are deployed in a private cloud but currently, the most popular choice is to host microservices in the cloud. Serverless technology can also have some additional implications with respect to security.

Cloud

Cloud computing presents a challenge in securing microservices due to the difficulties in developing, monitoring, debugging and auditing cloud-based applications, and due to the transparency of these services [14]. Cloud provider infrastructures are proprietary and they are not heavily scrutinized for security vulnerabilities. However, security-through-obscurity alone is known in the security community as a dangerous and unreliable approach [13]. Another issue is the privacy of the users which can be broken due to the access of cloud vendors to the data of their users [21].

Serverless

In cloud computing, considering three main factors, security, performance, and cost, is essential [13]. Users might prioritize security, which may contradict the priority of cloud vendors, minimizing cost. This priority can influence the choice of the execution environment. Virtualization technology, such as VMs, may be beneficial for security but that is not the choice the cloud vendors have made. For example, Amazon has created Firecracker, while Google has developed g-Visor as their execution environment for serverless functions. Additionally, it is important to distinguish between cold containers and warm containers. Cold containers completely isolate and offer higher security at the expense of performance. On the other hand, warm containers offer faster execution times but are less secure. Attackers may execute a novel, persistent attack by storing malicious code in the /tmp directory used by warm containers to keep the malicious code through multiple instances of a function. Another important factor is the choice of scheduling algorithms. Scheduling algorithms based on randomization have an advantage over deterministic Scheduling because they offer better protection against co-location exploits. However, it is worth noting that randomized scheduling algorithms have drawbacks with respect to resource utilization. Ultimately, finding the right balance between security, performance, and cost is crucial in cloud computing.

In addition to these concerns, new types of attacks specific to serverless technology need consideration [13]. Billing attacks known as "denial-of-wallet" is one of them. Serverless functions are vulnerable to various injection attacks, not limited to traditional SQL injections, due to multiple exploitable entry points, resulting in new attack types.

Another problem that is faced when using Serverless functions is that they do not possess sufficient information to identify the relationships of other services with the same application the function belongs to, resulting in the possibility of authentication bypass [13].

3.5 Service

One type of attack in this category is referred to as a race condition [13]. This can happen when some parameters, such as IAM roles or a microservice's code are modified while multiple replicas are running. During the transition period, the platform is in an unstable state where different microservice versions process requests. Attackers may take advantage of this opportunity to implement attacks with the aim of accessing data that they should no longer have access to. Another point of vulnerability is the data at rest, the data that is not actively accessed. These data should be considered vulnerable.

3.6 Orchestration

The structure of a microservice network can frequently change because services may be halted, initiated, or relocated, and service discovery plays a crucial role in identifying services through a central point that works similarly to a DNS. However, a risk exists for attacks, such as attackers taking control of the discovery service or adding malicious services to the system and diverting traffic to them [20]. It is essential to safeguard the orchestration platform and its components, but this area has not been thoroughly studied. It is crucial to develop a secure implementation of service discovery and registry components.

3.7 Configurations

Some problems can be caused by human error. Privilege or deployment misconfigurations can cause substantial damages that are sometimes easily avoidable. Microservice architectures have a large number of services; mistakes, such as leaving a database unprotected or giving public access

to a service that is intended to be private and internal are not acceptable. Subjects usually receive more permissions than they actually require. Often, software developers lack the needed information to define fine-grained security controls to minimize privilege. More importantly, no mechanisms exist to set up the minimum permissions needed by services dynamically [13].

4 Security Solutions

This section offers some solutions to some of the challenges mentioned in the previous section.

4.1 Redefining Perimeter Security

Microservices require defense-in-depth [20]. Zero-trust, a concept focusing on identity-based security where all consumers within a network not only have no authority but also have no knowledge of the location of services in the network, is crucial for the safety of microservices. Istio [9], a service mesh, offers security and monitoring features, including mutual transport layer security (mTLS) with Kubernetes [12]. Moreover, A hardened Istio configuration can also provide zero-trust security in microservice architectures. Cilium [3] is also an open-source solution that provides observability and security on top of Istio. Cilium can be integrated with Istio to enhance security and performance. Cilium provides Layer 3 and Layer 4 security policies for traffic between services and provides protection against a compromised sidecar proxy. Hubble [8] is also implemented on top of Cilium and utilizes eBPF to provide deep visibility into the services and their communication. Hubble can provide Layer 7 visibility by extracting traffic without encryption from within the Istio sidecars when integrated with Cilium. An implementation using these tools has been proposed in [4].

4.2 Securing Data at Rest

While defense in depth can help protect against various types of attacks, it is still crucial to ensure that data at rest is properly encrypted [20]. This indicates that data not actively being used or transmitted should be stored in an encrypted format. Encrypting data at rest provides an additional layer of protection, which will also provide security for the privacy

concerns brought forward by the access of cloud vendors to the data of users.

4.3 Better isolation

In [16], the authors discuss the potential of using microservices to develop critical systems that require secure and correct execution within each microservice. This is supported by [6] that secure containers and compiler extensions can help ensure the integrity, confidentiality, and proper functioning of microservices. To implement these secure containers, Docker containers were used with Intel Software Guard Extensions (SGX) enclave, which provides protection against attacks at the operating system, hypervisor, and cloud provider levels.

4.4 Better observability

The authors present an API called FlowTap in [18], designed for cloud environments to provide comprehensive monitoring of virtual networks. FlowTap forms links between microservices and security monitoring systems, enabling the latter to enforce rules on network traffic. This allows cloud service providers to offer security as a service to their clients.

4.5 Diversity through System Heterogeneity

The microservice architecture provides the opportunity that each individual component be developed in any programming language. This leads to a diversity of components within the microservice architecture. Otterstad and Yarygina proposed in [16] that diversifying the microservices can serve as a technique to mitigate low-level exploitations.

4.6 Fail Fast

Reliability is also an important part of security specially when it comes to DoS attacks. To prevent cascading failures, the Circuit breaker pattern is one of the most important patterns that can help [20].

5 Analysis and Discussion

Microservice architecture offers some benefits. One key advantage is isolating vulnerabilities, as microservices are individually deployable compo-

nents that can be secured independently. This compartmentalization minimizes the risk of a security breach affecting the entire system. Additionally, microservices allow for easier and more frequent security patches. However, the distributed nature of microservices can also introduce new security challenges. The increased number of communication points between services can create and increase the attack vectors, making it essential to ensure that all endpoints are secure. Thus, while microservices can enhance security through modularity and flexibility, they require robust security practices to mitigate the risks associated with a distributed system. Securing communications with mutual authentication, encrypting all data at rest and architectural patterns that increase reliability such as circuit breaker are some of the mentioned essential security practices that have been mentioned in this paper and that every microservice user must follow. In addition to these practices, academia and industry should invest in researching modern attack vectors. Secure implementation of service discovery and registry components and open-sourcing the infrastructures and services of cloud providers are two important steps in that direction.

6 Conclusion

This paper surveys the security of microservices, categorizing the associated challenges and presenting solutions and best practices for adopting a microservice architecture. Some of the mentioned challenges are previously known issues amplified in a microservice architecture due to the increased attack surface. However, most of the challenges described in this paper are new vulnerabilities unique to microservices, the cloud and serverless environments. Microservice users and cloud providers can both benefit from the provided solutions. With the increased usage of this paradigm in critical applications, it is essential for researchers to provide solutions and for the industry to implement and utilize them in their applications.

References

[1] Thomas Brewster. Massive intel vulnerabilities just landed – and every pc user on the planet may need to update, Jan 2018.

[2] Gabrial Brown. Service-based architecture for 5g core networks. *Huawei*

White Paper, 1, 2017.

[3] Cilium. [online]. https://github.com/cilium/cilium. Accessed: 2023-04-07.

[4] Catherine de Weever and Marios Andreou. Zero trust network security model in containerized environments. *University of Amsterdam: Amsterdam, The Netherlands*, 2020.

[5] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, pages 195–216, 2017.

[6] Christof Fetzer. Building critical applications using microservices. *IEEE Security & Privacy*, 14(6):86–89, 2016.

[7] Martin Fowler. [online]. https://martinfowler.com/articles/microservices.html. Accessed: 2023-04-07.

[8] Hubble. [online]. https://github.com/cilium/hubble. Accessed: 2023-04-07.

[9] Istio. [online]. https://github.com/istio/istio. Accessed: 2023-04-07.

[10] Pooyan Jamshidi, Claus Pahl, Nabor C Mendonça, James Lewis, and Stefan Tilkov. Microservices: The journey so far and challenges ahead. *IEEE Software*, 35(3):24–35, 2018.

[11] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020.

[12] Kubernetes. [online]. https://github.com/kubernetes/kubernetes. Accessed: 2023-04-07.

[13] Eduard Marin, Diego Perino, and Roberto Di Pietro. Serverless computing: a security perspective. *Journal of Cloud Computing*, 11(1):1–12, 2022.

[14] Nuno Mateus-Coelho, Manuela Cruz-Cunha, and Luis Gonzaga Ferreira. Security in microservices architectures. *Procedia Computer Science*, 181:1225–1236, 2021.

[15] Vasilios Mavroudis, Andrea Cerulli, Petr Svenda, Dan Cvrcek, Dusan Klinec, and George Danezis. A touch of evil: High-assurance cryptographic hardware from untrusted components. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1583–1600, 2017.

[16] Christian Otterstad and Tetiana Yarygina. Low-level exploitation mitigation by diverse microservices. In *Service-Oriented and Cloud Computing: 6th IFIP WG 2.14 European Conference, ESOCC 2017, Oslo, Norway, September 27-29, 2017, Proceedings 6*, pages 49–56. Springer, 2017.

[17] Long Sun, Yan Li, and Raheel Ahmed Memon. An open iot framework based on microservices architecture. *China Communications*, 14(2):154–162, 2017.

[18] Yuqiong Sun, Susanta Nanda, and Trent Jaeger. Security-as-a-service for microservices-based cloud applications. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 50–57. IEEE, 2015.

[19] Yuval Yarom and Katrina Falkner. Flush+ reload: A high resolution, low noise, l3 cache side-channel attack. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 719–732, 2014.

[20] Tetiana Yarygina and Anya Helene Bagge. Overcoming security challenges in microservice architectures. In *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 11–20. IEEE, 2018.

[21] Dongjin Yu, Yike Jin, Yuqun Zhang, and Xi Zheng. A survey on security issues in services communication of microservices-enabled fog applications. *Concurrency and Computation: Practice and Experience*, 31(22):e4436, 2019.

An Overview on Extended Reality for the Internet of Senses

Patrik Mäki
patrik.maki@aalto.fi

Tutor: Nassim Sehad

Abstract

This paper presents an overview of the use of different senses like haptic, scent, and taste in virtual and augmented reality, the main applications for those, and the current state of the related standardization work.

KEYWORDS: Virtual reality, augmented reality, haptic, Internet of Senses, Metaverse

1 Introduction

Real-world experience can be improved by the use of virtual technologies. Virtual reality (VR) creates an artificial environment from the physical one, whereas augmented reality (AR) improves physical reality by adding artificial elements [20]. Audio-visual senses have been classically used in virtual and augmented realities. Traditional use cases for these have been in the industry, military, and entertainment. To improve the experience of these extended virtualities, other senses like touch, temperature, motion, smell, or taste could be used. The added senses enable new opportunities for immersive experiences [5].

Critical challenges in extending virtual reality are the devices that would allow users to receive and transmit feedback on this new information. The

other challenge is the network technologies with low latency so that data can be quickly transmitted, for example, in the case of remote sensing, where one can experience real-world physical objects using senses like touch. Additionally, the synchronization of multiple sensorial experiences needs to be solved [4].

The physical world events must be measured, digitized, and reproduced or rendered to use different senses for virtual reality systems. The traditional audio-visual senses can be digitized using cameras and microphones and reproduced using display and speaker systems. The haptic senses, which refer to thermal-kinetic physical events, can be digitized using thermal and electromechanical sensors and reproduced using electrically controlled mechanical devices [17]. The scent senses measurements, and digitization can be done using electrochemical sensors and reproduced using electronically controlled chemical odor release systems [24]. The taste senses can be measured using a combination of thermal, chemical, and electrical sensors and reproduced with a similar combination of thermal, electric, and chemical release mechanisms [29].

Different combinations of the senses can also be used together. For example, the haptic senses can enhance the audio-visual experience or all together, as in a 1962 Sensorama system that combined 3D movie, physical vibrations, wind, and smell of odors [5].

In order to evaluate the practical possibilities for extended senses use in virtual reality, this paper discusses the technologies and their availability. Furthermore, the protocol and data formats standardization state is also covered, as creating compatible data providers, servers, and clients is vital.

This paper summarizes the current research and the emergence of technologies that will allow the realization of extended virtual reality for selected senses. Section 2 of this paper presents the technologies needed for extended virtual reality. Section 3 presents the emerging standards available for the Internet of Senses. Section 4 describes a few of the most revolutionary applications utilizing the virtual senses. Finally, section 5 summarizes and concludes this paper.

2 Senses and technologies

2.1 Haptic senses

In addition to audio-visual senses, touch sense and movement via a haptic system could be the next most important virtual sense. The haptic system means the use of sensory information from movement and thermal receptors from the skin together with other biological movement receptors from muscles and joints. This allows humans to get feedback from physical stimuli like touch, movement, wind, and pressure [17, 3, 35, 1].

Some forms of haptic feedback systems have been used historically when using remote-operated robotic or mechanical hands to process dangerous chemicals or materials. In order to utilize proper haptic sensory two-way transmission is needed. The user may introduce movement or pressure and should be replied back with a sense of backpressure, friction, movement, temperature, or vibration, as a few examples. All these senses should be measured, transmitted, and reproduced. Perhaps the first typical example used in virtual reality is haptic gloves which allow the user to transmit movement and relay back feedback.

Haptic data can be measured using physical sensors like piezo-electric vibration, 3D motion, temperature, wind, and pressure sensors. Additionally, data can be produced using automatic audio-visual data analysis. For example, low-frequency sound can be used to produce vibration effects or by manually producing the data based on the content [5]. The data can then be reproduced for the user by using a combination of mechanical vibrators, motors, air blowers, and heat cells, which can be either wearable interfaces like gloves, jackets, or even suits, platforms like chairs or floors, or special devices like force-feedback joystick [10].

2.2 Scent

Sense of scent is another interesting but challenging topic to cover in virtual reality. In theory, odor digitalization would require chemical gas chromatography and mass spectrometry analysis so that compositions of odors could be classified to the original chemical compounds [22]. Then a device that could produce the typical combinations of the chemical compounds could be used to produce odors.

In practice, a digital analyzer, i.e., an electronic nose, is used to collect

and analyze sets of different odors. The electronic nose can have multiple sensors capable of detecting different chemicals related to scents. The production of scents is called a digital scent synthesizer. Early prototypes of these were created in the 1960s, and since 1999 there were products available that could produce odors by combing 128 basic odors. In addition, chemical scents and additional stimuli like electrical stimulation with different frequencies have been researched to maximize the effects of smell [24].

There have been a few companies attempting to produce commercial Solutions, but some of the first ones have already closed their businesses. Currently, there are some commercial products to produce a digital smell, like Olorama Technology Inc products that produce about 200 different scents [12]. Another product is available from OVR Technology, which uses eight aromas to produce different kinds of odors that can be added to the VR experience [13].

However, there are challenges in using artificial scents; the scent is a very subjective sense, and different people have a different level of tolerance of scents [26]. Also, some people may be sensitive to certain odors, and the use of these odors could become a health issue due to allergies.

2.3 Taste

The taste is also a possible but technically challenging feature to be added to the Internet of Senses. The human taste sense is based on a combination of tongue and nose scent receptors. The taste receptors can detect five main types of tastes: sweetness, sourness, saltiness, bitterness, and savoriness [6]. Additionally, the food odor is part of the taste sensation. Also, the human tongue can detect the surface and temperature of the food. For a complete artificially produced taste experience, all of these factors should be taken into account.

Like in scents, a digital analyzer, i.e., an electronic tongue and synthesizer or virtual taste, is needed to measure and produce the components of taste. One research prototype of a taste synthesizer has been developed at Meiji University [19], which produces different tastes using five different gels and can be controlled electrically. Another research prototype was developed at Zhejiang University called E-Taste, which uses a combination of electrical and thermal simulation to trigger different basic tastes [29].

Commercial electronic tongue products are available, and they are used as analytical instruments in the agricultural and pharmaceutical indus-

try like Astree Electronic Tongue from SA Alpha MOS France [28]. However, commercial products for virtual taste are still in the early stages.

3 Applications

3.1 Industrial

In industry, the already existing remote mechanical movement applications can be extended so that experts don't need to necessarily move for examing mechanics but can use remotely operated mechanical actuators with virtual gloves.

3.2 Medical

In medical applications, there are already remote surgery applications available used in training, simulation, and operational cases. These use a combination of visual and haptic sensors to allow surgeons to remotely operate robot hands or other medical instruments when performing surgical operations or other great accuracy-required operations. This currently, however, requires that both doctor and patient must go to the specialist hospital which has the equipment for this kind of operation. In the future, perhaps we could see possibilities of wearing special haptic clothes, which would allow doctors to examine patients remotely.

Safety and usability are very important for medical solutions, and in order to measure those in virtual reality-based systems, a set of evaluation metrics needs to be defined. These can be categorized into "haptic feedback fidelity, stability, real-time performance, and user-friendliness" [18]. In the case of technical performance metrics, these relate to latency, data update frequency, and error to check the accuracy of medical operations like the drilling of a tooth.

3.3 Entertainment

Entertainment applications have been one of the first use cases for realizing the additional senses. There have been attempts to produce an enhanced movie experience with a 4D movie, where vibrations and movement have been added to the viewer's seats. Also, there was an attempt to create a personal viewer experience with movement, airflow, and selected

odors in 1962 with Sensorama products. This project was not a success as it was a bulky large personal device, the movies lasted only 10 minutes, and it did not get funding. Lately, the improvements and cost reduction of haptic devices have allowed the use of remote sensing to be applied to individual users [5].

3.4 Remote presence

Perhaps the largest scoped application for extended virtual reality is a remote presence experience [14]. This allows a multitude of professional and free time use cases like traveling to hazardous areas with the possibility of controlling the feedback on remote sensing as well as simulation of different locations or experiences. Another practical use case is to use this to improve training and education to experience disabilities or the impact of different ages on senses.

3.5 Metaverse

A concept called Metaverse utilizes virtual and augmented reality and enables multisensory experience [21]. The Metaverse combines Internet and Social connections and improves the experience with the use of Virtual and Augmented reality technologies. The Metaverse has been advertised by Facebook's creator Meta Inc.

Metaverse attempts to create a new way of experiencing the Internet. One use case is to improve traditional online 2D learning experiences that suffer from a lack of real human interaction. Metaverse would provide virtual human interaction that rivals real human interaction. The Metaverse could be implemented as a combination of massively multiplayer online games (MMO) and real-world AR. There are currently several challenges to adopting the Metaverse, like the high cost of user equipment which may be reduced in the future, physical risks due to focused attention to the virtual world leading to accidents, information overload causing psychological challenges, and moral, privacy, and security risks. There are also health concerns with 3D and VR effects causing motion sickness symptoms, head and neck pain due to the heavy weight of VR headsets, and the risk of social isolation.

4 Standards

In order to create solutions with different products, the interoperability between servers, clients, and devices should be standardized for formats and protocols. The requirements for standards of VR and AR could be considered being: low latency, high throughput, security, and configurable reliability. Interoperability could also be considered a requirement in the future, but so far, it seems to be so that solutions are developed by a single company or organization, and interoperability has not been focused on.

To understand the current state of standardization, a literature study can be done on several standardization bodies like ISO, IEEE, IETF, ITU, and W3C organizations:

- ISO/IEC group has specified MPEG-V Media context and control architecture standard as ISO/EIC 23005. This standard provides architecture and specifications for representing the data for interoperability between virtual and real worlds using XML. [7].

- IEEE has a working group for Virtual and Augmented Reality, but the standards are still in the early phase [25, 11]. IEEE has also published a group of standards for interfacing the cyber and physical works in IEEE 2888, which define a set of sensor and actuator interfaces [36].

- ITU group has also published recommendations on AR, VR, and XR related to Quality of experience in P.1320 [32], G.1036 [31] and Testing in Q.4066 [30] documents.

- IETF has a few drafts on AR and VR transport and use case topics [16, 9]. However, there are applicable standards for using AR and VR data transfers like RTP and RTPS.

- W3C has incorporated work on VRTP and VRML from the end of the 1990s, but those have not had much attention lately [34]. Currently, there is a working group on immerse web that covers AR and VR topics [33].

- NIST has formed an Extended Reality Community of Interest to improve the collaboration of researchers in different NIST laboratories

[23].

- Khronos group (the group behind OpenGL) has specified OpenXR developer API (not a protocol) for AR and VR systems [8].

4.1 Data transfer

Audiovisual data transfers are typically done using some form of stream encoders like MPEG-4/H.264 or MPEG-H/H.265 and low latency transfer protocols like RTP/RTSP or WebRTC family over UDP [2]. The information for other senses can be compacted with smaller structural data that could be presented in JSON or XML or binary variants of those. The information can then be transferred using some form of message protocol like MQTT, which is also used in IoT applications [27] or included in the video stream as additional metadata like is possible with MPEG-V [15] using XML encoded data.

The MPEG-V with presentation timestamps could be used to solve the problem of synchronizing multiple sensorial experiences, at least when personal devices are used [4].

Example of the MPEG-V XML data looks like [5]:

```
<sedl:SEM>
    <sed:Effect xsi="sev:RigidBodyMotionType" si:pts="1593000">
        <sev:MoveToward distance="200" acceleration="30"/>
    </sed:Effect>
    <sedl:GroupOfEffects si:pts="1647000">
    <sedl:Effect xsi:type="sev:VibrationType"
                  intensity-range="0 100" intensity-value="10"/>
    <sedl:Effect xsi:type="sev:WindType"
                  intensity-range="0-100" intensity-value="5"/>
    </sedl:GroupOfEffects>
</sedl:SEM>
```

Listing 1. An example of MPEG-V XML event

The example above describes timestamped (pts) effects which simulate movement by triggering body motion, vibration, and wind. This kind of event could be used together with a movie if a suitable VR/AR headset or chair is used, which can produce these effects.

5 Conclusion

This paper covered a summary of the current research and the technologies useful for extending virtual reality from traditional audio-visual experience with additional senses like haptic, scent, and taste. Also, the standardization efforts for using these additional senses in virtual reality were covered. The additional senses most realized are the haptic senses. The taste senses still lack commercially available products. Also, both scent and taste are individual and subjective experiences, so it is challenging to normalize these to match personal preferences.

There are several use cases for extending virtual reality with additional senses, including technology, the medical industry, and entertainment. The haptic senses are in use in several applications to provide the capability of remote sensing. In the case of the medical industry, there are also critical metrics that have to be measured and validated to ensure safety. However, the limited progress in standardization, cost of products, and subjective attitude have been limiting the use of the additional senses beyond the haptic senses.

Based on the literature review, there have been several standardization attempts to describe the virtual senses, which some have already been obsoleted, like W3C VRML. However, the most prominent standard for describing the presentation of the additional senses is MPEG-V. The MPEG-V allows additional metadata to be included in XML format that can be embedded into the audio-visual data stream.

In summary, the early stages of research, cost of devices, lack of realized standards, and user acceptance are still preventing the wider use of new senses like scent and taste for virtual reality.

References

[1] M. Azmandian, M. Hancock, H. Benko, E. Ofek, and A. Wilson. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 chi conference on human factors in computing systems*, pages 1968–1979, 2016.

[2] N. Blum, S. Lachapelle, and H. Alvestrand. Webrtc: Real-time communication for the open web platform. *Commun. ACM*, 64(8):50–54, jul 2021.

[3] S. Brewster and R. Murray-Smith. *Haptic human-computer interaction: First international workshop, glasgow, uk, august 31-september 1, 2000, proceedings*, volume 2058. Springer, 2003.

[4] A. Covaci, L. Zou, I. Tal, G. Muntean, and G. Ghinea. Is multimedia multisensorial? - a review of mulsemedia systems. *ACM Comput. Surv.*, 51(5), sep 2018.

[5] F. Danieau, A. Lecuyer, P. Guillotel, J. Fleureau, N. Mollet, and M. Christie. Enhancing audiovisual experience with haptic feedback: A survey on hav. *IEEE Transactions on Haptics*, 6(2):193–205, 2013.

[6] Institute for Quality and Germany Efficiency in Health Care, Cologne, 2016.

[7] International Organization for Standardization. ISO/IEC 23005 media context and control. Standard, 2020.

[8] Khronos Group. Openxr. https://www.khronos.org/api/index_2017/openxr.

[9] L. Han and K. Smith. Problem statement: Transport support for augmented and virtual reality applications. Internet-Draft draft-han-iccrg-arvr-transport-problem-01, IETF Secretariat, March 2017.

[10] Y. Huang, K. Yao, J. Li, D. Li, H. Jia, Y. Liu, C. Yiu, W. Park, and X. Yu. Recent advances in multi-mode haptic feedback technologies towards wearable interfaces. *Materials Today Physics*, 22:100602, 2022.

[11] IEEE. IEEE 2048 VR/AR working group.

[12] Olorama Inc. Olorama web page. https://www.olorama.com/.

[13] OVR Technology Inc. Ovr technology web page. https://ovrtechnology.com/.

[14] K. Kilteni, R. Groten, and M. Slater. The sense of embodiment in virtual reality. *Presence: Teleoperators and Virtual Environments*, 21(4):373–387, 2012.

[15] J. Kim, Y. Kim, and J. Ryu. MPEG-V standardization for haptically interacting with virtual worlds. In *2013 World Haptics Conference*, pages 55–60, 2013.

[16] R. Krishna and A. Rahman. Media operations use case for an extended reality application on edge computing infrastructure. Internet-Draft draft-ietf-mops-ar-use-case-09, IETF Secretariat, November 2022.

[17] S Lederman and R. Klatzky. Haptic perception: A tutorial. *Attention, perception, and psychophysics*, 71:1439–59, 10 2009.

[18] A. Lungu, W. Swinkels, L. Claesen, P. Tu, J. Egger, and X. Chen. A review on the applications of virtual reality, augmented reality and mixed reality in surgical simulation: an extension to different kinds of surgery. *Expert Review of Medical Devices*, 18(1):47–62, 2021. PMID: 33283563.

[19] H. Miyashita. Norimaki synthesizer: Taste display using ion electrophoresis in five gels. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, page 1–6, New York, NY, USA, 2020. Association for Computing Machinery.

[20] L. Muñoz-Saavedra, L. Miró-Amarante, and M. Domínguez-Morales. Augmented and virtual reality evolution and future tendency. *Applied Sciences*, 10(1), 2020.

[21] S. Mystakidis. Metaverse. *Encyclopedia*, 2(1):486–497, 2022.

[22] T. Nakamoto, M. Ohno, and Y. Nihei. Odor approximation using mass spectrometry. *IEEE Sensors Journal*, 12(11):3225–3231, 2012.

[23] National Institude of Standards and Technology. Extended reality at nist. https://www.nist.gov/information-technology/extended-reality.

[24] D. Panagiotakopoulos, G. Marentakis, R. Metzitakos, I. Deliyannis, and D. Dedes. Digital scent technology: Toward the internet of senses and the metaverse. *IT Professional*, 24(3):52–59, 2022.

[25] C. Perey. Open and interoperable augmented reality and the ieee [standards]. *IEEE Consumer Electronics Magazine*, 4(4):133–135, 2015.

[26] A. Sabiniewicz, E. Schaefer, C. Guducu, C. Manesse, M. Bensafi, N. Krasteva, G. Nelles, and T. Hummel. Smells influence perceived pleasantness but not memorization of a visual virtual environment. *i-Perception*, 12(2), 2021.

[27] L. Suzuki, K. Brown, S. Pipes, and J. Ibbotson. Smart building management through augmented reality. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 105–110, 2014.

[28] Y. Tahara and K. Toko. Electronic tongues–a review. *IEEE Sensors Journal*, 13(8):3001–3011, 2013.

[29] A. Ullah, Y. Liu, Y. Wang, H. Gao, H. Wang, J. Zhang, and G. Li. E-taste: Taste sensations and flavors based on tongue's electrical and thermal stimulation. *Sensors*, 22(13), 2022.

[30] International Telecommunication Union. ITU-T Q.4066 testing procedures of augmented reality applications. Recommendation, 2020.

[31] International Telecommunication Union. ITU-T G.1036 quality of experience influencing factors for augmented reality services. Recommendation, 2022.

[32] International Telecommunication Union. ITU-T P.1320 quality of experience assessment of extended reality meetings. Recommendation, 2022.

[33] W3C. Immerse web working group. https://www.w3.org/immersive-web/.

[34] W3C. Vrml virtual reality modeling language. https://www.w3.org/MarkUp/VRML/.

[35] C. Wee, K. Yap, and W. Lim. Haptic interfaces for virtual reality: Challenges and research directions. *IEEE Access*, 9:112145–112162, 2021.

[36] K. Yoon, S. Kim, S. Jeong, and J. Choi. Interfacing cyber and physical worlds: Introduction to ieee 2888 standards. In *2021 IEEE International Conference on Intelligent Reality (ICIR)*, pages 49–50, 2021.

Profiling stencil computations for GPUs using Astaroth library

Pawel Strozanski
pawel.strozanski@aalto.fi

Tutor: Maarit Korpi-Lagg

Abstract

Computational science is a field more and more reliant on GPUs, primarily due to physical limitations in the development of general-purpose CPUs. Even though GPU programming comes with its own set of challenges, it has brought significant performance improvements, with NVIDIA hardware and software stacks dominating the landscape of high performance computing. However, that dominance may not last long, as AMD has entered the market with their set of data center oriented GPUs, which have been warmly welcomed by supercomputers all around the world.

In this work, we compare the performance of NVIDIA and AMD GPUs in a particular application – stencil computations. We outline the background behind the CPU vs GPU matter in scientific computing, as well as explain the idea of stencil computations and the underlying Astaroth library. Then, we describe the methodology of the experiment. Finally, we provide and comment on the experiment results.

KEYWORDS: stencil computations, Astaroth library, GPU performance profiling, accelerator platforms

1 Introduction

In the past, the world of scientific computing has been primarily revolving around single-core, general-purpose CPUs. At the time, it was believed that processor speeds would keep doubling every 18 months, providing consistent improvements in high-performance computing (HPC) [1]. This stopped holding true as early as 2004, when CPU manufacturers encountered a "power wall" – the amount of power needed to be dissipated prevented the clock speeds from rising indefinitely [2, 3]. The universally accepted solution to this problem was acknowledging the clock speed limit and instead shifting towards a parallel, multi-core paradigm [4].

Because scientific computing processes, especially physics simulations, are able to take advantage of the parallelism offered by multi-core processors, a simple truth arises: the more cores, the better. State-of-the-art CPUs can have up to 96 physical cores [5], providing a significant edge in parallel operations. However, the quest for improving performance is an unending one, and scientists have been looking outside the realm of general-purpose processors, focusing on graphics processing units. Modern GPUs can have upwards of 15000 cores capable of executing arithmetic instructions in parallel, memory throughputs vastly superior to these of CPUs [6], and be noticeably more cost- and space-efficient than equivalent general-purpose processor clusters [7].

Importantly, modern GPUs can be programmed using frameworks such as CUDA Toolkit (for NVIDIA GPUs) or ROCm (for AMD GPUs) to run the scientific computations on these highly parallel processors [8]. For the most part, NVIDIA's hardware and software stack has been the primary choice for HPC due to its maturity and lack of similar well-developed alternatives [9]. However, as AMD accelerators have recently started to emerge into the market, their ecosystem could become a promising contestant for the long-standing NVIDIA dominance [9]. In fact, at the time of writing, the first and third supercomputers (by performance, according to the TOP500 list) are powered by AMD hardware [10]. Hence, it is desirable to conduct studies on how the ROCm framework compares to CUDA in terms of performance, efficiency and ease-of-use.

In this paper, we compare the performance of both NVIDIA and AMD accelerators in a scientific computing usage. Our objective is to verify whether the hardware from both companies is on comparable performance levels, and if not, how large is the gap between them.

This paper is structured as follows: in section 2, we introduce the necessary background on scientific computing. Section 3 outlines the methodology and expectations of the experiment. In section 4, we present, interpret, and discuss the results. Finally, in section 5, we provide concluding remarks.

2 Stencil computation on a GPU

In the past, GPUs were strictly designed as expansion cards for performing expensive graphics-related operations. With time, however, they became hugely powerful devices with thousands of parallel cores and gigabytes of high-speed memory. Software frameworks (e.g. CUDA Toolkit) have been developed to enable the possibility of running arbitrary computations on these coprocessors, and the scientific community has been quick to take advantage of the new possibilities.

The high throughput provided by GPUs is utilised not only in graphics-related fields, but also in common scientific applications, such as machine learning, linear algebra or differential equation solving [11]. This, in turn, accelerates various physics simulations, which are the foundations of modern computational sciences, including molecular and fluid dynamics, bioinformatics, and magnetohydrodynamics [11, 12, 13]. The possibilities are endless; in this paper, however, we will focus on a specific application – stencil computations.

A *stencil* is a particular type of recurring operation which accepts a multidimensional array and updates each element by performing some computation, taking variable amounts of neighbouring elements elements into account [14]. Stencil functions are widely used in physical simulations [13], and thanks to their structure stencils efficiently lend themselves to parallelisation on multi-core platforms. Here GPUs take the podium – thousands of cores and memory throughput in the range of hundreds of gigabytes per second make them ideal candidates for accelerating stencils [14].

That is not to say, however, that GPU programming is without challenges.

2.1 Domain-specific languages

Up until this point we have essentially presented GPU accelerators as highly parallel processors, however the reality is that achieving significant performance improvements over CPUs is a difficult process which requires expertise in several fields.

First, writing code for execution on a GPU necessitates the usage of platform-specific language extensions (CUDA for NVIDIA, HIP for AMD). Already at this point, the entry barrier is relatively high as these extensions are designed to work with low-level programming languages such as C or C++. Knowledge of these languages is therefore required.

Second, not all GPUs are made equal. Different models have varying parameters (e.g. thread count per block or preferred memory access patterns) and writing efficient, optimised code requires detailed knowledge about the target hardware [13]. This also creates difficulties when the code needs to be ported onto a different GPU – very often a reimplementation of the program is unavoidable in order to preserve high performance on new hardware [15, 16].

While neither of these problems in isolation is an insurmountable one, it is quite unreasonable to expect scientific researchers to not only design the mathematical foundations of their simulations, but also familiarise themselves with low-level programming and intricate details of the underlying hardware. These hurdles can be alleviated by using a domain-specific language (DSL) instead of GPU manufacturer's native language (e.g. CUDA). By focusing on a particular subset of GPU features and function types (e.g. stencils), a DSL enables writing simpler, more expressive code while avoiding hardware-specific optimisation quirks and still preserving most of the native-level performance [13, 17].

This paper focuses on one particular software library called Astaroth [13], which focuses on stencil computations and features a high-level DSL designed for writing stencil functions, as well as a compiler for the DSL that delivers "near handtuned performance" [18]. While the initial iterations of Astaroth targeted exclusively CUDA (that is, NVIDIA hardware), recently focus has been shifted to also include support for HIP (AMD hardware). This, combined with the fact that more and more supercomputers are featuring AMD GPUs, warrants conducting practical experiments to answer the question: right now, are AMD GPUs able to compare with NVIDIA's hardware in stencil computation?

3 The experiment

The experiment will be conducted using Astaroth. With the library comes a set of benchmarking tools, one of which (`benchmark-device`) is ideal for testing single-GPU performance in stencil computation. We will run the benchmark on the Triton computing cluster, which features several models of modern NVIDIA GPUs as well as an AMD Instinct MI100 [19], and compare the run times between different hardware.

The three GPUs selected for testing are NVIDIA Volta V100 (PCIe, 32 GB), NVIDIA Ampere A100 (SXM, 80 GB) and AMD Instinct MI100 (PCIe, 32 GB). At the time of writing, these models are the most modern offerings of Aalto Triton, making them reasonable candidates for comparison.

In the table below, we present the GPU specifications according to their corresponding manufacturer's datasheets [20, 21, 22]. It is, however, important to note that the claimed performance figures have been achieved in synthetic benchmarks (based on tasks related to machine learning) and thus may not represent the results obtained in Astaroth, which is largely a different type of algorithm.

GPU model	Cores	FP64 performance	FP32 performance
NVIDIA V100	5120	7 TFLOPS	14 TFLOPS
NVIDIA A100	7936	9.7 TFLOPS	19.5 TFLOPS
AMD MI100	7680	11.5 TFLOPS	23.1 TFLOPS

3.1 Methodology

When compiling Astaroth, there are certain parameters that can be fine-tuned in order to attempt to achieve the best performance on a particular piece of hardware. Normally one would not adjust these parameters manually, but as we are trying to compare two very different GPU architectures, the optimal values are likely to differ between each architecture. The two parameters in question are:

- Maximum threads per block[1]. By default this value is set to 0, which leaves certain parts of GPU code optimization to the compiler. We will test the default as well as override it with custom values between 64 and 1536, in increments of 32 threads.

[1] This corresponds to Astaroth's `MAX_THREADS_PER_BLOCK` compile-time macro.

- Caching strategy[2]. Astaroth allows us to choose either implicit (the default) or explicit caching. We will test both.

Since adjusting the aforementioned parameters requires recompiling Astaroth, we have prepared scripts to automate the testing process[3]. The repository also contains all intermediate and post-processing results.

Finally, we should also mention that the `benchmark-device` tool accepts the input mesh dimensions as command-line arguments. For this experiment we have rather arbitrarily chosen a mesh size of 192 in all dimensions – a value large enough to stress the GPUs, but not too large in order for the tests to complete in a reasonable amount of time.

4 Results

The results of the experiment are presented in the figures below. For each device, the best result has been recorded into the table along with the corresponding value for the maximum threads per block.

GPU model	Best run; implicit caching	Best run; explicit caching
NVIDIA V100	7.3 ms (TPB = 0)	33.2 ms (TPB = 768)
NVIDIA A100	3.3 ms (TPB = 0)	17.6 ms (TPB = 928)
AMD MI100	10.4 ms (TPB = 512)	111.5 ms (TPB = 1440)

Looking at the table, two observations become obvious. First, the explicit caching variant performed significantly worse than implicit caching. As the graphs below demonstrate, for some values of maximum threads per block explicit caching was in fact faster, however it does not negate the fact that the best results were always achieved with implicit caching.

Second, and more importantly, AMD MI100 did not perform as well as NVIDIA devices in this test, despire theoretically being the fastest out of the three tested GPUs. In fact, with the default TPB = 0 the AMD device achieved even worse result of 20.5 ms; manual tuning of the parameter halved that time, but it still did not come close to the competitors' results.

For completeness, we also present charts showing the performance of each device on all intermediate values of maximum threads per block.

[2]This corresponds to Astaroth's `IMPLEMENTATION` compile-time macro.
[3]`https://github.com/Taikelenn/astaroth-testing`

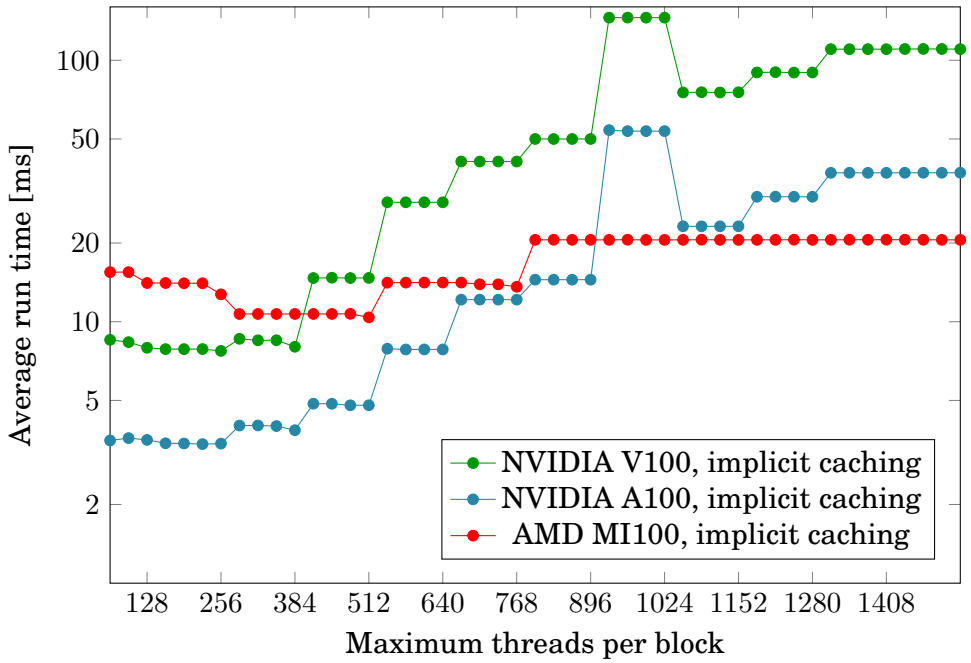


Figure 1. Average run time for various values of maximum threads per block. Caching strategy is set to implicit caching. Lower values are better.

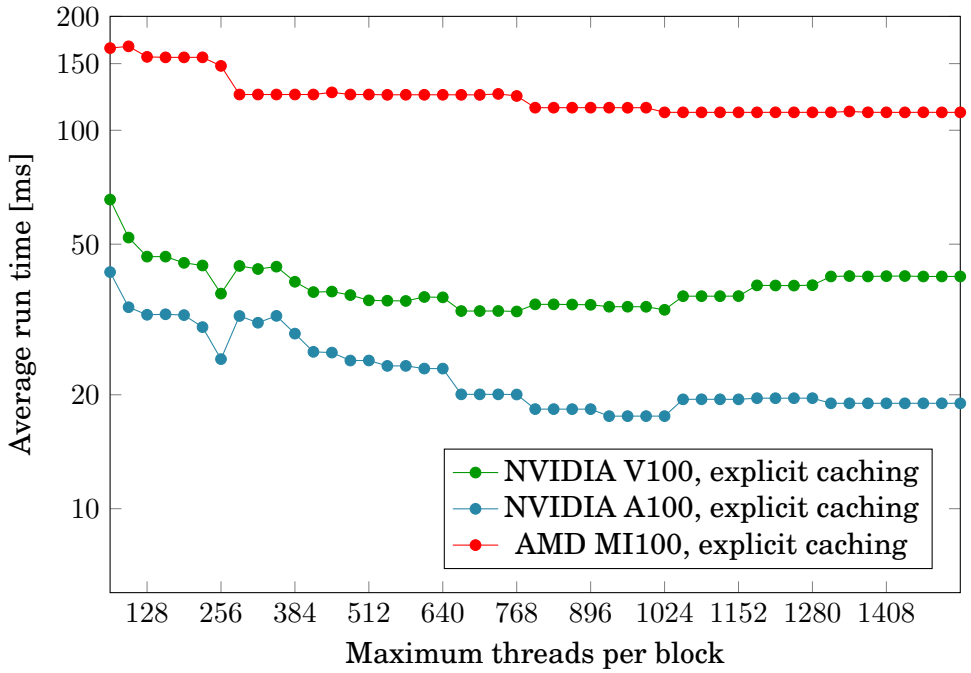


Figure 2. Average run time for various values of maximum threads per block. Caching strategy is set to explicit caching. Lower values are better.

5 Conclusion

This experiment has shown that while AMD GPUs certainly can be used with Astaroth, their performance is less than ideal. The closest NVIDIA competitor to AMD MI100 is the Ampere A100, which in this particular

setting has achieved three times faster run times.

It is imperative to remind, however, that this experiment does not demonstrate overall superiority of NVIDIA GPUs over AMD. Such a conclusion could only be reached by comparing algorithms implemented natively on both platforms. Astaroth was initially designed for NVIDIA devices, and as of today it generates CUDA code which is only then converted to HIP – a bias towards NVIDIA hardware should not be surprising.

Astaroth is currently seeing active development in an attempt to improve compatibility and performance on AMD hardware. Therefore, even if NVIDIA is currently the go-to platform for stencil computations with Astaroth, we should hope for AMD GPUs to become more and more viable candidates in the near future.

References

[1] H. Sowizral, "Scene graphs in the new millennium," *IEEE Computer Graphics and Applications*, vol. 20, no. 1, pp. 56–57, 2000.

[2] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "A view of the parallel computing landscape," *Commun. ACM*, vol. 52, p. 5667, Oct 2009.

[3] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Power limitations and dark silicon challenge the future of multicore," *ACM Trans. Comput. Syst.*, vol. 30, Aug 2012.

[4] P. F. Gorder, "Multicore processors for science and engineering," *Computing in Science and Engineering*, vol. 9, no. 2, pp. 3–7, 2007.

[5] Advanced Micro Devices, Inc. `https://www.amd.com/en/products/cpu/amd-epyc-9654`, 2022. [Online; accessed 2023-01-30].

[6] NVIDIA Corporation. `https://resources.nvidia.com/en-us-tensor-core`, 2022. [Online; accessed 2023-01-30].

[7] V. V. Kindratenko, J. J. Enos, G. Shi, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips, and W.-m. Hwu, "GPU clusters for high-performance computing," in *2009 IEEE International Conference on Cluster Computing and Workshops*, pp. 1–8, 2009.

[8] M. Papadrakakis, G. Stavroulakis, and A. Karatarakis, "A new era in scientific computing: Domain decomposition methods in hybrid CPU-GPU architectures," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 13, pp. 1490–1508, 2011.

[9] K. Shafie Khorassani, J. Hashmi, C.-H. Chu, C.-C. Chen, H. Subramoni, and D. K. Panda, "Designing a ROCm-aware MPI library for AMD GPUs: Early experiences," in *High Performance Computing*, (Cham), pp. 118–136, Springer International Publishing, 2021.

[10] TOP500 Supercomputer List. https://www.top500.org/lists/top500/2022/11/, 2022. [Online; accessed 2023-02-01].

[11] G. Pratx and L. Xing, "GPU computing in medical physics: A review," *Medical Physics*, vol. 38, no. 5, pp. 2685–2697, 2011.

[12] M. C. Schatz, C. Trapnell, A. L. Delcher, and A. Varshney, "High-throughput sequence alignment using graphics processing units," *BMC Bioinformatics*, vol. 8, p. 474, Dec 2007.

[13] J. Pekkilä, "Astaroth: A Library for Stencil Computations on Graphics Processing Units," Master's thesis, Aalto University. School of Science, 2019.

[14] P. S. Rawat, M. Vaidya, A. Sukumaran-Rajam, A. Rountev, L.-N. Pouchet, and P. Sadayappan, "On optimizing complex stencils on GPUs," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 641–652, 2019.

[15] M. Ravishankar, J. Holewinski, and V. Grover, "Forma: A DSL for image processing applications to target GPUs and multi-core CPUs," in *Proceedings of the 8th Workshop on General Purpose Processing Using GPUs*, GPGPU-8, (New York, NY, USA), p. 109120, Association for Computing Machinery, 2015.

[16] N. Zhang, M. Driscoll, C. Markley, S. Williams, P. Basu, and A. Fox, "Snowflake: A lightweight portable stencil DSL," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 795–804, 2017.

[17] B. Hagedorn, L. Stoltzfus, M. Steuwer, S. Gorlatch, and C. Dubach, "High performance stencil code generation with Lift," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, CGO 2018, (New York, NY, USA), p. 100112, Association for Computing Machinery, 2018.

[18] Astaroth - A Scalable Multi-GPU Library for Stencil Computations. https://bitbucket.org/jpekkila/astaroth/src/master/, 2023. [Online; accessed 2023-03-04].

[19] Triton - Cluster technical overview. https://scicomp.aalto.fi/triton/overview/, 2022. [Online; accessed 2023-03-05].

[20] NVIDIA Corporation. https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf, 2020. [Online; accessed 2023-04-07].

[21] NVIDIA Corporation. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf, 2022. [Online; accessed 2023-04-07].

[22] Advanced Micro Devices, Inc. https://www.amd.com/en/products/server-accelerators/instinct-mi100, 2020. [Online; accessed 2023-04-07].

Review of recent advances of leveraging symbolic execution for fuzzing

Philipp Giersfeld
philipp.giersfeld@aalto.fi

Tutor: Lachlan Gunn

Abstract

Fuzzing has become the de-facto standard for vulnerability discovery. While combining a fuzzer with concolic execution has been an interesting topic of research, only recently have they started competing for the highest scores on popular benchmarks. In this paper, we provide a comprehensive review of recent advances in utilizing program analysis for automated testing of software applications. Specifically, the focus is on symbolic execution techniques for hybrid fuzzing, and the paper compares approaches for tracing, Intermediate Representation (IR), constraint collection, and constraint solving. The research highlights the key differences between different approaches. Additionally, the paper suggests possible studies to further explore the advantages and disadvantages of these techniques, including more detailed evaluations of their effectiveness and applicability in real-world testing scenarios.

KEYWORDS: Security, Symbolic Execution, Concolic Execution, Fuzzing, Vulnerability Discovery

1 Introduction

With the digitalization of critical applications in recent years, their safety consequently depends on the security of the underlying software, which has become an attractive target for adversaries. Despite significant advances in platform security, which only restrict the impact of a bug once the code is already deployed, vulnerabilities have continued to occur. Memory vulnerabilities have been the continuous culprit for computer security, constituting 70% of all CVEs [16]. Consequently, to minimize the risk of exploitation, there has been a growing demand for effective methods of identifying implementation errors, as indicated by Darpa's Cyber Grand Challenge (CGC) in 2016. The results of which and Google's *OSS-Fuzz* projects demonstrate the effectiveness of automatic analysis tools, e.g., *OSS-Fuzz* was able to find 8800 vulnerabilities in 850 projects since its introduction in 2016 [5].

These automatic analysis tools can be classified into static and dynamic analysis methods. In the former category, symbolic execution is commonly leveraged for whitebox fuzzing to explore the programs [11]. Symbolic execution formalizes code paths as first-order logic equations, which can then be evaluated by powerful Satisfiability modulo theories (SMT) solvers to find inputs triggering the execution of said code path. This approach excels at finding inputs for simple fine-grained conditions, such as magic bytes; however, they suffer from performance limitations, e.g., due to path explosion or memory consumption. On the contrary, dynamic analysis in the form of greybox/blackbox fuzzers overcomes these limitations as fuzzers can be executed natively by applying an evolutionary approach, mutating its input to explore increasingly more code. However, fuzzing is incomplete, it cannot provide the same guarantees as whitebox fuzzing and frequently get stuck due to complex checks, such as the aforementioned ones.

Hence, recent research proposes methods to combine the strengths of both approaches. However, these methods generally suffer a throughput penalty with the benefit of finding higher-quality inputs. Especially recent works successfully focused on optimizing the symbolic execution-bounded analysis to improve fuzzing performance.

In order to compare recent advances in utilizing program analysis for fuzzing, this paper reviews the main contributions of a selection of approaches and compares their approaches for IR, constraint collection, and

constraint solving. Section 2 introduces fuzzing, symbolic execution, and concolic execution. Section 3 defines a comparison space, followed by selecting the approaches for the analysis and comparing them. In Section 4 we propose future research questions and in Section 5 we conclude our work.

2 Software analysis

2.1 Fuzzing

Fuzzing, first coined as a term by Miller et al. in 1990 [15], is the method of executing a program with random input and observing whether it crashes. While initial fuzzers were relatively simple, modern state-of-the-art fuzzers, such as *AFL++* or *Nyx*, employ complex strategies in order to test more code [9] [21]. For example, while *Nyx* improved the current state-of-the-art by focusing on the execution throughput and enabling fuzzing of complex targets, such as hypervisors, *AFL++* consolidates novel approaches in a customizable manner to quickly implement novel/custom approaches, such as different sanitizers, to catch bugs that may not crash the application, mutation strategies to increase the likelihood of exploring different code regions, or coverage metrics to guide the fuzzer towards using interesting inputs. These fuzzers are very powerful at quickly exploring code regions that are conditionally reachable by simple conditions, such as $x > 0$, as they mainly apply random mutations to existing test cases and execute those natively. However, due to their probabilistic nature, this also limits their explorative abilities as they struggle to satisfy specific conditions (such as $x == 0xdeadbeef$). These conditions can be solved by applying a concrete solving technique, such as symbolic execution.

2.2 Symbolic Execution

Symbolic execution explores a program by assigning symbolic variables to input data and executing every possible execution path, essentially forking at every branch, collecting branch conditions as constraints, and finally, stating the code path as a first-order equation [12]. Then, powerful SMT solvers, such as *Z3*, can be used to evaluate a code path and aim to generate concrete inputs to trigger all execution paths [8].

Consider the example program from Listing 1. Here, the input given

```c
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char** argv) {
    if (argc<2)
        return -1;
    int a = strtol(argv[1], NULL, 0);
    int b = a ^ 0xbadc0de;
    if (b == 0xdeadbeef)
        abort();
}
```

Listing 1. Example Program under Analysis

via *STDIN* can be marked as a symbolic input λ. After executing line 8, the value of b is $\lambda \oplus$ `0xbadc0de`. The symbolic execution forks on line 9. Now, there are two path constraints, $\lambda \oplus$ `0xbadc0de` $=$ `0xdeadbeef` for the erroneous path resulting in *abort()* on line 10 and the other path $\lambda \oplus$ `0xbadc0de` $\neq$ `0xdeadbeef`. Continuing, a SMT solver can be consulted to find solutions for both constraints, such as `0xd5007e31` and `1`.

This approach can provide a certain guarantee about the absence of bugs, although simplifications need to be made such that this approach becomes practically feasible. Moreover, this approach suffers from significant scalability issues [17], such as path explosion. Path explosion occurs since the number of paths doubles at every branch, hence, growing exponentially. While existing methods attempt to minimize path explosions [2], ultimately, these strategies only delay it and are not able to completely mitigate it [24].

2.3 Concolic Execution

Concolic execution reduces the overhead of symbolic execution by augmenting symbolic execution with concrete values; hence the name concolic (**conc**rete + symb**olic**) [10]. This approach executes a program concretely while simultaneously collecting symbolic constraints. Thus, it can reduce the number of paths to one, the concretely executed path. Then, it can flip one of the constraints to explore a different branch, commonly starting from the last one. Concolic execution can treat complex functions, such as hash functions, as concrete values, unlike symbolic execution. Still, running an SMT solver requires significant processing time, and using concrete values might restrict the solver so that it cannot find a concrete solution for the collected path constraints.

3 Comparison

3.1 Comparison space

In order to compare different approaches state-of-the-art concolic executors are employing, we selected four design choices we believe are most impactful on the resulting performance, namely:

Target type: What targets can be analyzed?

Constraint Collection: How are path constraints collected?

Constraint Solving: How are the resulting constraints solved?

Interaction with external environment: How does the engine interact with external dependencies, e.g., libraries?

We note that most works implement further strategies that fall outside these categories.

3.2 Approaches under analysis

Fuzzer (by avg. score)	Avg. Score
fuzzolic-aflplusplus-z3	98.67
symqemu-aflplusplus	97.51
fuzzolic-aflplusplus-fuzzy	97.46
aflplusplus-cmplog-double	96.82
symcc-aflplusplus-single	96.75
eclipser-aflplusplus	96.27
aflplusplus-qemu-double	94.72

Table 1. Fuzzbench results of a variety of hybrid fuzzers and *AFL++*.

In order to compare the recent advances in symbolic execution-based approaches, we review the following implementations as they competed with other state-of-the-art fuzzers on Fuzzbench[1] (c.f. Figure 1): *SymCC* [18], *SymQemu* [19], and *Fuzzolic* [4]. *Fuzzbench* is a popular fuzzing benchmarking service where each tool is evaluated on a variety of open source software, wherein after each tool is assigned a normalized score [14]. Furthermore, we decided to include *SymSan* as it was not released at the time of this benchmark although it outperformed the approaches in their own evaluation [6]. Finally, since many design decisions of this selection are influenced by *QSym* we also include it in our review [25].

[1] https anonysp2022.github.io/#experiment-summary

3.3 Target type

The approaches under analysis use a selection of different tools to instrument or trace the target program in order to gather symbolic constraints. These underlying tools pose the main limitations of targets for the corresponding concolic execution. Our selection of approaches used mainly three different techniques. First, *QSym* uses Intel PIN [13], a dynamic binary instrumentation tool that can only target binaries run on Intel processors. Secondly, *SymCC* and *SymSan* instrument the code at compile-time, hence, requiring source code access. While these tools are currently mainly implemented for C/C++ programs, as they leverage the LLVM compiler framework, they should be easily expandable to any LLVM-supported language. Finally, *SymQemu* and *Fuzzolic* instrument theIR of QEMU,Tiny Code Generator (TCG). Thus, they are able to target any executable supported by TCG, given negligible modifications [19].

3.4 Constraint Collection

Inspired by *QSym*, all approaches use a backend, which collects the constraints and then tries to solve them. Additionally, the backend can apply optimizations, such as deduplication, to improve performance and is easily interchangeable, enabling flexible experimentation with different backends. *QSym* sacrificed high implementation complexity to execute at the instruction level symbolically. This design decision allows for finer-grained optimizations and potentially less overhead since instructions that contain no symbolic inputs can be executed natively. *SymCC* integrated the creation of the expression into the instrumented binary as an LLVM compiler pass, thus, removing the dependency of a relatively slow interpreter. The compiler pass adds calls to a runtime library. These calls pass symbolic expressions to the runtime, which then tries to solve them. This design allows interchangeable usage of different backends, such as *QSym* or *Z3*. *SymSan* expands upon this idea by improving performance. Inspired by optimizations from Address sanitizer (ASAN) [22], *SymSan* is able to achieve constant-time lookup of symbolic variables, which were done in logarithmic time by *SymCC*. This, among other optimizations, leads to a non-negligible performance increase over *SymCC*. Regarding the concolic execution of binaries, *SymQEMU* and *Fuzzolic* follow the same approach by instrumenting the IR of QEMU at runtime. Generally, both follow the compile-time approach by instrumenting certain IR func-

tions with symbolic expression-building and calls to the solving runtime. As noted by the authors of *Fuzzolic*, both tools have three main differences. First, *Fuzzolic* decouples the expression building component and solving component into two different processes, which is a design choice with no further reasons given and, as they point out, is challenging to evaluate. Secondly, *Fuzzolic* instruments the IR on a basic block level, potentially enabling further optimization by sacrificing implementation complexity. Finally, *Fuzzolic* offers three different fuzzing-inspired modes, which increased performance in their tests.

3.5 Constraint Solving

While much progress occurred focusing on constraint collection, recently, new approaches for solving those constraints have been proposed in conjunction with the release of our approaches under analysis. As shown by Poeplau, much of the total execution time of concolic executors is spent in the solving component [18]. Commonly, concolic execution used the *Z3* solver to satisfy the collected constraints. However, *QSym* modified *Z3* to apply hybrid fuzzing-specific optimizations. Newer approaches can subsequently use this newly created backend, although this may lead to certain incompatibilities, such as pointed out by the authors of *SymSan*. While *SymCC* and *SymQEMU* rely on plain *Z3* or *QSym* as their backend, the *Fuzzolic* and *SymSan* authors presented novel solvers, namely, *fuzzy-sat* and *jigsaw* [3, 7]. *Fuzzy-Sat* borrows techniques from the fuzzing domain to trade accuracy for higher throughput. This is achieved by leveraging a new search algorithm, offering different modes, and incrementally solving the constraints. On the other hand, *jigsaw* uses a gradient descent-based search algorithm as suggested by *angr* [23] and increases efficiency by following a JIT-based approach to evaluate expressions. As their benchmarks have shown, this leads to a throughput improvement by multiple orders of magnitude.

3.6 Interaction with external environment

QSym treats the external environment as black boxes and concretizes all interactions. Altogether, this means that *QSym* executes everything symbolically until the system call level. This prevents it from using erroneous or incomplete models of external functions. However, this may pose a violation of process boundaries. *SymCC* concretizes all uninstrumented code

to avoid tricky situations, such as binary-only libraries. However, *SymCC* pays implementation complexity and increased error-proneness by providing standard wrappers around certain functions of the C standard. The authors point out that it is possible to compile an instrumented version of the LLVM standard library *libc++* for convenience reasons. Analogous, *SymSan* provides wrappers for common standard operations. *SymQEMU* symbolically executes all instructions up until the system call level, as only user mode is emulated. Finally, *Fuzzolic* follows this approach and additionally implements models of common routines, such as *memcpy*, to track the effects.

4 Discussion

Previously, traditional "dumb" fuzzers have been dominating bug discovery benchmarks. This trend has been overthrown by our approaches under analysis as suggested by recent *Fuzzbench* evaluations. We enumerated the different approaches; however, pointing out the strengths and weaknesses is difficult as most techniques improving precision introduce a performance hit and vice versa, and the evaluations, such as those performed on Fuzzbench, merely signal which fuzzer performs best on average. Hence, a rigorous study is necessary to compare the different approaches and extract their respective strengths and weaknesses. This study could allow for the distillation of the techniques inducing the actual performance benefit, e.g., the authors of *SymQEMU* reused the backend of *SymCC* to compare only the effects of using dynamic instrumentation, while *Fuzzolic* made changes to the frontend as well as the backend, hence, it is unclear which techniques resulted in the performance benefit. Furthermore, it is unclear which concrete scenarios each tool is excelling at. Perhaps, a specific tool performs exceptionally well on cryptographic checks while another tool performs better on parsing steps. Concluding, we propose two future research ideas. First, a systematic comparison of our comparison space, similar to Poeplau and Francillon [17], where each technique is tested in isolation, and each combination is evaluated since a particular technique may perform better in a general scenario, although a certain combination of two techniques may perform even better in synergy. This way, combinations, such as *SymSan* constraint collection, can be used with the *fuzzy-sat* solver. Second, a study to investigate where the respective strengths of each approach are. This could possibly be done by

triaging all bugs found or missed, classifying those bugs, and observing which classes are easily found by which tool. Furthermore, such a survey could also compare inherently different approaches, such as taint-based approaches [20] or inference-based approaches [1]. Such research could also hint towards future research into what code paths are currently difficult to cover for current state-of-the-art approaches.

While Poeplau and Francillon remarked that using a compilation-based instrumentation leads to simpler constraints [17], it is unclear whether this only occurs for their approach *SymCC* or it is generally the case and also occurs when using a different compilation-based approach, such as *SymSan*. Hence, we propose to redo the study with current state-of-the-art tools.

5 Conclusion

This paper has reviewed recent concolic executors for hybrid fuzzing and identified their core differences across four design criteria. We suggested possible studies to distill the advantages and disadvantages of the proposed techniques and possibly compare hybrid fuzzing with concolic execution with fuzzing methods leveraging other analysis techniques.

References

[1] Cornelius Aschermann, Sergej Schumilo, Tim Blazytko, Robert Gawlik, and Thorsten Holz. Redqueen: Fuzzing with input-to-state correspondence. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[2] Thanassis Avgerinos, Alexandre Rebert, Sang Kil Cha, and David Brumley. Enhancing symbolic execution with veritesting. In *Proceedings of the 36th International Conference on Software Engineering*, pages 1083–1094, 2014.

[3] Luca Borzacchiello, Emilio Coppa, and Camil Demetrescu. Fuzzing Symbolic Expressions. In *Proceedings of the 43rd International Conference on Software Engineering*, ICSE '21, 2021.

[4] Luca Borzacchiello, Emilio Coppa, and Camil Demetrescu. Fuzzolic: mixing fuzzing and concolic execution. *Computers & Security*, 108:102368, 2021.

[5] Oliver Chang. Taking the next step: Oss-fuzz in 2023, Feb 2023.

[6] Ju Chen, WookHyun Han, Mingjun Yin, Haochen Zeng, Chengyu Song, Byoungyoung Lee, Heng Yin, and Insik Shin. SYMSAN: Time and space efficient concolic execution via dynamic data-flow analysis. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2531–2548, Boston, MA, August 2022. USENIX Association.

[7] Ju Chen, Jinghan Wang, Chengyu Song, and Heng Yin. Jigsaw: Efficient and scalable path constraints fuzzing. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 18–35. IEEE, 2022.

[8] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[9] Andrea Fioraldi, Dominik Maier, Heiko Eißfeldt, and Marc Heuse. Afl++ combining incremental steps of fuzzing research. In *Proceedings of the 14th USENIX Conference on Offensive Technologies*, pages 10–10, 2020.

[10] Patrice Godefroid, Nils Klarlund, and Koushik Sen. Dart: Directed automated random testing. In *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, pages 213–223, 2005.

[11] Patrice Godefroid, Michael Y Levin, and David Molnar. Sage: whitebox fuzzing for security testing. *Communications of the ACM*, 55(3):40–44, 2012.

[12] James C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, jul 1976.

[13] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, page 190–200, New York, NY, USA, 2005. Association for Computing Machinery.

[14] Jonathan Metzman, László Szekeres, Laurent Simon, Read Sprabery, and Abhishek Arya. Fuzzbench: An open fuzzer benchmarking platform and service. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, page 1393–1403, 2021.

[15] Barton P. Miller, Lars Fredriksen, and Bryan So. An empirical study of the reliability of unix utilities. *Commun. ACM*, 33(12):32–44, dec 1990.

[16] Matt Miller. A proactive approach to more secure code. BlueHat IL, 2019.

[17] Sebastian Poeplau and Aurélien Francillon. Systematic comparison of symbolic execution systems: Intermediate representation and its generation. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 163–176, New York, NY, USA, 2019. Association for Computing Machinery.

[18] Sebastian Poeplau and Aurélien Francillon. Symbolic execution with symcc: Don't interpret, compile! In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 181–198, 2020.

[19] Sebastian Poeplau and Aurélien Francillon. Symqemu: Compilation-based symbolic execution for binaries. In *NDSS*, 2021.

[20] Sanjay Rawat, Vivek Jain, Ashish Kumar, Lucian Cojocar, Cristiano Giuffrida, and Herbert Bos. Vuzzer: Application-aware evolutionary fuzzing. In *NDSS*, volume 17, pages 1–14, 2017.

[21] Sergej Schumilo, Cornelius Aschermann, Ali Abbasi, Simon Wör-ner, and Thorsten Holz. Nyx: Greybox hypervisor fuzzing using fast snapshots and affine types. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[22] Konstantin Serebryany, Derek Bruening, Alexander Potapenko, and Dmitry Vyukov. Addresssanitizer: A fast address sanity checker. In *USENIX ATC 2012*, 2012.

[23] Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Audrey Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, and Giovanni Vigna. SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis. In *IEEE Symposium on Security and Privacy*, 2016.

[24] Nick Stephens, John Grosen, Christopher Salls, Andrew Dutcher, Ruoyu Wang, Jacopo Corbetta, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Driller: Augmenting fuzzing through selective symbolic execution. In *NDSS*, volume 16, pages 1–16, 2016.

[25] Insu Yun, Sangho Lee, Meng Xu, Yeongjin Jang, and Taesoo Kim. QSYM: A Practical Concolic Execution Engine Tailored for Hybrid Fuzzing. In *Proceedings of the 27th USENIX Security Symposium (Security)*, Baltimore, MD, August 2018.

Kubernetes for greener environment

Phong Tran
phong.tran@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

Energy consumption in cloud computing is increasing, and the amount of energy used in cloud computing is estimated to increase 15 times by 2030 [6]. Therefore, this paper discusses and analyzes different approaches using Kubernetes to address the rise in energy consumption in cloud computing. The most common issue that cause the energy usage to rise in cloud computing is the inefficient allocation of resources, and the approaches discussed aim to solve this common issue. The approaches discussed in this paper consist of workload scheduling, server consolidation, and mobility awareness. Although all these approaches are efficient in solving the issue of resource allocation in cloud computing, workload scheduling is the best approach since the other approaches contain many disadvantages regarding setting up and operating. As the main goal is to improve the energy efficiency in cloud computing, the tools that do not provide environmental benefits should be deprecated, and more projects to address the environment should be funded by multiple entities.

KEYWORDS: Kubernetes, energy efficiency, resource allocation, workload scheduling, server consolidation, mobility awareness

1 Introduction

Before cloud computing was introduced, software and services were deployed in physical data centers. Since 2006, cloud computing has become popular and many software and services were deployed using cloud computing [6]. Cloud computing has provided various business models, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Moreover, cloud computing leverages universal access to data center facilities.

Although cloud computing has provided significant benefits, it has many environmental problems. It is reported that the energy consumption of data centers is at 200 TWh in 2016, and that number is expected to rise to 2967 TWh in 2030 [6]. Based on the survey conducted by Stücke [11], the alarming amount of energy consumption is due to the complexity of cloud computing makes it difficult to estimate the number of resources needed. Moreover, the survey points out that it is a common pattern to unnecessarily allocate more resources to ensure that software and services are performing well. To address the energy consumption issue, a tool such as Kubernetes (K8s) is introduced to better allocate resources, thus preventing over allocating unnecessary resources of different services.

In order to evaluate the energy saving capabilities of K8s, this paper discusses how K8s is used to enhance the energy efficiency. In addition, this paper summarizes, discusses, and analyzes approaches conducted by multiple studies to help promote energy efficiency and carbon emissions reduction using K8s technologies.

This paper is organized as follows. Section 2 provides some background information about K8s and its benefits. Section 3 discusses how K8s is used to schedule and consolidate servers to provide better energies efficiency based on different researches. Section 4 analyzes and discusses about the different approaches presented in Section 3. Section 5 provides concluding remarks of this paper.

2 Kubernetes

This section introduces K8s as a tool and related concepts of K8s, such as clusters, nodes, and pods.

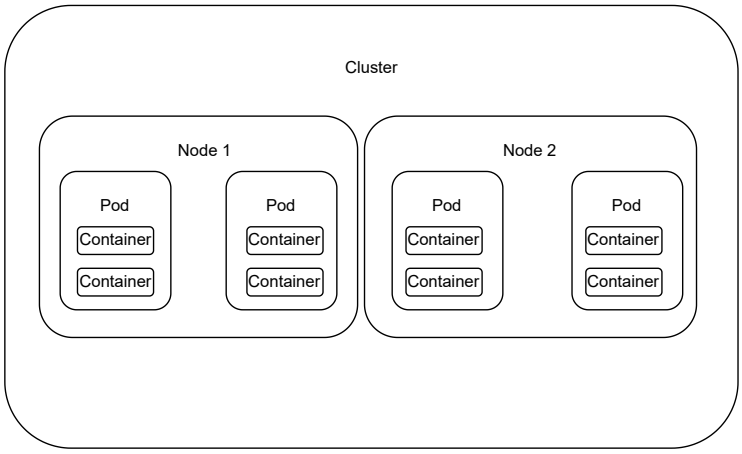


Figure 1. Cluster, nodes, and pods relationship

2.1 K8s as a tool

K8s is an open-source container orchestration platform that allows the automation of deploying, managing, and scaling of containerized applications [9]. K8s was found and developed by engineers at Google. K8s allows DevOps engineers to deploy and manage application better than many other platform since K8s provides certain functionalities that abstract many manual processes. For example, K8s can orchestrate containers across multiple hosts, and most importantly, K8s can scale containerized apps and resources on the fly. The ability to scale resources on the fly is important because during peak time, the application has enough resources to operate with the least chance of operation failure. In contrast, during idle time, it is also important to deallocate unnecessary resources to reduce as much cost as possible for companies. K8s is built on the concept of clusters, nodes, and pods [12].

2.2 Clusters, nodes, and pods

Clusters are the main building blocks of the K8s architectures, and nodes are part of clusters. Each cluster has one master node, which contains multiple worker nodes that deploy, run, and manage container applications. A cluster contains one or many pods, and a pod is a group of containers that share the same computing resources and network. Should a pod receive more traffic than it can handle, the cluster will create another pod in other nodes within the cluster. The last concept of K8s is Deployment, which specifies the number replicas of pods should be created and run on the cluster. The relationship between the cluster, nodes, and pods is illustrated by figure 1.

3 Energy Efficiency of K8s

K8s is a tool that provides significant benefits to DevOps engineers, and it is also a major cost saving opportunity to many companies since it is inexpensive to deploy different services using K8s. Moreover, K8s is used to enable energy efficiency when deploying and managing services in the cloud. According to Mustafa [8], Amazon Web Service (AWS) has increased the CPU utilization from 8% to 70% after switching from virtual machines to K8s. This enables Nordstrom to scale down their infrastructure with only 1/10 of the computing power they used before AWS switched to K8s.

Energy consumption in cloud computing can be caused by multiple issues; therefore, it is important to address the correct issue. The most common issue is the inefficient allocation of resources in cloud computing. When the allocation of resources is inefficient, this allows multiple unused resources to be active; when the number of active unused resources increases, a substantial amount of energy is consumed. To address this issue, several studies has been conducted to improve the energy efficiency in cloud computing using K8s based approaches, such as workload scheduling, server consolidation, and mobility awareness [1, 5, 8, 2, 4]. All of these approaches attempt to improve the allocation of resources in cloud computing, specifically inside a K8s cluster.

3.1 Workload scheduling

Workload scheduling in K8s is the concept of assigning pods to specific nodes while balancing utilization resources among them. Sobol [10] mentions that the simplest workload scheduling configuration that can be given to K8s is **nodeName**, which is the name of the node. However, this configuration approach can lead to having non-functional pods, such as an unexisting node name. Another powerful configuration can be given to K8s is **nodeSelector**, which defines the specification of a node, including disk type and node affinity. Node affinity is a setting that can specify the rules and stages of a node that a pod should be assigned to.

Green region workload scheduling

Using workload scheduling, James and Schien [5] built a conceptual scheduler that helps reduce carbon emissions to the environment. The idea of the concept is to extend the default behavior of K8s scheduler to consider

the carbon emissions based on a different numbers of factor, such as the carbon intensity of certain regions in world, the air temperature, and the solar irradiance.

In order to implement this concept, James and Schien [5] developed an algorithm that determines the greenest region based on the carbon intensity from a known list of regions. If there exists multiple regions with the same carbon intensity, then air temperature is another parameter this algorithm considers when selecting the greenest region. After the greenest region has been determined using this algorithm, the program sends the data to the cloud K8s or Infrastructure as a Service (Iaas) management API to provision a Resource Group at a dedicated cloud provider from that region, then the program verifies the successful state of the request. After the provisioning request has been confirmed successful, it commonly takes 10 minutes for a new cluster to be provisioned and for all credentials to be agreed upon. In addition, it takes one or two minutes for all components in K8s to be in **Ready** status. To address the time it takes for the provisioning of new cluster, the scheduler polls the status of the component of the new cluster at a regular interval, and once the **Ready** status has been confirmed, the necessary **Deployments** are executed. Once all the **Deployments** are executed, the scheduler deletes all Resource Group from the previous provisioned cluster.

Workload scheduling for HPC environment

Another approach of utilizing K8s workload scheduling is to create a scheduler that is suitable for Higher Performance Computing (HPC) workloads. Dakic et al. [1] state that HPC environments has been used extensively in many virtual machines in different cloud environments because of the need of computation power, agility, and manageability. However, virtual machines consume a substantial amount of resources because in order to operate a virtual machine, a minimum constant amount of memory in a computer must be allocated to operate a virtual machine, although the amount of memory needed is significantly lower than the minimum amount of memory. In order to address this issue, Dakic et al. [1] developed a K8s scheduler that can automatically allocate the workload to the nodes which contain higher computing power resources based on the amount of workload.

Heterogeneous HPC environments are based on all technologies, such as x86, ARM, RISC-V, and GPU. Therefore, setting up multiple K8s nodes

is the solution to running workload in HPC environments, and each node has specifications related to their respective technologies. Dakic et al. [1] setup four K8s nodes based on x86 and ARM technologies, and a scheduler that ensures that a node has enough computing power capabilities to operate the workloads. If a node fails, it allocates immediately to other nodes, and most importantly, it takes the pricing per kWh, carbon footprint, and energy consumption as a variable when deciding which nodes to use. As a result, the Advance RISC machines (ARM) architecture performs exceptionally well regarding performance per Watt, although ARM platform is an inexpensive platform. The x86 CPU based architecture does not perform as well as the ARM architecture, it is still a mature platform and should not be discarded. The scheduler can also be scaled up further with more nodes, and it can also be coupled with other CPU architectures, such as RISC-V. Dakic et al. are confident about their work and they believe that their solution is going to be both energy efficient and high performance.

3.2 Server Consolidation

Server consolidation is a concept of cloud computing which refers to the process of combining multiple servers into a single, more powerful cluster of servers [7]. This method has been used by multiple enterprises to get the benefits of reduced space, power, and administrative requirements to reduce cost, workloads, and power consumption [2]. Although server consolidation reduces the resources used, it increases the complexity of data configuration, along with additional overhead when making requests to multiple servers.

In order to address the issue mentioned above, Dewi et al. [2] use K8s to automate the process of server consolidation. K8s already has its own mechanism to horizontally scale pods based on the amount of concurrent users. This mechanism is known as K8s Horizontal Pods Auto-Scaling (KHPA). The mechanism can also be configured by giving a certain number of parameters to determine the maximum and the minimum numbers of pods, the CPU utilization, and the permission to allocate/deallocate resources to avoid the effects of instability. Dewi et al. [2] setup a microservice application with 8000 users to perform multiple concurrent HTTP requests to the services in the application. The services in the microservice are also created with single server setting and multiple servers with scalability setting. The result shows that the CPU Usage Pods of multiple

servers setting are drastically better than the single server setting, which translates to less energy consumed when running the test in multiple servers setting. Although the response time of the single server setting is better than the multiple servers setting, the latency is not significant [2]. The reason for the latency is due to the overhead of scaling containers in multiple servers setting.

3.3 Mobility awareness

Proximity tracking and assigning pods to the nodes closest to the location of a user are also issues when implementing K8s to manage software deployments in the cloud. Ghafouri et al. [4] address these issues in mobile edge computing when making K8s compatible with cloud decentralize platforms. The mobility constantly changes the proximity of the users to the services, which leads to having active nodes in the K8s cluster that do not use their full capacity. Ghafouri et al. developed a framework named Mobile-kube, which reduces the latency of mobile edge computing devices while maintaining the energy consumption at a reasonable level. The framework is backed by a reinforcement machine learning method known as Importance Weighted Actor-Learner Architecture (IMPALA). According to Ghafouri et al. [3], IMPALA is not only efficient for training data in a single machine, it is also efficient for training data in thousands of machines without sacrificing data efficiency and training stability. Using IMPALA along with another neural network, such as critic, can help reduce variance when calculating the reward [4]. From the reward, the state of the environments can be updated. Ultimately the goal of Mobile-kube is to determine where is the best location to host a service based on the the location of the user, and that decision has to be made automatically.

In order to implement and test the framework, Ghafouri et al. [4] designed a system where a device is connected to a service at a certain location. The system consists of a K8s cluster that contains nodes and pods, a mobility simulator which is a Python script assigning the users to the nodes, and a controller which wraps the information received from the K8s cluster and the mobility simulator into an environment referred to as OpenAI gym environment. The gym environment calculate the reward based on the information it is given, then the reward is passed to the reinforcement learning agent, which then decides the next placement of pods in the nodes. After that, using K8s Python API, the placement is passed back to K8s cluster and the pods will be placed into the new nodes.

As a result, with the same energy consumption, the Mini-kube framework is able to reduce 43% of user latency. The test is also conducted using two other reinforcement learning methods: Policy Gradient (PG) and Proximal Policy Optimization (PPO). When comparing the results, the IMPALA methods achieve the best energy performance with different number of users, and it is indicated by the number of empty servers between the different number of users. From this result, Ghafouri et al. are confident that the Mobile-kube framework will enable the most optimal performance to mobile edge computing while maintaining the energy consumption at the most sufficient level.

4 Analysis and Discussion

Different methods and approaches can enable better energy efficiency when deploying and maintaining software services in the cloud using K8s. The approaches mentioned in the previous section solve a common issue in cloud computing, which is the efficiency of resource allocation in cloud computing.

4.1 Workload scheduler

The workload scheduler approach is the best approach when maximizing the energy usage while maintaining the performance of the services. During idle time, the cluster can assign pods to the nodes with the lower specification to reduce the energy usage as much as possible, and only assign the pods to higher performance nodes during peak time. The number of nodes and pods can also be configured in workload scheduling to maximize performance if it is needed. This is the most flexible approach, as workload scheduling is also efficient when using it in heterogeneous HPC environment.

4.2 Server Consolidation

Server consolidation is also an efficient approach when maximizing energy efficiency when deploying and maintaining services in the cloud. The CPU Usage Pods is significantly lower when handling user requests in multiple servers environment. However, with the amount of latency when scaling containers in multiple servers setting, server consolidation may not be beneficial when the performance of the application is rely on

the response time to the users. To address this issue, workload scheduling is the better approach when the latency of the application when using workload scheduling is tolerable.

4.3 Mobility awareness

Mobility awareness approach is an efficient approach when handling the proximity of user location in mobile edge computing. It certainly reduces the latency of the user when determining the nodes with the closest location to the user while maintaining the energy efficiency. However, since it is using machine based approach, the algorithm requires training by feeding data into the algorithm; it is challenging to use real world user data to train the algorithm. The algorithm needs a large number of test subjects to train, and the mobility awareness approach can be costly to setup initially. Another problem with machine learning based approach is that the test data is not as diverse or varied as the real world data. Therefore, the lack of diversity in test data can lead to many cases of false positive or false negative when predicting the results, which can cause the mobility awareness approach to allocate the pods into the wrong nodes or into the nodes that are not the closest to the current location of the user. Although mobility awareness is an innovative approach to improve the energy efficiency with the use of machine learning, it is prone to error and not the most cost efficiency approach.

5 Conclusion

This paper has reviewed three approaches using K8s to maximize the energy efficiency in cloud computing. The main goal of the approaches is to improve resource allocation mechanism in cloud computing. Although the approaches previously mentioned are not fully optimized for all use cases of many enterprises, they are still efficient and the enterprises should consider implementing them into their deployment infrastructures. Regardless of which approach is taken into implementation, the enterprises will reduce the carbon emission and improve the energy efficiency when more services and software are deployed into the cloud. In the future, with the constant improvement of technology, researchers will be able to conduct more approaches using K8s to further maximize the energy efficiency; the legacy tools that do not provide environmental benefits will be

deprecated. Furthermore, since the environment is an issue that concerns the world, governments should fund projects aimed at improving the energy efficiency in cloud computing to ensure the best possible outcome.

References

[1] Vedran Dakic, Mario Kovac, and Jasmin Redzepagic. Optimizing kubernetes performance, efficiency, and energy footprint in heterogenous hpc environments. *DAAM*, 2021.

[2] Lily Puspa Dewi, Agustinus Noertjahyana, Henry Novianus Palit, and Kezia Yedutun. Server scalability using kubernetes. In *2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON)*, pages 1–4, 2019.

[3] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. Jun 2018.

[4] Saeid Ghafouri, Alireza Karami, Danial Bidekani Bakhtiarvand, Aliakbar Saleh Bigdeli, Sukhpal Singh Gill, and Joseph Doyle. Mobile-Kube: Mobility-aware and Energy-efficient Service Orchestration on Kubernetes Edge Servers. 7 2022.

[5] Aled James and Daniel Schien. A low carbon kubernetes scheduler - ceur-ws.org. https://ceur-ws.org/Vol-2382/ICT4S2019$_paper_28.pdf$, 2019.

[6] Avita Katal, Susheela Dahiya, and Tanupriya Choudhury. Energy efficiency in cloud computing data centers: A survey on software technologies. https://link.springer.com/article/10.1007/s10586-022-03713-0, Aug 2022.

[7] Anja Libo. Server consolidation in cloud computing. https://www.geeksforgeeks.org/server-consolidation-in-cloud-computing/, Jan 2023.

[8] Akhterul Mustafa. Reducing carbon emissions through kubernetes - labs.sogeti.com. https://labs.sogeti.com/reducing-carbon-emissions-through-kubernetes/, Sep 2022.

[9] Red Hat IT Professionals. What is kubernetes? https://www.redhat.com/en/topics/containers/what-is-kubernetes, Mar 2020.

[10] Ron Sobol. A deep dive into kubernetes scheduling. https://granulate.io/blog/a-deep-dive-into-kubernetes-scheduling/, Mar 2022.

[11] Jan Stücke. Kubernetes as a sustainability tool. https://www.stormforge.io/blog/kubernetes-sustainability-tool/, Jul 2022.

[12] Sai Vennam. Kubernetes architecture. https://www.ibm.com/topics/kubernetes, 2023.

Security and Privacy in the Metaverse

Praewpiraya Wiwatphonthana
praewpiraya.wiwatphonthana@aalto.fi

Tutor: Mario Di Francesco

Abstract

The Metaverse is an emerging concept that promises to integrate virtual reality experiences with our daily lives, creating new opportunities for social interaction and development. As the Metaverse becomes more popular and widely adopted, the need for addressing security and privacy concerns becomes even more critical. The key technology that enables access to the Metaverse is visualization technology, and thus it is the primary focus of our review. This paper analyzes the vulnerabilities that Metaverse systems inherit from existing technologies, including data breaches, identity theft, social engineering, and unauthorized access, and explores how these risks can be amplified in a virtual environment. Additionally, this paper highlights the need for novel authentication and authorization mechanisms. Overall, this study aims to provide insights into the current state of security and privacy in the Metaverse and to suggest ways to ensure secure and implement effective security solutions in this emerging technology landscape.

KEYWORDS: security, privacy, metaverse, visualization technology

1 Introduction

The word “Metaverse” is a combination of prefix “meta” with the meaning of transcending and suffix “verse” which implies the universe. There are different descriptions of this beyond-universe technology concept depending on its context. In short, the concept of the Metaverse refers to a virtual world that is immersive, interactive, and co-created by its users [5, 13, 22]. The Metaverse is not just a single, centralized virtual environment but rather a decentralized network of virtual spaces that are connected and interoperable [20, 22]. Lee et al. [13] divide metaverse development into three phases: (I) digital twins, (II) digital natives, and the last phase (III) co-existence as shown in Figure 1.

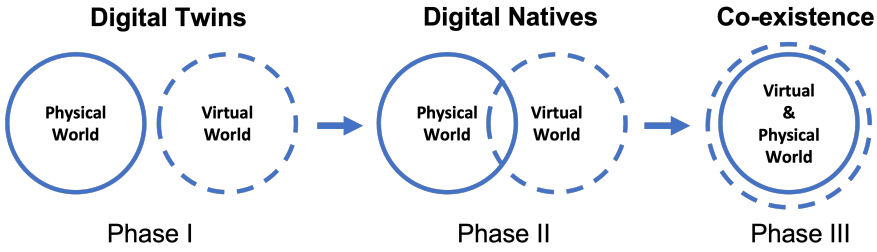


Figure 1. Metaverse Development Phases

The first phase called *digital twins*, mirrors the physical reality and duplicates its property, such as the motion and function of users or objects, into virtual environments. The second phase, known as the *digital natives*, focuses on the creation of native content within the virtual world. The content can be native only to the virtual world or linked to its physical counterpart, influencing innovation in both worlds and creating an intersection between them. This process connects the ecosystems of both worlds, enabling the development of new forms of culture, economy, and regulation that support the creation of physical goods and intangible contents. In the final phase, *co-existence*, both worlds co-exist and inter-operate with the physical world. The virtual world turns into a self-sustaining and persistent one. With the seamless integration of the two worlds, the objects and entities that do not exist in the real world now exist in virtual reality space, making the scope of the virtual world larger than the physical world. The space is built by combining various technologies, such as extended reality (XR), artificial intelligence (AI), and blockchain. The development of the Metaverse has presented vast opportunities and potential applications across multiple industries, including

entertainment, socialization, and education. However, with the Metaverse gaining more popularity and being increasingly integrated into our lives, there is a growing concern regarding security and privacy. These concerns arise from managing vast amounts of data, pervasive user profiling, and technologies that are inherently vulnerable [22]. Existing threats can be amplified in the virtual world, opening up new avenues for crimes. Examples of such incidents include identity theft [7, 12] and the hijacking of wearable devices [6].

Given these challenges faced by the Metaverse, it is crucial to review existing research exploring these security and privacy issues. This paper aims to provide an overview of recent publications on the challenges and solutions for security and privacy in the Metaverse. The review covers the Metaverse concept, related technologies, and potential threats, as well as existing solutions and future research directions. Specifically, the review focuses on interactive visualization technology, a key component of the Metaverse.

The paper is organized as follows. Section 1 provides an overview of the Metaverse concept and its challenges. Section 2 covers key technologies used in the Metaverse, while Section 3 focuses on visualization and interactive technologies. Section 4 reviews current security threats. Section 5 reviews potential solutions to address these issues. Finally, Section 6 summarizes the main findings and discusses potential future research directions.

2 Key technologies in Metaverse

The key technologies that drive the development of the Metaverse are Interactive technology, Cloud Service, AI technology, and Blockchain technology [5].

Visualization and Interactive technologies: In the Metaverse, visualization and interactive devices play a key role in offering users more immersive and interactive experiences in virtual environments, not only as an access point but also linking the user interaction to the virtual world. The movement of the user and interaction are captured by motion tracking and then projected into the virtual world. This technology allows users to navigate and interact with virtual environments in a more intuitive and realistic way [5, 19].

Cloud services: Cloud services enable the Metaverse to store, process, and distribute large amounts of data and computational resources, allowing it to scale and meet user demand. However, when Metaverse physics emulation and graphical rendering are implemented using a centralized cloud-based approach, the design suffers from several drawbacks caused by the high network latency, which results in low-quality visualization of the graphic. To address this problem [9], the concept of distributed computation is introduced to the Metaverse. The bottleneck computational process is pushed toward end-users instead.

AI technology: AI plays a major role in the processing of motion capture, system speech recognition, computer vision, and sentiment analysis in the Metaverse. This technology makes the Metaverse more dynamic and responsive, allowing for a more personalized and interactive experience for users. Moreover, AI facilitates the system infrastructure tasks, such as automatic resource allocation, traffic off-loading, and fault detection. The tasks improve the infrastructure reliability and system performance [10].

Blockchain technology: Blockchain technology enables a decentralized database to the Metaverse. In addition, Blockchain provides a secure and decentralized economic framework [15] for the exchange of digital assets and currencies within the Metaverse. While the regulation of the economic system creates a link between virtual space and reality, trust and transparency in virtual transactions are also established in order to ensure a secure and reliable experience for the users.

3 Visualization and Interactive technologies

In visualization and interactive technology, namely Extended Reality (XR) and Head-Mounted Displays (HMDs) give the user a visual representation of virtual content in the Metaverse. XR technology has emerged as a revolutionary innovation that integrates the physical and digital worlds, providing a seamless and immersive user experience. By combining the concepts of Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), XR technology provides an access point and an underlying infrastructure for the development of the Metaverse.

Virtual Reality (VR): Virtual Reality is an immersive technology that creates a simulated environment where users can interact with computer-

generated objects or scenarios. VR headsets are commonly used to fully immerse users in the virtual world, providing a sense of presence and realism [19, 23].

Augmented Reality (AR): Augmented Reality is an interactive technology that combines virtual reality with reality by superimposing computer-generated images onto a real-world environment [4, 19]. AR technology enhances the real-world experience by overlaying digital content, such as images, videos, or 3D models, onto the physical environment.

Mixed Reality (MR): Mixed Reality is a technology that combines elements of both VR and AR to create an interactive experience that merges the physical and virtual worlds. MR technology enables users to interact with digital objects in the real world, creating a seamless integration between virtual and physical environments [8, 19].

4 XR technology threats

The Metaverse, as a convergence of various technologies, poses several security threats. In the realm of Extended Reality, potential security threats share commonalities in their methods and goals. This section presents a selection of such threats, classified based on their shared characteristics.

4.1 Authentication-related threat

To access Metaverse platforms, users must authenticate their identity to ensure that they are legitimate and to avoid identity theft as well as impersonation attacks; hence the security of this process is crucial.

Keystroke logging attacks: A username and password combination is frequently used in conventional authentication methods. In the Metaverse, the user is often presented with a virtual keyboard to enter a username and password. The keyboard allows them to either air-tap using hand gestures or use a controller to point and click on virtual key. Meteriz et al. [14] demonstrated that hand gesture patterns could be exploited to infer keystrokes. The attack takes advantage of the fact that hands tend to follow specific patterns when individuals type in the air. Specifically, the attacker captures the traces of the hand movements of

the victim and subsequently employs keystroke detection and key identification, then generates a set of input inferences. In addition, Al et al. [3] implemented VR-Spy, a technique for recognizing virtual keystrokes that makes use of WiFi channel state information (CSI). The main concept behind the methods is side-channel information of fine-grained hand motions, which has a unique gesture pattern in the CSI waveforms.

4.2 Unauthorized access

A type of attack in which an attacker gains access to a system without permission, which can lead to concerns such as data and identity theft in the Metaverse.

Impersonation attacks: In the Metaverse, the usage of digital avatars can make it simple for attackers to impersonate legitimate users by stealing their authentication credentials. This can lead to identity theft and impersonation attacks where the attacker fools a victim into believing that they are interacting with a legitimate user. Attackers can also use recordings of users' gestures and voice commands captured by XR applications and devices to impersonate users [7].

Man-In-The-Room (MitR) attacks: Vondrek et al. [21] developed a new attack, the Man-In-The-Room (MitR). The attack can target any public or private room in the infected virtual environment and take control. The study shows that the MitR attack, along with the VR worm, had successfully accessed the target room, hid their presence, and modified the environment without the permission of users.

4.3 Data breaches

The Metaverse involves the creation and sharing of a large amount of data, including personal information, behavioral data, and virtual assets. These types of data are valuable and attractive to attackers.

Eavesdropping attacks: This attack is a potential consequence of unauthorized access. For example, in MitR attack [21], while the attacker is within the compromised environment, the attacker initiates an eavesdropping attack, and eavesdrop on the screen and microphone of the victims. Moreover, Shi et al. [18] designed an eavesdropping attack named Face-Mic, which exploits motion sensors in AR or VR headsets to capture

facial dynamics during speech. This technique can infer sensitive information, such as speaker gender, identity, and speech content, from live speech by capturing facial muscle movements and bone-borne vibrations.

4.4 Social engineering

In the Metaverse, social engineering attacks are becoming increasingly prevalent due to the growing number of digital interactions between users. Attackers leverage psychological manipulation techniques to trick people into giving sensitive information or performing actions that may compromise their security.

VR Phishing: Fraudulent activities in the Metaverse take on many different forms, such as imposter websites, fake social media accounts, deceptive emails, and bot-controlled messaging. While Metaverse environments are primarily designed for real-time speech communication, they also support text-based chat and instant message functions. Scammers utilize these tactics to mislead victims into clicking on malicious links, attachments, or web forms. Moreover, scammers have also adopted "3D social engineering", using 3D avatars that closely resemble familiar domains to impersonate co-workers, friends, or other contacts and gain access to sensitive information [1, 17].

5 Potential solutions

This section discusses existing work on defenses and countermeasures against the aforementioned selected threats in the Metaverse XR technology.

To mitigate the risks posed by XR technology, various measures can be implemented. Table 1 provides an overview of the selected attacks and their potential countermeasures that can be applied to improve the security of the Metaverse environments. Countermeasures of each attack will be discussed further in the following section.

5.1 Authentication-related threat

Randomize virtual keyboard: A study conducted by Meteriz et al. [14] suggests a straightforward but highly efficient approach to enhance security, which is to randomize the keys at each keyboard usage or after every

Category	Attacks	Countermeasures
Authentication-related	Keystroke logging	•Randomize virtual keyboard •Biometric authentication •Multi-model authentication
Unauthorized access	Impersonation MitR	•Multi-factor authentication •Regularly updating software
Data breaches	Eavesdropping	•Sensory noise to obfuscate •Ductile materials
Social engineering	VR Phishing	•User training for fraud awareness •Multi-factor authentication •Contents moderation

Table 1. An overview of potential XR attacks and their countermeasures

user keystroke. Dynamic keyboard mapping significantly increases the difficulty of tracing user keystrokes and hands motion, thereby reducing the risk of unauthorized access.

Biometric authentication: Biometric data can be used for authentication and verification in the Metaverse. The most typical uses of data are Electroencephalography (EEG), body movements, and Electrooculography (EOG). With the uniqueness of the data, there is no possibility of imitating authentication data. However, it is important to note that the use of biometric data raises privacy concerns, and it is essential to implement appropriate security measures to protect this sensitive information from misuse or unauthorized access [11].

Multi-model authentication: This authentication mechanism provides an additional layer of security in the Metaverse by requiring users to present two or more pieces of evidence during the authentication process. The use of multiple authentication factors makes it more difficult for attackers to gain unauthorized access to systems or sensitive data, as they would need to successfully bypass multiple security measures instead of just one. Kürtünlüoğlu et al. [11] suggest implementing Gaze-based authentication for password entry in virtual reality (VR) headsets. Gaze-based authentication tracks the unique movements of the user's eyes combined with extraocular muscle activations to identify the user. This method offers an increased security by eliminating the possibility

of imitation since it is difficult for an attacker to replicate the unique movements of the eyes of an individual or track the movements from the outside.

5.2 Unauthorized access

In order to infect the environment, attackers need to gain initial access to the environment. Multi-factor authentication [16] can also be implemented as a potential defense to reduce the risk of stolen credentials being used to gain unauthorized access. In addition, regularly updating and patching software and operating systems can prevent attackers from exploiting known vulnerabilities to gain access to the system.

5.3 Data breaches

Shi et al. [18] suggested a potential defense against the privacy risks posed by Face-Mic attacks , which involves adding sensory noise to obfuscate the reconstruction of facial movements and bone-borne vibrations. Another option is to include ductile materials in the foam replacement cover and headband of VR headsets, which can weaken facial vibrations detected by the built-in accelerometer/gyroscope.

5.4 Social engineering

There are various potential solutions that could be implemented to mitigate the risks of social engineering attacks in the Metaverse. One effective solution is to keep up to date on emerging scams and educate users on the risks associated with social engineering and how to identify and avoid these types of attacks. This can be achieved through awareness or training programs that teach users how to identify phishing emails, fake websites, and other deceptive tactics used by attackers. In addition to user education, implementing controls such as two-factor or multi-factor authentication and impartial content moderation or governance functions within community management platforms can help to identify and flag malicious messages or remove abusive users from the environment [1, 2].

6 Conclusion and Future work

Extended Reality technology has revolutionized the way people interact with digital content, and the use of visualization technology in the Metaverses is a prime example of its potential. Despite providing users with an immersive experience, these virtual worlds also present serious security vulnerabilities.

In conclusion, this paper has reviewed and highlighted the risks posed by XR technology, including the possibility of unauthorized access, data breaches, identity theft, social engineering attacks, and other malicious activities. This review has revealed that while there are security concerns associated with XR technology, there are also several measures that can be implemented to mitigate these risks. These measures include strong encryption, multi-factor authentication, and secure data storage systems. Multi-factor authentication, in particular, plays a crucial role in mitigating identity threats related to unauthorized access and phishing attacks. Furthermore, this paper has highlighted the need for collaboration between developers, users, and policymakers to ensure that security and privacy are prioritized in the design and development of the Metaverse.

However, there are limitations to this review. Due to space restrictions, we could not address all potential security risks. Additionally, given the novelty of the Metaverse concept, resources for research in this area remain limited.

In terms of future work, more research is needed in order to develop effective security measures to mitigate the risks posed by the Metaverse. While this paper has focused on the security concerns related to XR technology, there are other emerging technologies that could also have security implications for the Metaverse, such as blockchain and AI. Therefore, future work in this area should focus on identifying and addressing emerging security concerns and exploring new technological solutions to ensure user safety in the Metaverse. This could include developing stronger encryption and multi-factor authentication methods as well as exploring new approaches to secure data storage and communication. Furthermore, the public is in need of a comprehensive education and training programs to raise awareness of the risks associated with social engineering attacks and other malicious activities in the Metaverse. Such programs could help to promote a culture of cybersecurity and contribute to development of a secure Metaverse.

References

[1] Metaverse security: Emerging scams and phishing risks. https://www.pwc.com/us/en/tech-effect/cybersecurity/emerging-scams-and-phishing-risks-in-the-metaverse.html.

[2] Hussain Aldawood and Geoff Skinner. Educating and raising awareness on cyber security social engineering: A literature review. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 12 2018.

[3] Abdullah Al Arafat, Zhishan Guo, and Amro Awad. VR-Spy: A side-channel attack on virtual key-logging in vr headsets. *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, 2021.

[4] Yunqiang Chen, Qing Wang, Hong Chen, Xiaoyu Song, Hui Tang, and Mengxiao Tian. An overview of augmented reality technology. *Journal of Physics: Conference Series*, 1237:022082, 06 2019.

[5] Zefeng Chen, Jiayang Wu, Wensheng Gan, and Zhenlian Qi. Metaverse security and privacy: An overview. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2950–2959, December 2022.

[6] Ke Ching and Manmeet (Mandy) Mahinderjit Singh. Wearable technology devices security and privacy vulnerability analysis. *International Journal of Network Security & Its Applications*, 8:19–30, 05 2016.

[7] Yang-Wai Chow, Willy Susilo, Yannan Li, Nan Li, and Chau Nguyen. Visualization and cybersecurity in the metaverse: A survey. *Journal of Imaging*, 9(1), 2023.

[8] Enrico Constanza, Andreas Kunz, and Morten Fjeld. Mixed reality: A survey. *Lecture notes in computer science*, 5440:47, 01 2009.

[9] Sahraoui Dhelim, Tahar Kechadi, Liming Chen, Nyothiri Aung, Huansheng Ning, and Luigi Atzori. Edge-enabled metaverse: The convergence of metaverse and mobile edge computing, 2022. arXiv: 2205.02764.

[10] Thien Huynh-The, Quoc-Viet Pham, Xuan-Qui Pham, Thanh Thi Nguyen, Zhu Han, and Dong-Seong Kim. Artificial intelligence for the metaverse: A survey. *Engineering Applications of Artificial Intelligence,Volume 117*, 2022.

[11] Pınar Kürtünlüoğlu, Beste Akdik, and Enis Karaarslan. Security of virtual reality authentication methods in metaverse: An overview, 2022. arXiv: 2209.06447.

[12] Jesse Lake. Hey, you stole my avatar!: Virtual reality and its risks to identity protection. *Emory Law Journal*, 69:833, 2020.

[13] Lik-Hang Lee, Tristan Braud, Pengyuan Zhou, Lin Wang, Dianlei Xu, Zijun Lin, Abhishek Kumar, Carlos Bermejo, and Pan Hui. All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda, 2021. arXiv: 2110.05352.

[14] Ülkü Meteriz-Yıldıran, Necip Fazıl Yıldıran, Amro Awad, and David Mohaisen. A keylogging inference attack on air-tapping keyboards in virtual environments. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 765–774, 2022.

[15] Shilpi Mishra, Himanshu Arora, Garvit Parakh, and Jayesh Khandelwal. Contribution of blockchain in development of metaverse. In *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, pages 845–850, 2022.

[16] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2, 01 2018.

[17] Sara Qamar, Zahid Anwar, and Mehreen Afzal. A systematic threat analysis and defense strategies for the metaverse and extended reality systems. *Computers & Security*, 128:103127, 2023.

[18] Cong Shi, Xiangyu Xu, Tianfang Zhang, Payton Walker, Yi Wu, Jian Liu, Nitesh Saxena, Yingying Chen, and Jiadi Yu. Face-mic: Inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, page 478–490, New York, NY, USA, 2021. Association for Computing Machinery.

[19] Maximilian Speicher, Brian D. Hall, and Michael Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–15, New York, NY, USA, 2019. Association for Computing Machinery.

[20] Jean-Philippe Vergne. The future of trust will be dystopian or decentralized: Escaping the metaverse. *SSRN Electronic Journal*, 01 2021.

[21] Martin Vondráček, Ibrahim Baggili, Peter Casey, and Mehdi Mekni. Rise of the metaverse's immersive virtual reality malware and the man-in-the-room attack & defenses. *Computers & Security*, 09 2022.

[22] Yuntao Wang, Zhou Su, Ning Zhang, Rui Xing, Dongxiao Liu, Tom H. Luan, and Xuemin Shen. A survey on metaverse: Fundamentals, security, and privacy. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2022.

[23] J.M. Zheng, K.W. Chan, and I. Gibson. Virtual reality. *IEEE Potentials*, 17(2):20–23, 1998.

Zero trust network security model in cloud networks

Rasmus Blässar
rasmus.blassar@aalto.fi

Tutor: Tuomas Aura

Abstract

This paper explores the zero trust network security model and its applicability in cloud networks. With the increasing adoption of cloud services and microservice architecture, the need for effective security measures has become more critical. The traditional perimeter-based security model is no longer sufficient to protect against modern cyber threats. The zero trust model offers a new approach to securing cloud networks that assumes no user or service is trusted by default. The paper presents three established techniques for a successful implementation of zero trust architecture: identity governance, micro segmentation, and software-defined networks. It then shares experimentation results of micro segmentation implementation in a Kubernetes environment. The results of this study can help organizations secure their cloud networks by better understanding the benefits and challenges of implementing zero trust architecture.

KEYWORDS: Zero Trust, Kubernetes, Access Control

1 Introduction

The widespread adoption of cloud services and shift to distributed application architectures has created new network security challenges. Network

controls must adapt to the dynamic nature of cloud services and support automated and rapid provisioning of computing resources. In addition, the controls must cover an increasing number of network endpoints in multiple cloud locations. A traditional perimeter-based security model, which focus on firewalls and gateways at the edge of the trusted network, is no longer sufficient to defend the increased attack surface of cloud networks. The perimeter-based model lacks flexibility and leaves a large part of cloud network traffic unprotected, particularly the internal service-to-service traffic [5]. As a result, a new security model called zero trust architecture has emerged as an alternative approach to securing dynamic cloud networks.

Zero trust architecture (ZTA) is an information security model that aims to improve network security by bringing the security perimeter closer to critical applications and data [8]. ZTA assumes that every access request is risky regardless of the network location of the endpoints. ZTA uses a variety of techniques to bring the perimeter closer to the applications and data, including identity governance, micro-segmentation and software-defined perimeter. These techniques will help organizations prevent unauthorized access and lateral movement inside internal networks and provide secure remote access for users outside the perimeter.

Despite the benefits of the zero trust model, many organizations still rely on traditional perimeter-based protections [2, 9]. Many organizations are not aware of the limitations of the perimeter-based model and ZTA benefits. Available literature has focused mainly on the zero trust concepts and technical aspects but lacks concrete examples that explain the implementation of ZTA. In addition, it is difficult to find the right solution because there are different techniques and tools that achieve the ZTA objectives. Organizations must adopt different techniques to meet their specific security requirements and find the tools that work with their existing technologies.

This paper aims to clarify how the zero trust architecture can be implemented in cloud networks. The paper reviews established techniques that can be used to overcome the limitations of traditional perimeter-based solutions. The paper goes into more detail on micro segmentation and shares experimentation results of its implementation in a Kubernetes environment.

This paper is organized as follows. Section 2 explains the limitations of the traditional perimeter security architecture. Section 3 discusses

the benefits and different implementation techniques of ZTA. Section 4 shares the experimentation results of micro segmentation implementation in a Kubernetes environment. Section 5 discusses the key findings and advantages and disadvantages of each technique. Finally, Section 6 concludes the paper.

2 Limitations of perimeter-based security architecture

Perimeter-based network architecture is a traditional approach to network security that relies on the creation of a secure boundary around the network to prevent unauthorized access from Internet [6]. The network perimeter is typically guarded by firewalls and other security devices, which are responsible for filtering and blocking incoming traffic based on predefined rules and policies.

In traditional cloud architectures, the network is divided to the trusted cloud network and the untrusted Internet. Between these, there may be a perimeter network, which has historically been called demilitarized zone (DMZ). The perimeter network includes an API gateway or web traffic load balancer that routes client web requests to trusted applications as shown in Fig. 1. The API gateway might be equipped with additional security features such as web application firewall (WAF) and request rate limiting. In addition, the perimeter network includes a VPN gateway that accepts connections that operate over non-HTTP protocols (e.g., SSH and RDP). The usage of VPN has decreased with the proliferation of API interfaces. However, it is still commonly used by administrators to gain remote access to applications servers and databases. However, it is still commonly used by administrators to gain remote access to applications servers and databases.

While the firewalls and gateways have been effective in protecting traditional networks from external threats, they have several limitations in modern cloud networks:

- Limited protection against insider threats [2, 6, 3, 9]: Perimeter-based security architectures are designed to prevent external threats from accessing the network, but they provide limited protection against threat actors that have already gained access to the trusted network (e.g., through a phishing attack or exploited application vulnerability). A threat that is already inside the trust network can often move laterally and remain

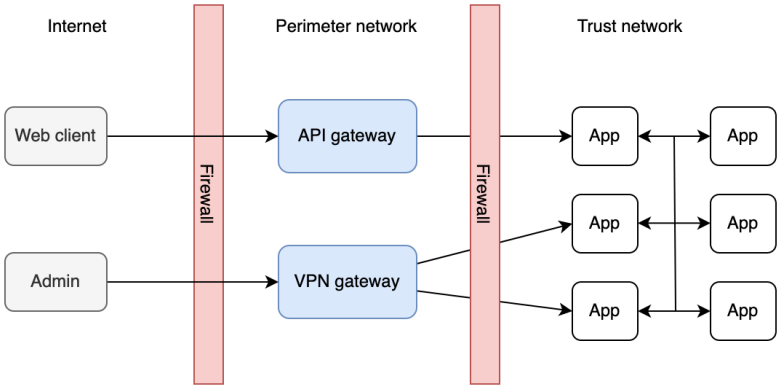


Figure 1. Traditional perimeter based security model in cloud networks

undetected by the perimeter security devices.

- Lack of flexibility in host placement [6]: Placement of cloud hosts (e.g., virtual machines, container instances) is limited by the security policies enforced at the perimeter. For example, virtual machine replication to another cloud region or scaling up container instances may be blocked by the perimeter policy.

- Single point of failure [2, 6]: In a perimeter-based network architecture, firewalls and security gateways become a single point of failure that will compromise the entire network if breached. The overall network security is determined by weakest device or application on the network.

To overcome these limitations, organizations have begun to adopt a cloud-native security approach such as zero trust architecture.

3 Zero trust architecture (ZTA)

The zero trust architecture (ZTA) is a security model that uses a set of modern cybersecurity principles to secure a broad range of infrastructure and workflows [8]. In the context of cloud networks, it provides guidance on how to secure both service-to-service and user-to-service communication. National Institute of Standards and Technology (NIST) has defined seven tenets that a ZTA should use for optimal security:

- All data sources and computing services are considered resources.

- All communication is secured regardless of network location.

- Access to individual resources is granted on a per-session basis.

- Access to resources is determined by a dynamic policy, meaning the access is determined by various factors, including the user's previously observed behavior, device posture and time of the day.

- The security state of all devices and applications is continuously monitored. The access may be denied if the device is observed vulnerable or unmanaged.

- Resource authentication and authorization is constantly evaluated considering the dynamic policy.

- All available logs about the current state of the resources, network infrastructure and communications are collected and used to improve access decisions.

Overall, the zero trust tenets are highly relevant to cloud networks because they provide a framework to secure complex networks with distributed applications and data in multiple locations. By implementing the zero trust architecture, organizations can overcome the limitations set by the traditional perimeter security architecture and take full advantage of cloud services.

However, the implementation of a ZTA is challenging because there is not a single standard approach that can be used. This paper reviews three established techniques, including identity governance, micro segmentation and software-defined perimeter (SDP) that can be used to implement ZTA in different use cases [8, 2].

3.1 Identity governance

Identity governance is an identity-driven approach to implementing ZTA [10]. The approach uses authentication and access controls at the application layer to validate and grant access to users and applications. The access requests are processed by a centralized ZTA policy engine that uses input from multiple external sources, such as IAM and SIEM systems [8]. The policy engine considers the validity of user and workload (e.g., virtual

machine or container) identities, contextual information and data sensitivity when making an access control decision. For example, if a client attempts to access a sensitive resource from an unfamiliar location, the policy engine may require additional authentication steps, for example require admin approval, before granting access.

A ZTA based on solely identity governance approach can be deployed in an open network model, without the use of traditional network firewalls or micro segmentation [8]. This means that network endpoints are directly exposed to the internet and services can communicate without network boundaries. This provides flexibility in cloud service deployments. However, granting open network access exposes the services to malicious attacks on the network layer, including reconnaissance or denial-of-service attacks.

According to NIST, the Identity governance approach is well suited for cloud services delivered by the software as a service (SaaS) model [8]. The network infrastructure in the SaaS model is managed by the service provider, which may not allow the use of security devices maintained by the organization. In the SaaS model, the client identity can be used to form and enforce the zero trust policy. However, ZTA requires that the service supports connecting to the centralized policy engine. In addition, the use of SaaS services requires trust that the service provider has secured its own infrastructure.

3.2 Micro segmentation

Micro segmentation is a technique in ZTA implementation that brings network controls closer to the workload being protected [8, 10]. The approach involves dividing the trusted network into smaller segments containing individual or a small group of workloads. The network segments are typically divided by virtual firewalls as seen in Fig X. Alternatively, micro segmentation can be implemented using host-based firewalls.

The micro segmentation technique requires setting a strict network policy for every user and workload in the network. According to NIST's ZTA tenets, all communication must be monitored and expressly permitted in the policy, including internal service to service communication. This makes managing network rules challenging, especially in distributed cloud networks, because all interactions between users and workloads must be considered [10].

The micro segmentation network policy is defined more efficiently using

workload identities instead of IP addresses. In cloud networks the workloads are becoming more dynamic and IP addresses change frequently because workloads are often moved between hosts and cloud regions. In contrast, the workload identities can be used to define network controls that works in any network location. In addition, the identities can be grouped logically (e.g., multiple containers can share the same identity). The workload identity can be derived from the fingerprint of an application executable (e.g., JAR) or attached workload properties (e.g., Kubernetes labels) [11]. Once the identity has been derived, it can be attached to a network packet header (e.g., TCP option field) and further evaluated by the packet receiver. Different methods for deriving workloads identities are discussed in [11] and implementation using Kubernetes labels is examined in section 4.

The benefits of micro segmentation include increased protection against lateral movement, as attackers are prevented from moving laterally across the network once they have gained access to one segment. Additionally, micro segmentation provides better visibility into network traffic and allows the service provider to detect and respond to threats more quickly. Furthermore, the identity-based micro segmentation approach is network-independent, meaning workloads can be moved flexibly between network locations. However, implementing identity-based segmentation may not be possible in existing networks because it requires special hardware and software components that support deriving workload identity information and tagging network packets. In the end, the network-level access restrictions might not prevent malicious activities at the application layer after the connection has been established. Therefore, it does not replace the need for application-level access control.

3.3 Software-defined perimeter (SDP)

Software-defined perimeter (SDP) is an emerging network architecture developed by Cloud Security Alliance (CSA) to help organizations achieve zero trust objectives [5]. SDP uses an overlay network to dynamically create secure network connections between clients and resources over the internet. In contrast to traditional perimeter networks, SDP is designed to provide secure access to individual resources rather than entire networks. This means that clients are only granted access to the resources they need, rather than having access to all resources on the network. In addition, SDP eliminates the need for complex firewall rules and intru-

sion prevention systems by hiding open ports and services exposed to the internet.

The SDP architecture is defined in detail in a specification document published by CSA. According to the specification, the architecture consists of three main types of components including a controller, initiating hosts and accepting hosts [5]. The controller is a central SDP component responsible of validating and orchestrating communication between initiating and accepting hosts on the network. The initiating host is a software component in the client system, which is used to connect to other SDP protected applications. The accepting host is a dynamic firewall in front of the target application either on the same server or on a gateway. Both the initiating and the accepting hosts must be on-boarded to the network by configuring the controller's connection details, including IP addresses and TLS certificates.

The SDP components interact over separated control and data planes as shown in Fig. 2 [5]. At the beginning of a new network connection, the initiating host sends an authorization request to the controller over the control plane. The controller validates the request and, if it is allowed, instructs the accepting host to allow the connection from the initiating host. Thereafter, the initiating and accepting hosts may proceed to establish an encrypted data channel using mutual authentication protocols, such as mTLS or Internet Key Exchange (IKE). Notably, all connections to the controller and accepting host components must be initiated by a single packet authorization (SPA) packet. The SPA packet is a computationally lightweight UDP-based packet that allows the listening SDP components to quickly reject unauthorized connection attempts. This allows the SDP components to effectively drop all unauthorized requests, which makes the endpoints undiscoverable by attackers on the network.

Overall, the SDP architecture provides an effective technique to protect cloud network endpoints. The SDP controller evaluates access on a per-session basis and can enforce access based on a dynamic access policy. In addition, SPA packets provide protection against denial-of-service attacks (e.g., TCP SYN flood) by effectively dropping all unauthorized packets [5]. However, SDP comes with implementation challenges because it differs significantly from traditional networking practices [10]. The SDP controller must be added to the network infrastructure and both the client and server applications must be refactored to work with SDP components. In addition, the control plane message between the SDP controller

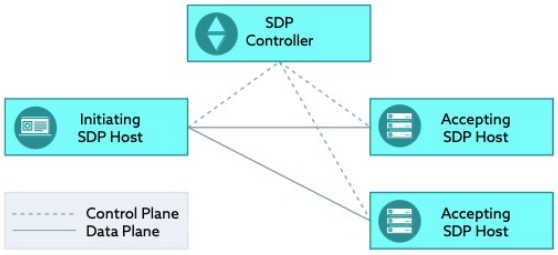


Figure 1: SDP Architecture (previously published by CSA in Software Defined Perimeter and Zero Trust)[5]

Figure 2. Software-defined perimeter defined by CSA [5]

and hosts delays connection establishment [7]. This makes SDP a poorly suited solution for latency-critical services.

4 Implementing micro segmentation in a Kubernetes environment

This section demonstrates the micro segmentation implementation in a Kubernetes environment. The micro segmentation is implemented using Cilium identity-based network policies.

Cilium is an open-source software that provides network connectivity and access control between containerized applications [1]. In a Kubernetes cluster a Cilium software agent is installed on each cluster node. The agent uses eBPF (extended Berkeley Packet Filter) technology to tag workload (e.g., pod) identity information in network packets at the Linux kernel level. The workload identity is derived from Kubernetes labels which are attached to the workload during deployment.

The test environment includes three pods named client1, client2, and server. The client1 and client2 pods have curl command-line tool installed, while the server pod is running a NGINX web server. A Cilium identity-based network policy is created and applied to the cluster. The Cilium policy restricts incoming traffic to the server pod and allows only requests from the client1 pod. All other traffic to the server pod is denied.

The test results show that the Cilium network policy successfully restricted incoming network connections to the server. After the policy was applied network connection attempts from client2 times out when trying to reach the server as shown in Fig. 3. Connections from client1 are still allowed because the pod label is was whitelisted in the policy.

Former research [4] has demonstrated how ZTA can be implemented in a Kubernetes environments using identity governance technique. The

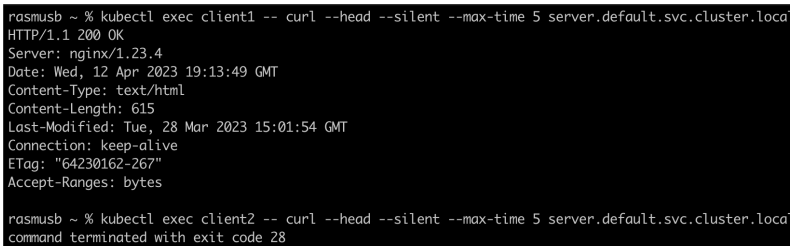


Figure 3. HTTP HEAD request from client pods to server pod after Cilium newtork policy has been applied

Cilium network policy complements the former implementation by adding network layer controls which would prevent lateral movement inside the network if a user or application identity is compromised.

The experiment demonstrated that Cilium network policies provide an easily achievable security improvement in Kubernetes environments. With Cilium, network policies can be defined using Kubernetes labels, making them easy to create and maintain. This allows for a more intuitive and flexible approach to network security, where policies can be defined based on application-level requirements instead of IP addresses and ports.

5 Discussion

The ZTA advises organizations to transition from a traditional network-centric perimeter security model to a more identity-centric model. In ZTA, identity is the primary factor in determining access to resources. Every client is authenticated and authorized before it is granted access to any resource, regardless of whether they are inside or outside the network perimeter.

Despite this, network firewalls and security devices still have a critical role in securing cloud networks. However, their role is different from the traditional perimeter security model, where they are responsible of protecting the network perimeter. In ZTA, network firewalls and security devices are still used to protect network segments and limit access to resources based on identity and contextual factors. However, they are no longer the only line of defense, and their role is to complement other security measures, such as identity and access management (IAM) systems and endpoint security solutions.

This paper reviewed three established techniques associated with the ZTA implementation, including identity governance, micro segmentation,

and software-defined perimeter. The covered techniques are not mutually exclusive, and they can be used together to achieve optimal security. Identity governance ensures that only authorized clients are granted access to resources. Micro segmentation reduces the attack surface and prevents lateral movement within the network. Software-defined perimeter provides a secure and flexible way for users to access resources from any location. Together, these techniques create a security model that is flexible, scalable, and effective in protecting an organization's resources against the most common threats.

However, implementing the ZTA in cloud networks may be challenging. Access policies must be constantly updated to reflect changes in the network and to ensure that clients are granted the appropriate levels of access, which requires a greater level of management oversight than in traditional security models. To address this challenge, organizations should leverage automation tools to help manage their access policies and consider adopting a phased approach to the implementation.

6 Conclusion

The zero trust network security model provides an effective approach to securing cloud networks. An effective zero trust architecture implementation requires combing techniques presented in this paper, including identity governance, micro-segmentation, and software-defined networks. While challenges exist, careful planning and investing in the right technologies, such as Kubernetes Cilium integration, can help organizations achieve the benefits of this approach. Overall, the zero trust model represents a significant shift in network security that is necessary to address modern cyber threats in cloud environments.

References

[1] Cilium documentation, 2023.

[2] Christoph Buck, Christian Olenberger, André Schweizer, Fabiane Völter, and Torsten Eymann. Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust. *Computers & Security*, 110:102436, 2021.

[3] Ramaswamy Chandramouli. Guide to a secure enterprise network landscape. Technical report, National Institute of Standards and Technology, 2022.

[4] Daniel D'Silva and Dayanand D. Ambawade. Building a zero trust architecture using Kubernetes. In *2021 6th International Conference for Convergence in Technology (I2CT)*, pages 1–8, 2021.

[5] Jason Garbis and Juanita Koilpillai. Software-defined perimeter (SDP) specification v2.0. Technical report, Cloud Security Alliance, 2022.

[6] Evan Gilman and Doug Barth. *Zero trust networks*. O'Reilly Media, Incorporated, 2017.

[7] Abdallah Moubayed, Ahmed Refaey, and Abdallah Shami. Software-defined perimeter (SDP): State of the art secure solution for modern networks. *IEEE Network*, 33(5):226–233, 2019.

[8] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture. Technical report, National Institute of Standards and Technology, 2020.

[9] Sirshak Sarkar, Gaurav Choudhary, Shishir Kumar Shandilya, Azath Hussain, and Hwankuk Kim. Security of zero trust networks in cloud computing: A comparative review. *Sustainability*, 14(18), 2022.

[10] Naeem Firdous Syed, Syed W. Shah, Arash Shaghaghi, Adnan Anwar, Zubair Baig, and Robin Doss. Zero trust architecture (ZTA): A comprehensive survey. *IEEE Access*, 10:57143–57179, 2022.

[11] Zirak Zaheer, Hyunseok Chang, Sarit Mukherjee, and Jacobus Van der Merwe. EZTrust: Network-independent zero-trust perimeterization for microservices. In *Proceedings of the 2019 ACM Symposium on SDN Research*, SOSR '19, page 49–61, New York, NY, USA, 2019. Association for Computing Machinery.

Uncertainty estimation in model-based reinforcement learning with ensembles

Roope Kajoluoto
roope.kajoluoto@aalto.fi

Tutor: Shibei Zhu

Abstract

Model-based reinforcement learning (MBRL) algorithms have emerged as a promising approach to reinforcement learning that also provide excellent sample efficiency. MBRL algorithms approximate the environment dynamics based on prior experience to perform planning without the need for environment interaction. However, state transitions in complex environments generally include stochasticity and the model is not always accurate. Therefore, many practical variants of model-based algorithms rely on either quantifying the amount of uncertainty in the dynamics model or otherwise mitigating the effects of potentially unreliable plans. One technique to achieve uncertainty-aware algorithms is presented by ensemble learning. Ensembles of dynamics models can stabilize planning and have proven to be effective in deriving explicit uncertainty estimates. MBRL algorithms that utilize ensemble learning have reached state-of-the-art asymptotic performance and can outperform model-free approaches in many environments. This paper reviews different uncertainty estimation and mitigation approaches within model-based reinforcement learning that are based on ensembles.

KEYWORDS: Reinforcement learning, Model-based reinforcement learning, Uncertainty estimation, Ensemble methods

1 Introduction

Reinforcement learning (RL) algorithms generally fall into one of two distinct types: model-free and model-based algorithms [26]. Model-free algorithms have shown promising results in many optimal control tasks, such as playing videogames [17] and more recently with natural language processing [25, 20]. Model-free algorithms can perform well in a wide range of environments, but typically require large amounts of data to converge [12]. Model-based reinforcement learning (MBRL) approaches differ in that they attempt to model the environment in addition to learning a policy. Using this estimated dynamics model allows for predicting the outcomes of actions without having to explicitly simulate them. This is also referred to as planning [26]. As a result, model-based algorithms are generally much more sample-efficient when compared to their model-free counterparts [18].

However, an imperfect model of the environment dynamics can direct the training to fall into a local minima or to fail completely due to misled exploration. Additionally, some approaches use the model's plans directly as candidate action sequences when acting in the environment [4, 19]. Therefore, a central problem in MBRL is building algorithms that take this uncertainty into account in their exploration and decision-making. One way to estimate uncertainty is presented by ensemble methods. Algorithms that account for the uncertainty in decision making are also referred to as uncertainty-aware algorithms [4]. This paper reviews different ensemble-based approaches to uncertainty estimation within MBRL.

This paper is organized as follows. Section 2 provides necessary notation and background information. Section 3 provides a short motivation for the use of environment dynamics models in RL. Section 4 reviews different ways uncertainty can be estimated with ensembles. Section 5 presents different ways estimations of uncertainty can be used in RL algorithms. Section 6 gives brief conclusions.

2 Background

Reinforcement learning is applied in Markov decision processes (MDP), which are defined as tuples $(S, A, \rho, \gamma, D, R)$. S is a finite set of states (state space), and A is a set of actions (action space). ρ is a state transition function, for which it applies that $s_{t+1} = \rho(s_t, a_t), \forall s_t, a_t$. For probabilistic dynamics the output of ρ is defined as a conditional probability distribution over possible

successor states, $\rho(s_t, a_t) = Pr(\cdot|s_t, a_t)$. $\gamma \in [0, 1]$ is a discount factor, which is used to emphasize immediate rewards over future ones. D is the distribution of possible initial states from which the initial state s_0 is drawn and $R : S \times A \to \mathbb{R}$ is the reward function. The goal of RL is to find an optimal policy π^* that achieves a maximal reward R_{π^*} over trajectories:

$$R_{\pi^*} = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{N} \gamma^t R(s_t, a_t) \right] \tag{1}$$

Here $\tau = \{(s_n, a_n)\}_{n=1}^{N}$ stands for trajectories sampled using the policy π.

Additionally, MBRL methods aim to construct an approximation ρ_θ of ρ given a set of trajectories $\overline{\tau}$ from the environment. With this approximation, it is possible to predict trajectories resulting from applying sequences of actions, usually sampled from a policy π [4]. It is worth noting that depending on the complexity of the transition model, the true environment dynamics model can be either directly recovered (i.e., for environments such as Go or chess) or approximated with certain error (i.e., complex multi-degree of freedom robots).

Ensemble learning encompasses algorithms that combine the predictive power of multiple models. An ensemble of models are trained for the same task, and their predictions are used jointly to achieve a greater performance than with a single model [1]. In the context of MBRL, multiple environment dynamics models $\overline{\rho}_\theta = [\rho_{\theta 1}, \rho_{\theta 2} ... \rho_{\theta b}], b \in \mathbb{N}_+$ are trained for the task of predicting state transitions. These ensembles are often deep ensembles (DE) as deep neural networks are used. As pointed out by Abdar et al. [1], DEs are applied to obtain better predictions on test data and to provide model uncertainty estimates when the learners are provided with OoD data. Ensemble methods can also be applied to other parts of the MDP. For example, SUNRISE [12] presents a framework for building model-free algorithms that employ an ensemble of agents. However, this paper focuses on model-based approaches that approximate uncertainty present in the environment dynamics model.

A key aspect of all MBRL is the class of models used to predict the environment dynamics. This choice is crucial to how well the MBRL algorithm performs. Historical work focused on simpler function approximators, namely linear functions [2] and Gaussian processes [11, 6]. While these functions can work well in low-data situations, they do not scale well to high-dimensional or discontinuous state spaces [4, 18]. Recent work has transitioned to more complex functions, mainly neural networks [21]. The use of ensemble methods is not reliant on the type of model used, but the model's stochasticity is often a crucial part of the overall estimate of uncertainty.

2.1 General notion of uncertainty

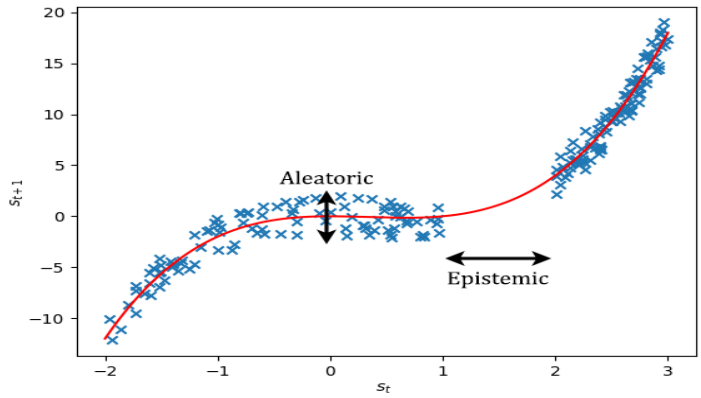


Figure 1. A simple visualization of the two types of uncertainty regarding the successor state s_{t+1} when executing action $a \in A$ in different states $s_t \in S$. The blue crosses are samples and the red curve is the true, noiseless function.

Uncertainty within decision making can be broadly divided into two types: aleatoric and epistemic [15]. They are also known as observation and process uncertainty [4], or data and knowledge uncertainty [16]. In essence, uncertainty that is deemed irreducible is aleatoric while reducible uncertainty is epistemic [7].

Aleatoric uncertainty arises from stochasticity in the data generation process, as the data can be noisy or imprecise. In addition, it is also impractical to fully represent every detail about the environment in non-trivial problem settings. Therefore, the policy is given input x instead of the "true" input x^* [15]. This noise also applies to the dynamics model, as it is predicting transitions from one uncertain state to another. Aleatoric uncertainty cannot be removed or even reduced with any predictive model or amount of data, as it is always present in the state transitions.

On the other hand, epistemic uncertainty stands for uncertainty that is present in a specific model due to a lack of knowledge about the data generation process. This can be intuitively thought of as the "confidence" in the predictions of the model aligning with the real environment dynamics. An optimal model of the environment dynamics has no epistemic uncertainty, which is noted as the red curve in Figure 1. This uncertainty can be characterized as either parametric or structural [14]. The parametric epistemic uncertainty defines the uncertainty regarding the parameters of the model. This can be reduced by collecting more data. Structural epistemic uncertainty reflects whether the chosen model family is expressive enough to represent the true dynamics model ρ. For example, a linear model will always have structural

epistemic uncertainty when fit to the data shown in Figure 1, as the true transition function is polynomial. This can be reduced by changing the model family.

It is desirable to not only be able to quantify the amount of uncertainty, but also to be able to differentiate between the two types of uncertainties. This is especially true in the context of guiding exploration. In the interest of removing uncertainty from the current dynamics models, the agent should be guided towards state transitions that have high epistemic uncertainty. If the two types of uncertainty are indistinguishable, the policy might favor transitions with high aleatoric uncertainty instead, which will not improve the model [4].

3 Motivation for MBRL

This section provides a short overview of the advantages and disadvantages of model-based RL approaches. Firstly, the section discusses some of the benefits of having access to an accurate dynamics model. In other words, why approximating the environment dynamics is beneficial as part of the RL paradigm. Secondly, the section discusses additional considerations and challenges that need to be addressed when using the model.

3.1 Benefits

- **Transferability:** A sufficiently accurate model of the environment dynamics can assist in training a policy. This ability is not specific to the reward function used. In essence, once the environment model is learned it can be used to train new policies that may maximize different reward functions. This is referred to as transfer learning [31]. An example of this type of transfer learning is presented by Sekar et al. [24], where the algorithm first explores the environment without a reward function and then uses the model built from this exploration to train a policy. Similarly, PILCO [6] first trains a dynamics model and then trains a policy that maximizes the given reward function using only state transitions sampled from the model. Therefore, a dynamics model be used for training any number of possibly dissimilar policies.

- **Sample complexity:** Sample complexity can be characterized as the amount of samples required for an algorithm to reach the desired degree of accu-

racy in learning the target function [26]. In other words, the more useful information that can be extracted from given data, the lower the sample complexity and the higher the data efficiency. In contrast to model-free methods, MBRL algorithms also use the collected trajectories to infer information about the environment itself. Once an environment dynamics model has been approximated, it can be used to create new data by sampling. This data can be used identically to data sampled directly from the environment to, for instance, learn a policy [9]. Therefore, MBRL algorithms require inherently less environment interaction than model-free RL algorithms, and can be characterized as being more data-efficient [18].

- **Safety:** The ease and feasibility of collecting trajectories varies greatly between environments. RL algorithms generally require collecting trajectories with a suboptimal policy. This is known as online learning [13]. This will usually not pose safety concerns in a simulated environment, but could lead to catastrophic failure in the real world. Due to its data efficiency, MBRL algorithms are often able to converge with much less environment interaction. In fact, given a highly accurate dynamics model they may not require any trajectories to be sampled with the policy during training [6]. This approach, which is also referred to as offline learning [13], does not require any environment interaction during policy optimization. This can be very beneficial in safety-critical environments. In addition, some MBRL algorithms provide safety guarantees on policy stability, given a safe initial policy [3].

- **Generalizability:** In contrast to online RL algorithms that sample from the environment during training, offline RL algorithms learn a policy exclusively from previously collected data [13]. This restricts the algorithm's ability to test whether the policy has improved, but may be unavoidable if sampling from the environment is not possible. In the offline learning setting, existing model-based reinforcement learning methods significantly outperform their vanilla model-free counterparts [30]. This implies that MBRL methods are able to generalize better to states outside the data, leading to improved generalization. Additionally, this result also implies that MBRL methods may not suffer from overfitting and overestimation to the extent of model-free approaches.

- **Explainability:** The model can be used to understand the decisions made by a policy. MBRL algorithms rely on planning with the dynamics model,

which means that this dynamics model can also be evaluated independently. In this way, it is not only possible to see what the policy is trying to achieve, but also how it intends to achieve it [18].

3.2 Challenges

- **Performance:** The asymptotic performances of MBRL methods have historically lagged behind their model-free counterparts [4]. This means that the benefits of converging more quickly have been overshadowed by converging at a less optimal solution. This gap has narrowed in recent years with new algorithms, such as probabilistic ensembles with trajectory sampling (PETS) [4] and model-based policy optimization (MBPO) [9]. Both algorithms matched the current benchmarks for model-free RL performance achieved by PPO [23] and SAC [8] in the tested environments. Though more modern model-free algorithms such as SUNRISE [12] have proven to provide improved performance, model-free and model-based algorithms both continue to have environments in which they are superior. Therefore, there is often no clear answer as to which approach will provide the best performance.

- **Long trajectories:** Long trajectories sampled from the dynamics model tend to face compounding errors, which make them increasingly error-prone. An efficient approach to countering this has been to employ ensemble methods, which can mitigate the effects of single, inaccurate dynamics models [4, 5, 22]. Alternative approaches include training the model specifically to predict further into the future instead of compounding single-state transitions [28], avoiding uncertain state transitions [30] and training the model on its own outputs [27].

4 Uncertainty estimation with ensembles in MDPs

This section discusses different ways ensembles can be used to estimate uncertainty in MDPs. The section covers a few notable uncertainty estimates, and the motivations for them. The ways in which these estimates can be used in the MBRL algorithms are covered in the next section.

4.1 Training the models

Firstly, for the ensemble to be useful, the individual models need to be sufficiently different. An ensemble of identical models provides no benefits to predictive performance. Uniqueness of the models could be achieved by training the models with differing seeds, model architectures or with different subsets of the training data [16]. Many different methods to define the data for the models have been presented. For example, the data can be chosen with or without replacement or such that any single trajectory can or cannot be used in the training of multiple models. A generic way to train these models is to utilize bootstrapping [4], where the data for each model is drawn randomly and independently with replacement. These approaches can clearly be combined in a multitude of ways. For instance, MOReL [10] initializes the models with different weights but also trains them with different mini-batches of trajectories.

4.2 Different estimates of uncertainty

Perhaps the simplest uncertainty estimate that utilizes the trained ensemble is the ensemble discrepancy [10]:

$$\mathrm{disc}(s,a) = \max_{i,j} \|\rho_{\theta_i}(s,a) - \rho_{\theta_j}(s,a)\|_2 \tag{2}$$

Here, the maximal disagreement between different models is interpreted as the epistemic uncertainty. With a stochastic dynamics model, this can be extended to state distribution expectations: $\mathrm{disc}(s,a) = \max_{i,j} \|\mathbb{E}[\rho_{\theta_i}(s,a)] - \mathbb{E}[\rho_{\theta_j}(s,a)]\|_2$. Many different approaches can be built simply from the differences of the state distributions. For example, the variance of the different models' predictions: $\mathrm{Var}(\bar{\rho_\theta}(s,a))$, where $\bar{\rho_\theta}(s,a) = [\rho_{\theta_1}(s,a), \rho_{\theta_2}(s,a) ... \rho_{\theta_b}(s,a)]$. Intuitively, the larger the variance of the predictions is, the more disagreement there exists between the models. It is clear that if all the models in the ensemble were identical to a single model (including the true dynamics model), all these metrics would be equal to zero. Therefore, all these metrics can be interpreted as the epistemic uncertainty that could be reduced by collecting more data, especially in low-data regions.

On the other hand, the aleatoric uncertainty is usually approximated from the stochasticity of the individual models' predictions. In PETS [4], an environment dynamics model is a probabilistic neural network that defines parameters for a multivariate Gaussian distribution. The variance of the resulting state distribution can be interpreted as the aleatoric uncertainty, averaged

over the ensemble [4]. In this way, the ensemble is not used directly for uncertainty estimation but as a method for averaging over models which may be inaccurate on their own.

Some estimates attempt to capture both the aleatoric and epistemic uncertainty. MOPO [30] uses an ensemble of models $\overline{\rho}_\theta = [\rho_{\theta 1}, \rho_{\theta 2} ... \rho_{\theta b}]$, each of which are expressed as a multivariate Gaussian distribution with a diagonal covariance matrix: $\rho_{\theta i}(s_{t+1}|s_t, a_t) = N(\mu_i(s_t, a_t), \Sigma_i(s_t, a_t)), 0 \leq i \leq b$. The level of uncertainty is defined as the maximal product of the standard deviations of the individual state dimensions. This is equivalent to the maximal Frobenius norm of the (diagonal) covariance matrices Σ_i in the ensemble:

$$\mathrm{U}(s, a) = \max_i \|\Sigma_i(s, a)\|_F \tag{3}$$

The maximal product of standard deviations is used instead of the mean to improve the robustness of the estimates.

4.3 Disadvantages

Although ensembles have been proven to be a viable way to estimate uncertainty, they are not applicable in every situation. Training an ensemble of models scales the computational complexity and time consumption approximately linearly with respect to the number of models [1]. Additionally, storing multiple models increases the memory requirements of the algorithms. These problems are not exclusive to reinforcement learning but encompass all uses of ensemble methods. Some approaches have been proposed to mitigate these problems, such as ensemble distillation [16]. Ensemble distillation techniques attempt to distill the ensemble into a single model, with the promise of retaining the uncertainty estimate but mitigating the computational cost.

In addition to scaling the required training resources, ensemble-based uncertainty estimates may perform poorly in some environments. Yu et al. [29] show empirical proof that uncertainty estimates will be inaccurate in certain offline learning scenarios where generalization to unknown behaviors is difficult. This conclusion extends to uncertainty estimates that are derived from ensembles. These samples of unknown behaviors are often referred to as Out Of Distribution (OOD) samples, as they were not part of the data used to train the models. The authors proposed Conservative Offline Model-Based policy Optimization (COMBO), an algorithm that does not incorporate uncertainty estimations. The authors also present experimental results, where the algorithm outperforms previous uncertainty-aware MBRL methods such as MOPO [30] and MOReL [10] in three environments where generalization is

critical.

5 Using the uncertainty in MBRL algorithms

This section presents different approaches to using the uncertainty in MBRL algorithms. These approaches can be generally thought of as belonging into one of two classes. Firstly, some algorithms use an explicit uncertainty estimate that is calculated and used directly in the algorithm. On the other hand, some approaches don't have an explicit uncertainty estimate but combat uncertainty by incorporating the ensemble of dynamics models into the algorithm. This is generally done to mitigate the effects of single inaccurate models.

5.1 Stabilizing sampling

Using complex function approximators such as probabilistic neural networks increases the risk of overfitting. Overfitting occurs especially at the start of training when data is scarce [4]. When sampling trajectories from the ensemble, algorithms such as PETS [4] and MBPO [9] sample state transitions from the models uniformly. Consequently, if one model is unreliable or has overfit due to a lack of data, its predictions are not always used. In MBPO, this prevents the policy from exploiting an unreliable model especially in low-data regions. This is because the policy is trained using samples from the dynamics model ensemble. In PETS, this approach prevents single unreliable models from repeatedly misdirecting the agent.

5.2 Partitioning the state-action space

In the MOReL algorithm [10], the ensemble discrepancy from Equation 2 is used as part of an uncertainty function. The function determines whether the level of disagreement between the models in the ensemble is acceptable regarding the successor state of a specific state-action pair:

$$U^{\alpha}(s,a) = \begin{cases} \text{Known}, & \text{if } \operatorname{disc}(s,a) \leq \alpha \\ \text{Unknown}, & \text{otherwise} \end{cases} \tag{4}$$

Where α is a tunable hyperparameter that defines the level of acceptable discrepancy. This has the effect of dividing the state-action space into "known" and "unknown" transitions, the latter of which are undesirable. With this definition, the authors define an (α, κ)-pessimistic MDP as a tuple: $(S \cup$

HALT, $A, \rho_p, \gamma, D, R_p$). Here, HALT defines a special absorbing state. The state transition function ρ_p and reward function R_p are defined as follows:

$$\rho_p(s_{t+1}|s,a) = \begin{cases} \text{HALT}, & \text{if } U^\alpha(s,a) = \text{Unknown or } s = \text{HALT} \\ \rho_{\theta i}(s_{t+1}|s,a), & \text{otherwise} \end{cases}$$
$$R_p(s,a) = \begin{cases} -\kappa, & \text{if } s = \text{HALT} \\ R(s,a), & \text{otherwise} \end{cases} \tag{5}$$

Here, $\rho_{\theta i}$ is sampled uniformly from the ensemble. Otherwise, the definition (α, κ)-pessimistic MDP follows a normal MDP. In the (α, κ)-pessimistic MDP, state transitions in which the model's accuracy cannot be guaranteed are directed to the HALT state. The HALT state induces a negative reward and cannot be escaped. Intuitively, this forces the policy to follow trajectories that have less uncertainty to them. The policy is expected to do this even in situations where, in the absence of uncertainty, some other trajectory would appear superior.

5.3 Penalizing unreliable transitions

As uncertainty makes the trajectories sampled less reliable, state transitions with high uncertainty can be seen as less desirable. Therefore, one approach is to use the uncertainty directly as a penalty term in the reward function. In MOPO [30], the uncertainty-penalized reward function is defined as $R_\lambda(s,a) = R(s,a) - \lambda U(s,a)$, where $U(s,a) = \max_i \|\Sigma_i\|_F$ as discussed in Section 4. Here, λ is a tunable hyperparameter. Intuitively, the more preferred transitions with less uncertainty are, the higher λ should be. Additionally, λ should be higher if the uncertainty function underestimates the true uncertainty and lower if the opposite is true. This approach has the benefit that it only restricts the implementation of the reward function. This means that any RL algorithm, either model-based or model-free, can be used to maximize it and remain uncertainty-aware.

6 Conclusion

This paper reviewed notable ensemble-based approaches to quantify environment dynamics model uncertainty in model-based reinforcement learning. MBRL is an approach to reinforcement learning that also approximates the environment dynamics model. This dynamics model is then used to simulate

trajectories, otherwise known as planning. MBRL can have many benefits over model-free methods, including sample efficiency and tranferability. If the environment's state transitions include stochasticity, a deterministic or fully certain model of the environment dynamics will be inaccurate. Additionally, a complex environment dynamics model will not be accurate until a sufficient amount of data has been collected, often leading to unavoidable uncertainty during training. Therefore, incorporating uncertainty into the model of the environment dynamics is crucial to the performance of many MBRL algorithms. This can be achieved by using ensembles of models to approximate state transition uncertainty and to stabilize trajectory sampling. Ensemble-based uncertainty estimates can face problems with computational complexity, memory requirements and poor generalisability to certain types of OOD data. However, uncertainty-aware MBRL algorithms that employ ensemble methods can often perform exceptionally well when compared to model-free algorithms. The use of ensembles in building uncertainty-aware MBRL algorithms continues to be an active field with many open questions. More research is required into building different uncertainty estimates from ensembles and how the estimated uncertainty can be better incorporated into the learning process.

Bibliography

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[2] J Andrew Bagnell and Jeff G Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 1615–1620. IEEE, 2001.

[3] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

[4] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[5] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

[6] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[7] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[9] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[10] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

[11] Malte Kuss and Carl Rasmussen. Gaussian processes in reinforcement learning. *Advances in neural information processing systems*, 16, 2003.

[12] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

[13] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[14] Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. Accurate uncertainty estimation and decomposition in ensemble learning. *Advances in neural information processing systems*, 32, 2019.

[15] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[16] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076*, 2019.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[18] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[19] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[20] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback, 2022. *URL https://arxiv. org/abs/2203.02155*.

[21] Ali Punjani and Pieter Abbeel. Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. IEEE, 2015.

[22] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[24] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

[25] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[26] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[27] Erik Talvitie. Model regularization for stable sample rollouts. In *UAI*, pages 780–789, 2014.

[28] William Whitney and Rob Fergus. Understanding the asymptotic performance of model-based rl methods. 2018.

[29] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[30] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[31] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

Algorithmic voting power

Roope Karppinen
roope.karppinen@aalto.fi

Tutor: Antti Rinnasto

Abstract

KEYWORDS: information society, critical sociology, voting advice application, algorithm, power

1 Introduction

Within an average household, the diversity and the quantity of algorithmic application, that reside inside a household, have drastically increased in the last decade. With every passing day, society relies increasingly on applications that use different types of algorithms to operate.

Most of these algorithms gather some form of data from the surrounding area, such as a sports watch that counts the wearer's heartbeat, sleep duration and daily exercise durations. While these new strategies of collecting data have helped with everyday life, they have also established a more robust information based society. Different companies, such as Google and Facebook, have thrived in this new datadriven world with their different types of algorithms.

Zuboff [15], defines this new age of data capitalization as "surveillance capitalism". She more broadly defined it as "This new form of information capitalism aims to predict and modify human behavior as a means

to produce revenue and market control". This type of algorithm usage can be very detrimental in the long run, when looking at things such as web search algorithms, like Google's PageRank [12], or Facebook's Edgerank [4], because they can take away human agency from decision making. That is one possible *power* that algorithms can have, including voting advice applications (VAAs).

VAAs are a new way to improve electoral participation in different countries, which utilize the strengths of algorithms. This is mostly done through simplified issue-based voting, which is a way of figuring out which politician aligns most closest with a voter based on different political issues. However, because it is a simplified way, there can be some problems that arise from using this type of metric to choose a candidate. Simplified issue voting often relies on having stances on a linear scale, for example, does a voter agree or disagree with supporting abortion? Abortion policies are an incredibly complex and multileveled topic, and trying to simplify that issue is practically impossible because there are often too many stances to consider. Also, considering that voting is one of the most effective ways to create social change, the importance of working VAAs becomes that much more significant.

This paper reviews the critical sociology of voting advice applications, i.e. to identify potential problems and dangers that can arise with VAAs, as well as propose possible solutions to some of these problems.

The paper is structured as follows. Section 2 covers the basic concepts and definitions, such as information society and power through the lens of sociological theory. Section 3 inspects voting advice applications, their history and basic functionalities. Section 4 goes over different studies from a critical sociology viewpoint on VAAs and looks at recent changes done to YLE's VAA. Lastly, Section 5 covers the closing statements.

2 Core concepts

This section summarizes the important basic ideas behind algorithmic power and builds a foundation before Section 3.

2.1 Algorithms

When someone uses the word algorithm, it can convey different ideas from person to person. The question in itself can be quite deep depending from

what type of viewpoint it is answered from, such as Hill [9] did from a ground level of computer science. For the sake of simplicity, this paper views algorithms as a set of instructions that produce an optimal output based on the given inputs. A good example of this type of algorithm is the PageRank algorithm that is used in Google search engines.

As explained by Brin and Lawrence [3], the entire search engine of Google started of from the PageRank algorithm that is still used today. It uses a formula to determine the rank of each page within a web search result link. The formula is constructed as follows:

A represents the page, which has $T_1...T_n$ number of citations to it, and C gives the total number of outgoing links from the citation. d is the dampening factor, which is chosen between $0 \leq d \leq 1$.

$$PR(A) = (1 - d) + d \times (PR(T_1)/C(T_1) + ... + PR(T_n)/C(T_n)) \quad (1)$$

While the specific details within the PageRank equation are more complex, e.g., the number chosen for d, the basics of PageRank follow the definition given in Equation 1. The formula takes d and $T_1...T_n$ as its inputs, and outputs the optimal pagerank $PR(A)$ of page A for the Google search engine.

2.2 Power (Sociology theory)

Power can be seen everywhere in the world. It is a tool which can effect lives in either positively or negatively depending on who uses it and how it is used.

In sociological theory, power can be used to explain why different events happen as they do. For example, what events lead to inequality or discrimination and how they can occur. Algorithms are one way of inspecting these types of power relation dimensions, due to the fact that they can be seen as tools which might affect the decision making of different people. In the past 10 years, there have been enormous amounts of studies on the social power of algorithms [2, 11], as well as on specific algorithms, such as Facebook's EdgeRank algorithm [4]. These are important aspects to study, in order to counter act the possible malicious effects that come with algorithmic power.

One example of malicious use of algorithms is the Cambridge Analytica scandal. In it 50 million Facebook profiles informations were harvested

to build an algorithm which was designed to target individual US voters with personalised political advertisements, in order to change election outcomes [5]. Similarly, VAAs could quite easily be used in a malicious way, namely favouring certain candidates over others.

2.3 Information society

The increasing use of algorithms in current daily life has caused society to take a new form, one based on information. This new society has been named by many different researchers as an information society.

Information society can be defined through six different definitions: *technological, economic, occupational, spatial, cultural and theory* [13]. The most relevant definition in VAAs relates to the technological one, because the growth in algorithmic usage can be correlated with the rise of VAAs popularity, which will be shown in section 3.1.

According to the technological definition, the increase in creation of different new technologies, such as different information and communications technologies (ICTs), has been an indicator for a new information based age [13]. By increasing the usage of these ICTs in different sectors, society begins to revolve increasingly around information as a resource. This definition fits perfectly with applications, including PageRank, because it requires information about links into and from a page to function, which can be seen as a way to create profit through new methods of using information. This growth can also be seen through VAAs history.

3 Voting advice application

This section will cover information about VAAs: their history, how they function and different types of VAAs.

3.1 History

The origins of VAAs can be traced back to 1989, when a package called *StemWijzer* was developed in the Netherlands. It was originally aimed towards the education sector, where, at the time, it found most of its success. A few years later, the original designers of *StemWijzer* developed an internet-based model, which was utilized in the 1998 Dutch parliamentary elections [6]. Around the same time in Finland, the Finnish Public Broadcasting Company (YLE), had developed a VAA for public usage.

This was followed by *Helsingin Sanomat*, the largest daily newspaper in the country, with their own VAA for the 1999 European Parliament elections, and by 2001, the number of different VAAs grew to as much as 11, for the Finnish voter to freely use [6].

With time, and the widespread reach of the internet, VAAs started to gain traction all over Europe and the world. During the 2002-2003 Dutch elections, the *StemWijzer* gained over two million different users [6]. Due to the success of VAAs in Finland and Netherland, other European countries started to develop their own VAAs. This rise in VAA's popularity can be correlated with the growth of information society, based on the technological definition as given in the previous section. This is because the growth of ICTs created new opportunities for different sectors to utilize them in new ways.

3.2 Functionality

The general functionality of every VAA model works the same. It takes the voter's preferences on different issues, i.e. abortion or taxes, as inputs, and compares them with the answers of parties/politicians to determine the closest matching candidate as output. Every issue within the VAA has a choice ranging from agree to disagree, and the questionnaire comprises around 25-30 different questions about general and current political issues. Using the basic input/output idea given in Section 2.1, the input of VAAs are the voters choices on different issues, and the output is the total correlation of choices between a voter and a candidate.

While the main functions might be the same for all VAAs, they still differ in some aspects from each other. Garcia and Diego [6] showed differing aspects of different VAAs in their book. Some of these differing aspects include the weighting of different propositions, calculation methods and the answer pattern. As an example, VAAs might give an option between 'agree', 'disagree' and 'undecided', while others could have different number of ranging opinions between 'strongly agree' to 'strongly disagree'. There might also be differences, based on what type of elections the VAAs are used for. Most often, VAAs are used for party-based elections, but these are not the only types of elections in the world. For example, in Finnish elections, the VAAs compare the voter's opinion to every single politician, rather than a party. There also can be the question of fact checking each politician's opinion on every topic. Do the answers of each candidate/party correspond to their real opinions, or do they differ

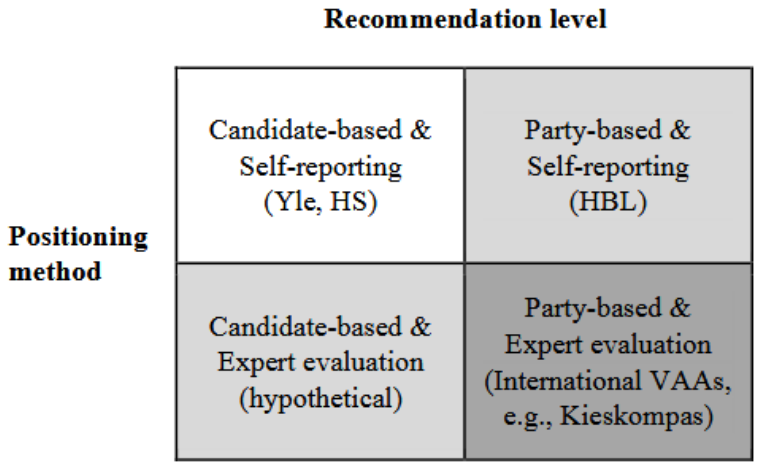


Figure 1. Four different types of VAAs [10]

drastically? While this is not a problem exclusive to VAAs, nevertheless, it still affects possibility of taking advantage of VAAs algorithmic power.

Figure 1 shows one way of differentiating VAAs by their functions. The VAAs are divided into four different types, with two different axes. One axis compares if the VAA is used in a party or candidate context, and the other separates them on how the opinions are fact checked. One type of VAAs within Figure 1 are purely hypothetical, because getting expert evaluations on thousands of different candidates would be incredibly tasking [10].

4 Analysis

The purpose of VAAs is to make voting for every individual easier and more straightforward. A study by Gemenis and Rosema [8] into the 2006 Dutch parliamentary elections indicated this to be the case. In it, the researchers used two waves of face-to-face interviews on a large random sample of the Dutch electorate. The first wave was done before the election, using questions that measured the voter turnout and VAA usage. The second wave was done after the election, which asked the participants if they knew any VAAs, had they used any and what were the results from said VAA? The results of the study suggest that the most increase in electoral participation was found in young voters, lower education levels and limited political knowledge, which are the optimal target groups for electoral participation increases. Overall, the VAAs resulted in a 4.4% voting increase within the Dutch election. These results seem to

correlate with VAA's impact in other European countries (Germany, Finland, Switzerland, Netherlands), which ranged from a 0.7% increase in Germany's 2009 election to a 6.3% increase in the Netherlands 2012 election's [7]. However, while the effectiveness of VAAs seems to be positive in electoral participation, there are still some design features that are still in need of improvements.

Isotalo noted in his thesis [10] that one of the biggest faults in the Finnish VAAs designs was a lack of transparency. This was due to the algorithms not being open for the public, no information for voters on how recommendations are constructed and the lack of sharing the datasets of candidate responses. The fact that open transparency is still missing from some VAAs is a large detriment to their trustworthiness to the general public. Having an open-source VAA algorithm would facilitate detecting algorithmic faults and assist in grasping how they work for those that are interested in them. There are also no real disadvantages to this either, due to the fact that VAAs are, by design, free to use and available for everyone. It should be a core design feature in every VAA, in order to avoid the negative affects of algorithmic power.

Another disadvantage tied to VAAs comes with using simplified issue-based voting, concerning the ambiguity of each choice. For example, when choosing from a scale between agree to disagree, the middle option is often used when the voter has no opinion or little knowledge on the topic. However, a study by Baka et al. [1] found out that approximately 75% of the studies participants did not choose the middle option due to lack of knowledge or formulated opinion. Those 75% chose the middle option either because of certain issues dilemmatic nature (45%) (eg, 'economy' vs 'culture') or how the questions were formulated (30%) in either ambiguous or biased ways. This type of classification of differing choices is demerit for simple issue voting, because it tries to simplify complex topics to simple groups of answers. And by combining differing opinions to the same set, it raises questions about how accurate VAAs truly are in their results.

Before the 2023 parliamentary elections in Finland, YLE changed their VAA in a few ways. According to YLE [14], the questions in the VAA were made to be as simple and straightforward as possible, in order to remove ambiguity and bias. The revised VAA also comprises of two different questionnaires instead of only one, as was done before. The first questionnaire scales the participants' answers with different candidates, like most Finnish VAAs do, and the second one scales the answers for different po-

litical parties. The party based version is more straightforward to use than the candidate one, by changing the response to 3 possible answers ('yes', 'no' or 'skip') from 5 possible answers in the candidate questionnaire ('agree', 'somewhat agree', 'somewhat disagree', 'disagree' or 'skip'). This seems to be an attempt to distinguish which politicians' parties, among the top resulting candidates, the voter is closest to. While the changes have tackled some of the issues covered in this section concerning VAAs, ie. ambiguous questions, there are still needs for improvement in areas, namely the transparency of the VAA.

5 Conclusion

This paper has conducted a review into the power of VAAs, from a sociological theories perspective. It gives a basic idea of what VAAs are and how they function, as well as showing the benefits and possible negative affects that VAAs have concerning their algorithmic power. Algorithmic power refers to the power which algorithms hold in affecting society from a multitude of angles. VAA's can quite easily bring positive change by, i.e. increasing political participation, but it also can affect society in negative ways.

VAAs are designed for people who do not have much political awareness. These types of people are also the perfect demographic for algorithmic power to get a hold of. Most of the polically unaware people do not question the results of an algorithm, such as VAA, and are satisfied with the minimal participation. Therefore, it is important to ensure that there are no malicious affectors within the VAA's algorithm.

One way of defending against the algorithmic power of VAAs relies on the creators and researchers of VAAs. Always inspecting and analyzing if an algorithm holds some sort of bias or unfair advantage within it, and improving on those mistakes. By making every VAA open source would be one of many possible ways in improving VAAs from the creator side, because it would increase the detection of problematic elements within VAAs.

Since VAAs use issue-based voting as a primary function, this causes some problems with the accuracy of the results. Because simplified issue voting puts people who might have different stances in the same group, VAAs can almost never find the most optimal candidate for a voter. Therefore, a VAA should not be the only way of determining a candidate to vote

for. Researching the top candidates from a VAA is an optimal way of avoiding the possibility that an algorithm makes the vote for the voter.

References

[1] Aphrodite Baka, Lia Figgou, and Vasiliki Triga. 'neither agree, nor disagree': a critical analysis of the middle answer category in voting advice applications. *International Journal of Electronic Governance*, 5(3-4):244–263, 2012.

[2] David Beer. The social power of algorithms, 2017.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[4] Taina Bucher. Want to be on the top? algorithmic power and the threat of invisibility on facebook. *New media & society*, 14(7):1164–1180, 2012.

[5] Carole Cadwalladr and Emma Graham-Harrison. Revealed: 50 million facebook profiles harvested for cambridge analytica in major data breach. *The guardian*, 17(1):22, 2018.

[6] Diego Garzia and Stefan Marschall. *Voting advice applications*. Oxford University Press, 2019.

[7] Diego Garzia, Alexander H Trechsel, and Andrea De Angelis. Voting advice applications and electoral participation: A multi-method study. *Political Communication*, 34(3):424–443, 2017.

[8] Kostas Gemenis and Martin Rosema. Voting advice applications and electoral turnout. *Electoral studies*, 36:281–289, 2014.

[9] Robin K Hill. What an algorithm is. *Philosophy & Technology*, 29:35–59, 2016.

[10] Veikko Isotalo. Designing voting advice applications: The finnish case, 2020. Master's Thesis.

[11] Maximilian Kasy and Rediet Abebe. Fairness, equality, and power in algorithmic decision-making. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 576–586, 2021.

[12] Matteo Pasquinelli. Google's pagerank algorithm: A diagram of cognitive capitalism and the rentier of the common intellect. *Deep search: The politics of search beyond Google*, pages 152–162, 2009.

[13] Frank Webster. *Theories of the information society*. Routledge, 2014.

[14] Yle. Yle's election compass finds the parties and candidates that share your views. https://yle.fi/a/74-20021433, 2023. Accessed: 2023-03-29.

[15] Shoshana Zuboff. Big other: surveillance capitalism and the prospects of an information civilization. *Journal of information technology*, 30(1):75–89, 2015.

Supply chain security in the npm ecosystem

Roope Räsänen

roope.rasanen@aalto.fi

Tutor: Bufalino Jacopo

ABSTRACT

Node package manager (npm) is heavily used in modern JavaScript application development, which also makes it attractive to attackers looking to inject malicious code into software. As packages can depend on other packages, dependency trees in projects can be very large making identifying vulnerabilities more difficult. Most common attack methods are typosquatting and combosquatting, where a malicious actor publishes a package with a name that is similar to a popular package hoping that victims accidentally install the wrong one. Dependency confusion may also be used to target companies with internal package managers by publishing a malicious package with the same name as an internal package to a public repository, such as the npm registry. Malicious packages can have different behaviors, including stealing of information, leaving a backdoor, and sabotage. These goals are often attempted via a post- or preinstall scripts in the package.json file. Almost all vulnerabilities can be avoided by being careful about which packages developer installs and keeping existing packages up to date. Using a third-party tool such as Snyk can make detecting vulnerabilities easier and accidentally installing malicious packages harder.

Keywords: npm, Node package manager, supply chain security, typosquatting, combosquatting, dependency confusion, Snyk

INTRODUCTION

Software supply chain security is an increasingly critical issue as software development and distribution continue to become more complex and global [15]. Worries of malicious actors tampering with supply chains are justified because if any link of the chain is compromised, the consequences can be varied and vast, starting with theft of data and including business disruption [15]. Such worries have escalated into nation-state-level intervention in the operations of the global supply chain, such as several Western nations banning the import and use of 5G networking equipment developed by Huawei Technologies Co., Ltd [15].

Package managers have become a vital part of the modern software development process [3]. These allow developers to reuse third-party code, share their own code, minimize their codebase, and simplify the build process [3]. However, recent reports showed that package managers have been abused by attackers to distribute malware, posing significant security risks to developers and end-users [3].

The npm ecosystem, as the largest software registry in the world, is particularly vulnerable to security risks due to its open and decentralized nature [20]. The npm ecosystem is an important platform for developers and organizations, as it provides access to a large repository of software components that can be easily integrated into their applications [16]. However, this ease of access also creates significant security risks, as malicious actors can exploit the npm ecosystem to distribute malware or to compromise the integrity of software components [18]. For example, eslint-scope, a package with millions of weekly downloads in npm, was compromised to steal credentials from developers [3].

This research paper aims to examine the current state of software supply chain security in the npm ecosystem, identifying the potential threats and vulnerabilities, as well as exploring best practices and solutions for securing the software supply chain in this context. I will start by providing some background on npm and supply chain vulnerabilities. This paper will then examine what supply chain vulnerabilities have been used in the npm ecosystem. I will also examine proposed solutions to reduce the amount of these vulnerabilities. Finally, this paper presents a discussion on what areas of npm supply chain security should be studied further.

BACKGROUND

Node package manager

Package managers, such as npm, serve collections of typically open source software packages, distributed through some package registry [19]. In the case of npm, this

registry is called npm registry. Each of these packages has one or more releases. Releases of a package are called package updates. Package releases are denoted by a unique version number. Higher version number means that a package is newer. Version numbers typically follow semantic versioning, which states that packages should follow a multi-component version number scheme major.minor.patch[-tag]. Backwards incompatible package updates increment the major component, important backwards compatible changes increment the minor component, and backward compatible bug fixes increment the patch component.

Node package manager (npm) serves as a frontend to a large repository of JavaScript software packages, which are used both in browser-side and in server-side JavaScript applications [16]. With over a million packages, it is the biggest package manager in the world [9]. The rapid growth of npm can be attributed to its open and free nature, where arbitrary users can share and reuse code by using a single command [20].

When a developer installs a package from npm, this package including its version is defined in a package.json file [16]. Listing 1 shows a basic version of a package.json file, which was obtained by generating a new Create React App [5] project. The installed packages are listed in the dependencies-block of the json file. There is also a devDependencies-block, which represents packages that are only used for local development, but are not installed when the application is deployed.

```
{
  "name": "example",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  }
  "devDependencies": {
    "eslint": "^8.34.0"
  }
}
```

Listing 1. Example package.json file

In addition to a package.json file, npm uses a package-lock.json file, which is used to lock versions of packages to a specific version [20]. This ensures that all developers

and users use the same versions of the package [20]. However, this approach can cause issues. If a vulnerability is fixed for a given dependency package, projects using said package will not update the dependency until a new lockfile is generated [20]. Users may then be using an application with vulnerabilities.

Packages can depend on other packages, which can result in a huge tree of packages being installed when user installs a single package [16]. For example, installing the express package [4] also installs 31 other packages, which the express package depends upon. These packages may then depend on other packages and so on. This reliance on dependencies may cause problems.

Many npm packages are very small [8]. These so-called micro-packages often have a single line of code. For example, a package called is-negative-zero exists and its only purpose is to determine whether a given input is negative zero. This package has 637 dependents and over 26 million weekly downloads on npm [10].

Software supply chain

Supply chain traditionally refers to a process of transforming resources to finished products, which are then used by the end-users [11]. Software supply chains share similarities with these traditional supply chains. For example, software can be used as part of another larger software and it will end up in the use of some user eventually.

Software supply chains can also suffer from same problems as traditional supply chains [11]. There can be issues with late delivery, counterfeit problems, human errors, and other risks. In addition, attackers may try to inject malicious code to some part of the supply chain. Figure 1 illustrates the various points at which malicious code can enter it.

NPM SUPPLY CHAIN VULNERABILITIES

Typosquatting and combosquatting

Typosquatting and combosquatting refer to techniques where a malicious actor publishes a package with a name that is similar to a popular package and waits for users to accidentally misspell the package name when installing and therefore download the malicious package [3] [7].

Typosquatting attacks hope that a victim makes a typo, meaning they misplace a single letter or mistype the word [7]. One example of this is the popular lodash package incident, where a package called loadsh was published on the registry hoping that users download it instead. Because load is an actual English word and loda is not, it is easy for users to make this spelling mistake and therefore fall victim to this attack.

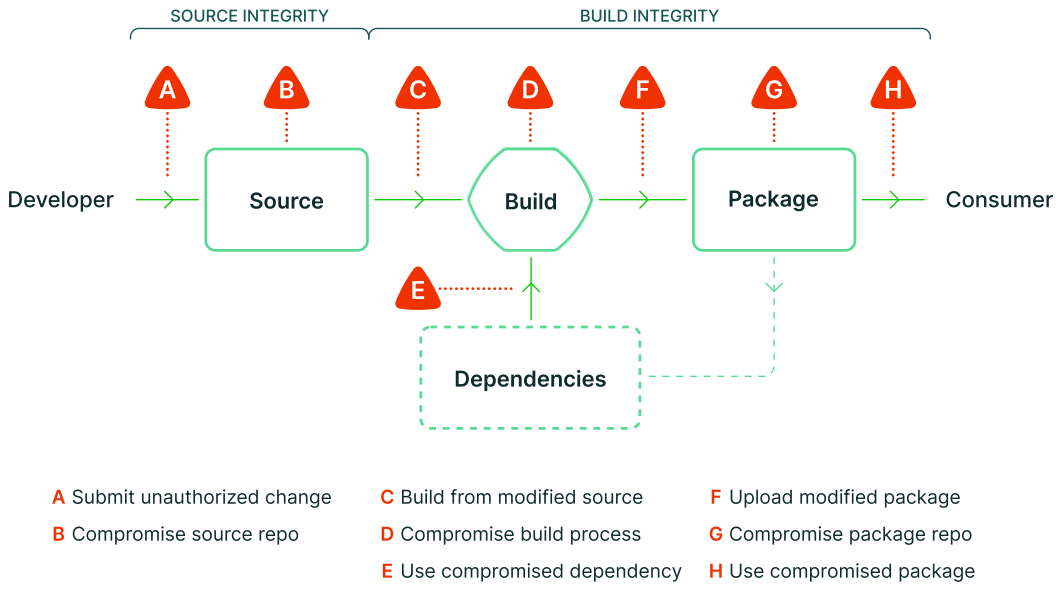


Figure 1. Software supply chain threats [12]

Combosquatting attacks, on the other hand, target users downloading packages with multiple words in their name [7]. An example of this could be changing the order of the words of a popular package, that is creating a package with a name of name-package when a popular package called package-name exists. Remembering the order of the words can be difficult and many can fall victim to this attack as well.

These attacks are dangerous because the malicious package affects not only the user that installs it but all packages that depend on it [7]. For example, some developer could create a package and publish it to npm all while being unaware that they were targeted by a typosquatting attack. Then, this package that the developer created will share the malicious code further. Other packages could then depend on this package and so on.

Big dependency trees

Reliance on dependencies may cause problems if users who host these packages decide to change or remove them [2]. For example, in 2016, a package called left-pad was removed from the npm registry [2]. This caused massive issues as largely used packages such as React used this package as a dependency, and could thus no longer work because this dependency was missing [2]. The package contains only 11 lines of code and this left many developers questioning whether these type of packages should exist at all [2]. The issue was finally resolved by npm itself, restoring the code to the registry [2].

Dependency confusion

Dependency confusion is a software supply chain attack that substitutes malicious code for a internal dependency [1]. This can be done, for example, through researching a name for a private package that is used internally by some company and then publishing a package with the same name to a public repository hoping that developers mistakenly pull the malicious package from the public repository instead of the private one [1].

Snyk security research team wrote a blog post [14] about a targeted npm dependency confusion attack, which serves as a good example of how malicious packages work. A package called gxm-reference-web-auth-server was found by the research team they found suspicious. The package contained a postinstall-script, which invoked a file that contained obfuscated code. The package.json also contained two gibberish dependencies, ldtzstxwzpntxqn and lznfjbhurpjsqmr. Listing 2 shows the package.json of this malicious package and listing 3 shows the contents of the obfuscated file that the script evoked.

```
{
  "name": "gxm-reference-web-auth-server",
  "version": "1.33.8",
  "description": "",
  "main": "index.js",
  "scripts": {
    "postinstall": "node confsettingsaaa.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "dependencies": {
    "axios": "0.26.0",
    "targz": "1.0.1",
    "ldtzstxwzpntxqn": "^4.0.0",
    "lznfjbhurpjsqmr": "^0.5.57",
    "semver": "7.3.5"
  },
  "author": "",
  "license": "ISC"
}
```

Listing 2. Package.json file of gxm-reference-web-auth-server package

```
const a0_0x489fde=a0_0x5400;
(function(_0x552d48,_0x2cc03c){
const _0x462334=a0_0x5400,_0x2fedb3=_0x552d48();while(!![]){try{const
    _0x5c5667=-parseInt(_0x462334(0x167))/0x1*(-parseInt(_0x462334(0
   x10b))/0x2)+-parseInt(_0x462334(0x116)).......# trimmed
```

Listing 3. Contents of confsettingsaa.js

This malicious package is targeting a single, unknown, company that the attacker knows has a package with this name in their private repository [14]. Snyk researchers were able to deobfuscate the code and find out that the package was attempting to spawn a shell on the machine of the victim.

Malicious behaviors

Packages can have many different malicious behaviors [3]. These include stealing of information, leaving a backdoor, and sabotage. Malicious packages can be divided into two categories, (1) packages that perform malicious actions, such as ex-filtrate sensitive information, and (2) packages that manipulate behavior of other packages [17]. The latter is also called dependency-based attack [17].

Malicious packages often utilize post/preinstall scripts in the package.json file [17]. These scripts can do virtually anything, because they are able to spawn a shell and run any code they desire. A dependency based attack could download a malicious file from existing packages in the user's computer or from a non-local repository in the internet. The package itself can also contain malicious code that the script runs, which would fall into first category.

AVOIDING VULNERABILITIES

Snyk [13] lists 10 npm security best practices to avoid being targeted by a supply chain attack.

Ignoring runscripts

Many malicious packages utilize post/preinstall scripts to run their malicious code [1]. These types of attacks can be completely avoided by ignoring run-scripts [13]. This can be done by always installing packages with the flag –ignore-scripts or by adding ignore-scripts to a project's .npmrc file.

Updating outdated dependencies

Always updating packages to their latest releases is not necessarily a good practise if it is done without reviewing release notes, code changes and testing [13]. In my experience, it is also not a realistic goal because updates can fundamentally change how packages work and can introduce breaking changes to production if not done carefully. However, it is important to keep packages relatively up to date to avoid vulnerabilities [13]. Snyk [13] recommends that one should often check how outdated packages are using the npm outdated command. In addition, a command called npm doctor exists, which acts as a health assesment tool to diagnose your environment [13]. This tool checks that

npm registry is reachable, git is available, reviews installed npm and node.js versions, runs permission checks on local and global node_modules folders and checks the npm module cache for checksum correctness [13].

Auditing for vulnerabilities

Snyk hosts tools such as Snyk advisor and Snyk vulnerability database, which should be used to crosscheck your packages for vulnerabilities [13]. However, it is recommended that projects on GitHub add snyk to the project, which will then automatically scan any added packages for vulnerabilities in pull requests [13]. Snyk also offers a command-line tool if adding it to GitHub is not feasible [13].

DISCUSSION

Node package manager is a tool that has become standard when working with JavaScript. Usage of libraries makes writing code much faster and easier. However, I find it questionable whether really small packages containing less than 10 lines of code are actually needed in the ecosystem. These micropackages add up quickly and bloat trees of dependencies.

One aspect on software supply chain security that could warrant more research is the developer aspect when reacting to warnings or errors when building dependencies. For example, in 2021, there was a vulnerability warning reported about nth-check [6], which comes with react-scripts, which is a React library that is often used to create new React projects. The warning turned out to nothing, as it only introduced a vulnerability when the package accessed the network, which it could not do.

I saw this warning myself in 2021 and thought nothing of it because I am used to seeing warnings and errors in console. I would be interested to know whether this is the case with other developers and whether this affects supply chain security. If warnings are ignored, then perhaps the warnings should be shown in some other place than the console. Additionally, if there are warnings shown that turn out to be nothing, as was the case with nth-check [6], does this make developers pay less attention to warnings?

CONCLUSION

This paper has examined the current state of software supply chain security in the npm ecosystem, identified potential threats and vulnerabilities, as well as explored best practices and solutions for securing the software supply chain in this context.

There are many malicious packages in the npm registry. Most common attack vectors are typosquatting and combosquatting, because these are impossible to patch in the npm

side. That is, npm will always allow creating packages with names similar to existing ones. Having a big dependency tree makes being the victim of an attack more likely because more packages results in a bigger attack surface. Any one of the packages that a project uses could be hijacked by social engineering or credential theft and then updated with malicious code.

Most common attack vectors, typosquatting and combosquatting, can be avoided by being careful about packages that are installed. Using a third-party tool such as Snyk adds a safety net if developers accidentally install malicious packages.

REFERENCES

[1] ActiveState (2022). Dependency Confusion.

[2] Collins, K. (2016). How one programmer broke the internet by deleting a tiny piece of code.

[3] Duan, R., Alrawi, O., Kasturi, R. P., Elder, R., Saltaformaggio, B., and Lee, W. (2020). Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages. arXiv:2002.01139 [cs].

[4] Express (2022). express npm package.

[5] Facebook (2023). Create React App.

[6] GitHub (2021). nth-check vulnerability found in react-scripts@4.0.3 · Issue #11647 · facebook/create-react-app.

[7] Kaplan, B. and Qian, J. (2021). A Survey on Common Threats in npm and PyPi Registries. In Wang, G., Ciptadi, A., and Ahmadzadeh, A., editors, *Deployable Machine Learning for Security Defense*, Communications in Computer and Information Science, pages 132–156, Cham. Springer International Publishing.

[8] Kula, R. G., Ouni, A., German, D. M., and Inoue, K. (2017). On the Impact of Micro-Packages: An Empirical Study of the npm JavaScript Ecosystem. arXiv:1709.04638 [cs].

[9] Nassri, A. (2020). npm Blog Archive: So long, and thanks for all the packages!

[10] npm (2021). is-negative-zero.

[11] Sabbagh, B. A. and Kowalski, S. (2015). A Socio-technical Framework for Threat Modeling a Software Supply Chain. *IEEE Security & Privacy*, 13(4):30–39. Conference Name: IEEE Security & Privacy.

[12] SLSA (2023). Introduction to slsa.

[13] Snyk (2019). 10 npm Security Best Practices.

[14] snyk (2022). Targeted npm dependency confusion attack caught red-handed.

[15] Viega, J. and Michael, J. B. (2021). Struggling With Supply-Chain Security.

Computer, 54(7):98–104. Conference Name: Computer.

[16] Wittern, E., Suter, P., and Rajagopalan, S. (2016). A Look at the Dynamics of the JavaScript Package Ecosystem. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 351–361.

[17] Yip, D. Y. K. (2022). *Empirical study on exploitation of dependency-based attacks in Node.js*. PhD thesis, Iowa State University.

[18] Zahan, N., Zimmermann, T., Godefroid, P., Murphy, B., Maddila, C., and Williams, L. (2022). What are weak links in the npm supply chain? In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '22, pages 331–340, New York, NY, USA. Association for Computing Machinery.

[19] Zerouali, A., Mens, T., Decan, A., and De Roover, C. (2022). On the impact of security vulnerabilities in the npm and RubyGems dependency networks. *Empirical Software Engineering*, 27(5):107.

[20] Zimmermann, M., Staicu, C.-A., and Pradel, M. (2019). Small World with High Risks: A Study of Security Threats in the npm Ecosystem.

Exploring the Threats of White-box Targeted Adversarial Examples for Automatic Speech Recognition

Rui Liao
rui.liao@aalto.fi

Tutor: Blerta Lindqvist

Abstract

Adversarial machine learning (AML) is an emerging field of interest. Previous research showed that attackers could construct adversarial examples that lead to misclassification in deep learning models. Inspired by studies in the computer vision domain, more recent attacks are seen on automatic speech recognition (ASR) systems. It raises serious security concerns with the increasing adoption of voice interfaces in various devices. This paper aims to raise awareness of the topic, introduce the characteristics of white-box targeted adversarial examples against ASR systems for a wider audience and explore the threats they pose. It suggests that a framework may be needed to quantify their full impact and the current technologies may not yet pose a practical threat.

KEYWORDS: adversarial machine learning, evasion attack, ASR

1 Introduction

Artificial neural networks (ANN) are often utilised in pattern recognition and classification problems, where they process information in a fast yet massively parallel manner [2]. Automatic speech recognition (ASR) is an important application of ANNs. The models' robustness is very

critical when the human voice is used for biometric authentication or when speech is used as a primary interface. Adversarial machine learning (AML) addresses this general concern and connects machine learning (ML) with cybersecurity principles [1].

In 2013, Szegedy et al. [22] discovered that deep neural networks (DNN) are susceptible to adversarial examples for image classification, by demonstrating how small crafted perturbations in inputs can successfully mislead the networks' classifiers to incorrect outputs. Attacks with the same purpose, such as the Fast Gradient Sign Method (FGSM) [10], projected gradient descent (PGD) [3] and many others have been published. However, their effectiveness has been evaluated primarily on the networks inside the visual domain. Generating adversarial examples for audio is considered more challenging [8], but it has been shown that white-box targeted attacks regardless of the input audio sample and desired output can be achieved, where the input can even contain no speech at all [6]. Some more recent studies have successfully made the targeted perturbations inaudible to human ears [7].

This paper aims to raise awareness of the topic and discusses the viability of white-box targeted adversarial attacks on ASR systems. It includes an overview of the necessary background, a review of common challenges, some specific approaches taken and a discussion on both the attack and defence mechanisms. Section 2 maps out some important terms in the literature. Section 3 extends the discussion on threats posed by white-box targeted attacks and briefly touches on their defences. Section 4 delivers some closing remarks.

2 Background

2.1 The Victim System

ASR systems can convert audio signals of human speech to text. Some well-known ASR services include Cortana from Microsoft, Google Assistant and Siri by Apple. As they are rapidly being integrated into devices used daily, they have access to a lot of personal information. Protecting them against maliciously crafted voice commands is important for their security. Apart from these commercial products, there are open source options that can be thoroughly studied such as Kaldi and Project Deep-

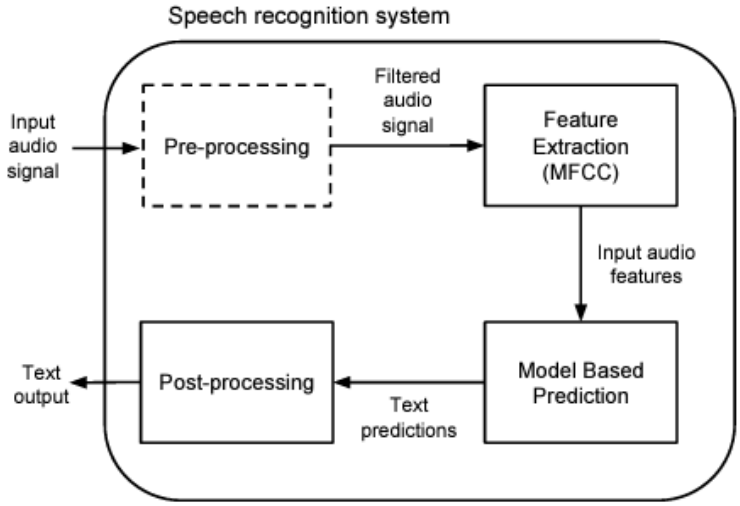


Figure 1. Typical ASR workflow [23] has 4 stages: Pre-processing, feature extraction, model based prediction and post-processing.

Speech by Mozilla.

To make a conversion, an ASR puts an audio sample through several processing stages, the typical steps [23] are summed up in Fig. 1.

The system first filters the input frequencies based on their energy levels, then extracts acoustic features from the filtered signals. This preprocessing stage removes the frequencies irrelevant to speech recognition, only keeping signals above an energy threshold. The most used parametric representation for extracted features is the Mel-frequency cepstral coefficients (MFCC) [11] based on human hearing perceptions. It segments the audio into short overlapping frames of uniform length and calculates the feature vector for each frame, where the coefficients of the vector capture acoustic information within the frame. In short, the audio signal is translated into a feature matrix with MFCC coefficients. The number of frames is the theoretical maximum density [6] (or rate per second) of characters an input audio can be transcribed to. This allows an adversarial target string to be longer than the original phrase, though synthesising new characters is harder than silencing existing ones selectively.

Next, the results are fed into a trained model for phonemes recognition before they are mapped to a string. Finally, some post-processing is needed to improve the usability of the transcription; handling of the grammar rules, formatting and spell checking, for instance.

For the prediction stage, different types of models are available. The focus here is on models that are neural networks. The networks for speech recognition differ from a standard feedforward classification network [6], because there are too many possible labels for the output phrase. It is

infeasible for any system to exhaustively enumerate them. Thus, possibility distributions over individual characters are produced using recurrent neural networks (RNN) [12]. An RNN is difficult to train, but it has many recurrent connections and can represent information within a context window, hence allowing sequential and time-dependent data to be processed.

When an ASR is at work [25], it looks for two types of inputs: An activation followed by a command. Many state-of-the-art systems accept specific wake words, like "Hey Siri" or "Hey Cortana". Then, the microphone records ambient sound until it captures a voice command to be processed. The activation could be speaker-dependent and has to be trained by a user, but the command processing phase will more likely use a speaker-independent algorithm. Different attack scenarios will arise from whether the attacker can obtain any voice recordings from the user that the activation may depend on. It is possible [16] to create complete machine synthesised speech or use voice morphing - altering an existing voice - to easily clone a victim's voice with mere minutes' worth of their recorded speech. However, it is different [25] from directly generating wake words with features extracted from the legitimate speaker's voice recordings. Without any user voice samples, brute-force attempts can be made using voices generated with text-to-speech(TTS) systems.

2.2 The Threat Model

For our threat model, white-box refers to having complete access to the victim ASR systems including all parameters.

An evasion attack [3] in the context of AML refers to the scenario whereby an attacker, armed with white-box knowledge of the victim model, is allowed to modify either the input data directly, the feature vectors or specific features during test time. The extent of the attacker's knowledge or capability may vary, but the goal is to create a sample that the victim will misclassify. Such a perturbed sample is called an "adversarial example" [22]. It is considered a targeted adversarial example if it has a target class to be misclassified as.

A targeted evasion attack for an ASR system [6] is illustrated in Fig. 2. Perturbations are added to a sample waveform as noise, so the system produces a different target string or nothing at all. For the latter, it is possible to hide speech by adding adversarial noises to transcribe the audio to a sequence of space characters; this is easier than targeting spe-

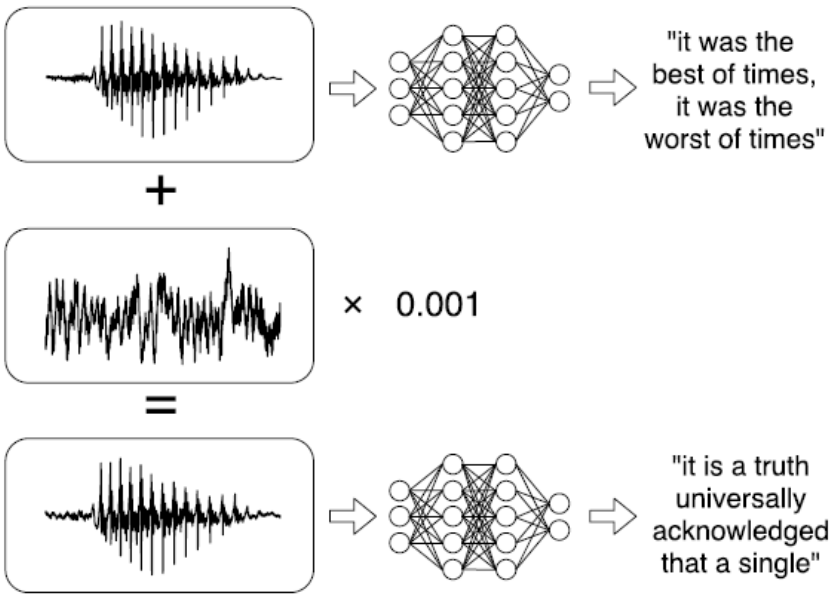


Figure 2. Targeted evasion attack on audio [6]: Adding a small perturbation to the original input to change its transcription.

cific phrases. The attacker intends for the victim system to follow the malicious commands in the adversarial examples. The commands could ask a mobile device to turn on airplane mode to create a denial-of-service scenario, interfere with voice-controlled car navigation systems or simply make the victim device visit attacker-controlled pages for a drive-by download of malicious content.

Some perturbations developed for one model are also effective against others. This perplexing phenomenon is known as transferability. It is believed to be a property of neural networks [6] [18]. Not all perturbations are transferable, but transferability led to the discovery that universal perturbations exist for audio [17]. These perturbations can be combined with arbitrary input audio samples to cause misclassification and transfer across models.

There are two common ways to carry out a targeted evasion attack: Feeding the adversarial sample directly into the victim model(end-to-end) or playing the recorded sample near the victim system's microphone(over-the-air). In the second case, it is a natural goal to keep the perturbations inaudible to avoid alerting the owner of the victim system. One available technique is psychoacoustic hiding [19][20]. It takes advantage of human hearing thresholds and use frequency masking to conceal the perturbations. While over-the-air attacks are more realistic and practical, the sample must be robust against unpredictable noises. Empirical evidence suggests [7] that signal distortions come from the hardware(the playback

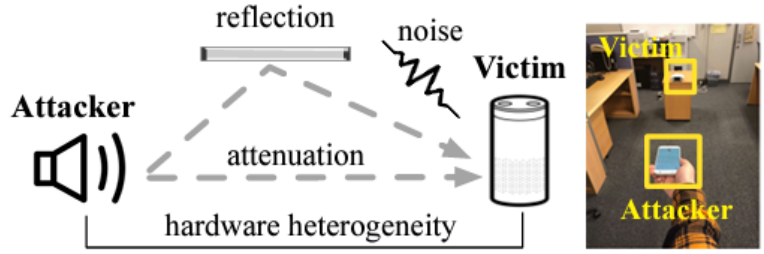


Figure 3. Over-the-air execution of audio adversarial examples [7] face distortions from 3 sources: the hardware, the acoustic channel and the environment.

speaker and receiving microphone), the acoustic channels(signal strength reduction, multi-path propagation etc.) and the playback environment, as shown in Fig. 3. These effects must be accounted for in the generation process. If a sample robust to the above is centrally broadcasted, it could simultaneously attack all active ASR systems within the signal coverage. For samples not generally robust or adaptive enough to be transmitted over-the-air, they require precise environmental conditions for a successful execution. It is trivial to pre-code measured environmental effects at specific locations, but that makes an assumption about attackers' prior access and cannot generalise well.

2.3 The Attack Methods

Constructing an adversarial example is commonly formulated as an optimisation problem [6][3], typically based on the loss function of the victim model. A loss function serves as a quantitative measure during the model-based prediction stage to show how good the predictions are; the greater the value, the worse the classifier. Well-crafted perturbations are optimised to be minimal but still effective in maximising the loss, meaning it will cause the victim model to produce the target label. The choice of a suitable loss function greatly impacts the model's sensitivity to the type of errors that will produce meaningful feedback during its training for improvement. It also influences the level of distortions needed for its adversarial examples [6].

Below are some of the best-known methods used in creating adversarial examples presented progressively.

FGSM

The Fast Gradient Sign Method shown as Eq. (1) [10][6] is the foundation of many attacks. It is widely used in both the image and audio domains. It combines the original input x with the product of a small parameter ϵ and

the sign of the gradients of the loss function l with respect to x, to create a new adversarial example x'. The small parameter ϵ is used to limit the magnitude of the perturbation; it constrains the step size. The parameter y refers to the correct label for x. FGSM is fast and low-cost because it has a single iteration.

$$x' \leftarrow x - \epsilon \cdot sign(\nabla_x l(x, y)). \tag{1}$$

PGD

Projected gradient descent [13] is a powerful first-order attack. It is based on FGSM but more resource-intensive: it generates multiple iterative perturbations with a smaller step size, projects the result onto a region of interest and finds a point in which loss is maximised.

PGD plays an important role in adversarial training. In ML, adversarial training refers to training models on datasets that contain adversarial examples. The models trained using adversarial examples generated with PGD appear more robust [14] to these multi-step attacks, and it is a prevalent choice for defence. However, adversarial training requires careful tuning of parameters that control the learning process and is not a panacea.

Auto-PGD (APGD)

Auto-PGD [9] makes further progress on PGD. It eliminates the need to choose a fixed step size and utilises an alternative loss function other than the frequently used cross-entropy loss. "Auto" comes from the fact that everything else apart from the number of iterations is automatically adjusted. It allows its step size across iterations to be aware of the optimisation trend and iteration budget this attack has been given.

Similar to PGD, APGD is also helpful in providing a way to evaluate a model's robustness. Because the step size is now flexible, it complements some failure cases of PGD due to fixed step sizes. All test cases provided by the authors of APGD report a lower robust accuracy of the models involved, which indicates that APGD is a stronger advresarial attack. Its adaptive behaviour and self-aware optimisation lead to better results than PGD regardless of the step size chosen.

3 Discussion

3.1 Attacks: Threat Landscape, Feasibility and Future Work

This section discusses potential threats posed by white-box targeted evasion attacks for ASR systems, analyses their implications and proposes future research objectives. Here, threats are events that can inversely affect the confidentiality, integrity or availability of the ASR systems and their users. They could be categorised based on the attackers' motivations, including data theft, financial gain and general sabotage or disruption.

Expanding upon the few scenarios outlined in section 2.2, many more threats exist for each category. For data theft, attackers may target voicemail systems or smart home devices with a camera to obtain audio and video recordings. Sending files, credentials, phishing or spamming emails and texts from victim devices' voice assistants can also lead to further harm. For financial gain, telephone banking services with a voice ID and voice-controlled mobile payment applications could be top victims. The threats posed in general disruption of ASR services can be indirect. The denial-of-service by silencing legitimate voice commands can be immediate - even critical, for healthcare systems - but disrupting transportation navigations and sabotaging smart door locks or other access control systems may indirectly create life-threatening situations for the users.

For the above, the victim systems' capability is a central element to consider. But successful attacks face two key challenges. Drawing a comparison between the image and audio domains can aid the feasibility analysis hence the likelihood of the threats. Attacks on audio pose unique challenges in manipulating their perception and execution [19]. The modifications on pixels can be imperceptible for human eyes [22] and can stay adversarial even when taken as a picture. In contrast, the modified audio samples often suffer from obvious perturbation [23] - even when the target transcription stayed inaudible [20] - and some could not stay adversarial if played over-the-air [6]. For end-to-end attacks, it can be argued that if the attacker already has direct access, placing malware in the victim device may be more effective in compromising the target systems. Despite their generally high success rate, threats associated with this execution method are limited.

Much progress has been made to address them and expand the attack surface. For example, to hide suspicious audible commands, *DolphinAt-*

tack [25] leverages the ultrasound channel with a frequency higher than 20kHz, above the upper bound of human voices and hearing. But like [21], these attacks need to carefully modulate the input and their success rates are more dependent on the audio hardware than the ASR system. Despite the novelty of some attacks, their computational expenses may be too high to be considered practical. The other front, where transmission on air in unpredictable environments nullifies the sample's adversarial properties, seems to remain a challenge even among the state-of-the-art.

Overall, it is currently hard to estimate the full capability of over-the-air attacks because existing publications employ diverse methodologies and seem to lack a standardised framework to represent the practical impact for the wider audience. Adversarial attacks on audio may be a recent subject, but it is never too early to start building a baseline for quantitative representations of threats. Among metrics seldom addressed when publications define their real-life simulated setups, the range of reasonable attack distances and whether a clear line-of-sight is needed are critical aspects that affect the feasibility of execution. A tentative conclusion, for now, is that both execution methods of the attacks are not mature enough to be deployed in practice, and the likelihood of realising said threats is low. More primitive attacks seem more effective.

Apart from outlined above, examining the generation procedure reveals possible problems. Again, one obvious drawback of the attacks is that they all require great prior effort. With the activation requirement in place, many targeted attacks must be tailored, hence relatively costly to execute. Another observation is the repetitive use of existing datasets. Many publications report their results based on the same training data; this could limit the scope of their findings. In reality, attackers can hardly gain white-box access with commercial systems. White-box attacks are learning opportunities to develop attacks against unknown victim systems (black-box). This makes transferability a crucial factor to be evaluated with new attacks, which is sometimes not mentioned. Targeted universal perturbations would pose the most significant threat across systems once they are found. In the future, it would be more than alarming if robust, imperceptible and targeted universal perturbations could be generated as a pipeline process in a timely fashion. As briefly mentioned in section 2.2, it could create opportunities for attacks on a massive scale.

3.2 Defence Strategies and Model Robustness

Addressing the threats of adversarial attacks through defence is a must. The general countermeasures for adversarial examples in ML may not work equally in the audio domain [20] because time dependencies are involved. Common strategies are either transforming the audio input at the pre-processing stage (2.1) to hopefully remove the adversarial properties or adversarial training (2.3). Although a guideline [5] exists, the evaluation of new defences is not straightforward. This calls for a reliable and autonomous protocol similar to the attack scene. Prior work [15] has shown that PGD - while being one of the most effective [24] and the most popular evaluation method - can be fooled into giving false estimations of model robustness.

Because imperceptibility is a primary aim, staying vigilant as users will have limited effect. ASR system developers should incorporate adversarial defence practices in their implementation. It is necessary to include detection, mitigation and response mechanisms to said attacks. However, the question remains: What does it mean for an ASR system to be robust? Will there be a trade-off between robustness and usability? Keep adding more parameters to the models may not be the solution in the long term. Continued efforts such as [4] can keep the definition of robustness up-to-date to better guide adversarial defence research. Since adversarial examples seem inevitable for neural networks, a paradigm shift inspired by a completely different architecture may be seen in the future.

4 Conclusion

This paper introduces the basics of ASR systems and the generation of white-box targeted adversarial examples in the audio domain. It provides an analysis exploring the threats they pose. It presents key challenges of practical attacks utilising adversarial examples. It suggests that the attack feasibility is low and emphasises the need for a framework for impact analysis for the general audience. It points out potential future research directions for stronger attacks and better defence. Future work could involve looking into the adversarial defence aspects. In particular, investigate how the theoretical defence strategies can translate to practical hardening techniques to be employed in ASR systems.

References

[1] *AISec '11: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, New York, NY, USA, 2011. Association for Computing Machinery.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Š rndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Advanced Information Systems Engineering*, pages 387–402. Springer Berlin Heidelberg, 2013.

[4] Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry, 2021.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[6] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text, 2018.

[7] Tao Chen, Longfei Shangguan, Zhenjiang Li, and Kyle Jamieson. Metamorph: Injecting inaudible commands into over-the-air voice controlled systems. *Network and Distributed Systems Security (NDSS) Symposium*.

[8] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models, 2017.

[9] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, 2020.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[11] Wei Han, Cheong-Fat Chan, Chiu-Sing Choy, and Kong-Pang Pun. An efficient mfcc extraction method in speech recognition. In *2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 4 pp.–, 2006.

[12] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[15] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks, 2018.

[16] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. All your voices are belong to us: Stealing voices to fool humans and machines. In Günther Pernul, Peter Y A Ryan, and Edgar Weippl, editors, *Computer Security – ESORICS 2015*, pages 599–621, Cham, 2015. Springer International Publishing.

[17] Paarth Neekhara, Shehzeen Hussain, Prakhar Pandey, Shlomo Dubnov, Julian McAuley, and Farinaz Koushanfar. Universal adversarial perturbations for speech recognition systems, 2019.

[18] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples, 2016.

[19] Yao Qin, Nicholas Carlini, Ian Goodfellow, Garrison Cottrell, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition, 2019.

[20] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding, 2018.

[21] Liwei Song and Prateek Mittal. Inaudible voice commands, 2017.

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R.Fergus. Intriguing properties of neural networks, 2013.

[23] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: Exploiting the gap between human and machine speech recognition. In *Proceedings of the 9th USENIX Conference on Offensive Technologies*, WOOT'15, page 16, USA, 2015. USENIX Association.

[24] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training, 2021.

[25] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. DolphinAttack. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2017.

Effects of habituation on security warnings and ways to minimize it

Salem Wollel
salemgetachew.wollel@aalto.fi

Tutor: Sanna Suoranta

Abstract

Security warnings are crucial in communicating security issues with the user. Thus, they are vital in securing the information system. However, users often ignore those warning messages mainly because of habituation, which refers to the attention reduction due to multiple exposures. Although habituation is generally considered as a beneficial phenomenon, it is often mentioned as a reason for a poor security behavior. Habituation had been difficult to measure. Fortunately, recent researchers use various NeuroIS methods to directly measure habituation. These methods include eye tracking and functional magnetic resonance imaging (fMRI). Those experiments are used to examine the difference in habituation between static and polymorphic warnings. According to the results, the polymorphic warnings were successful in preventing habituation. The effectiveness of such warnings could be further enhanced by the addition of dynamic content to the dynamic dialogs. This paper assesses the results from both the experiments and Implications for further studies based on personal observation.

KEYWORDS: Security warnings, Habituation, Eye tracking, fMRI, polymorphic warnings

1 Introduction

In an ideal scenario, users should not constantly engage with security-related decisions [10]. Consequently, scientists are working on fully automated security solutions. However, in reality, only a few security problems are automatically solved, and the rest are processed according to the input they receive from the user. The security problems are presented to the user by using security indicators and warnings. Those alerting warnings serve as the last line of defense against a potential threat [1]. Thus, users should pay attention to the warnings to identify the best courses of action. However, the warning dialogs are often ignored, which weakens their significance. Even in critical situations, users tend to respond before reading the message and dismiss the warnings right away.

The major factor that contributes to this kind of oblivious behavior is the adaptation to the security warnings. Overexposure to the warning dialogs creates habituation, which refers to the "decreased response to repeated stimulation" [1]. As a result of habituation, warning dialogs that were once catchy become unnoticeable and unremarkable, similar to a familiar wallpaper in a room [5]. Researchers reported that habituation develops after only a few exposures to a new security warning, or even after a single exposure when the dialog resembles a known one [6].

Previously, researchers tried to study habituation by observing its effects on human behavior instead of directly measuring it [5]. It is mainly because habituation is very challenging to measure as it occurs at the neurobiological level. In order to fill this gap, new studies have emerged that use NeuroIS methods, which refer to the study of information systems-related behaviors at the neurobiological level [3]. Eye tracking and functional magnetic resonance imaging (fMRI) experiments are widely used to measure visual activities as an index of cognitive processing and activated brain regions in response to stimuli, respectively.

To slow down the process of habituation, Anderson et al. [3] proposed polymorphic warnings that change their form and appearance in order to prevent users from habituating to them [3]. Polymorphic warnings were examined against static warnings using eye tracking and fMRI experiments, and it was found that the former were more efficient at deterring habituation.

This paper discusses the formation of habituation and the effectiveness of polymorphic warnings over static warnings based on the outputs of the

eye tracking and fMRI experiments. In addition, it outlines potential enhancements that could be applied to the polymorphic warnings to boost their effectiveness. The study also contains personal observations that may be taken into account in subsequent research.

The paper is organized as follows: Section 2 establishes the theoretical development. Section 3 discusses the eye tracking and fMRI experiments, along with their results and discussions on the results from both experiments. Section 4 explores potential improvements to the polymorphic warnings. Section 5 provides implications for future studies. Finally, Section 6 closes with a concluding statement.

2 Theoretical background

2.1 Habituation

When interacting with a stimulus for the first time, humans create a mental model of it. Instead of re-analyzing the experience, we frequently rely on the mental model that was previously constructed for future interactions with the stimuli [1]. With the absence of a mental model, the attention given to the stimulus increases, which is referred to as sensitization. It is the opposite of habituation, which is the "reduced attentional response to repeated exposure to a stimulus" [6].

Habituation is a form of non-associative learning [3] that helps organisms filter irrelevant stimuli in favor of those that are essential to their survival [1]. The development of habituation is accelerated by the increased exposure to the stimuli. It even becomes considerably more severe with the lack of a reward or consequence following the initial action [6].

2.2 Habituation in security warnings

Computer system dialogs are composed of *iconic* and *informational* elements [6]. Iconic elements include icons, size, color and typography. The informational elements refer to the textual elements that communicate a message to the user. Those visual elements are often kept standardized and uniform. Maintaining those aspects as simple, common and identifiable plays a vital role in minimizing confusion and enhancing usability. However, the comfort of the user with these dialogs could lead to a secu-

rity flaw. Users often dismiss system dialogs because they downplay their importance and message.

Anderson et al. [3] found that participants tend to rely on the mental model they created for a previous security warning to interact with other warning dialogs, even if the content and the associated risk have changed. This effect is evident in the decreased attention maintenance of the user [11]. Attention maintenance is measured by tracking the amount of time spent on a repetitive warning dialog. Results show that attention span drops from 15 seconds to 2 seconds only after three exposures to a security warning. This indicates the formation of a mental model, the main cause of habituation.

Studies suggest that strategic user interface design can reduce habituation [3]. Therefore, researchers proposed security warning dialogs that change their form to regain the attention of the user, which was lost due to habituation. Those dynamic security indicator dialogs are termed as polymorphic warnings.

2.3 Polymorphic warnings

Although habituation can occur with properly designed warnings, design improvements can decelerate the habituation process [4]. In comparison to static warnings, polymorphic warnings are more effective in resisting and reducing habituation [1]. Unlike static warnings that are constantly identical [1], polymorphic warnings repeatedly change their appearance [4] by varying their size, color and other graphical elements [1]. Although the graphical elements of those warnings vary, the textual information remains unchanged. Consequently, users could still habituate to the text content by creating a mental model of it.

Due to the difficulty in the measurement, these phenomena had only been explored indirectly [5]. Thus, the effectiveness of the polymorphic warnings in reducing habituation was not clearly understood in the domain. However, recent experimental studies have used NeuroIS tools, such as eye tracking and fMRI, to directly and biologically measure habituation.

2.4 Eye tracking

Eye tracking technologies are useful for identifying areas of interest by measuring eye position and movement such as eye fixation, pupil dilation

and gaze length [3]. Eye tracking is effective in measuring habituation at the neurobiological level. In addition, this tool provides highly accurate and precise outputs [7]. It has been widely used in human-computer interaction to compare the usability and effectiveness of user interfaces [3].

Due to previous exposures to a stimulus, eye fixation and gaze duration are shorter even though individuals are unable to recall that they have seen the stimuli before [3]. This phenomenon is referred to as the eye movement-based memory (EMM) effect, a neurobiological phenomenon where individuals unconsciously assess the objects that they have seen before [7]. Eye-tracking can accurately measure EMM in the case of security warnings and can be used in the analysis of their design [3]. The effectiveness of this method is due to three reasons [7]. First, it is a NeuroIS method that is excellent in capturing hidden mental processes, which are rather difficult to measure. Second, it helps to study the appearance and content of security warnings in a detailed manner. Third, habituation and the given response occur very quickly.

2.5 fMRI

fMRI is another method that excels in its ability to study the activation of the brain in the decision-making process. It measures the changes in the blood flow to different areas of the brain, which is known as the blood oxygen level dependent (BOLD) effect [4]. The changes in the blood flow indicate the activated part of the brain in association with specific mental processes. Therefore, fMRI is widely used to study the brain processing in the case of habituation.

Some experimental designs use fMRI together with eye tracking to better understand habituation. In the case of the eye-tracking experiment, the number of eye-gaze fixations and level of gaze durations are measured by presenting participants with multiple exposures of an image. The results will then be examined in relation to the BOLD response level obtained from the fMRI experiment, in order to acquire a thorough understanding of the habituation effect [7]. Thus, fMRI, with a higher spatial resolution, and eye tracking, with a higher temporal resolution, collaborate by measuring both attention in behavior and the neural activity driving it [9].

3 Experiments

3.1 Experiment 1: The eye tracking experiment

Anderson et al. [3] used Tobii 120, equipment used to track the eye movements of participants for measuring the EMM effect. Participants were directed to sit in front of a camera-equipped monitor that ran Tobii software. The eye tracker was configured in such a way that it captures millions of eye movements by recording fixation at a rate of 60 hertz.

Polymorphic warnings were designed for the experiment [3]. They were presented with nine variations. Participants were presented ten warnings: five polymorphic and five static warnings [7]. Each warning was repeated ten times. Participants saw nine variations and the original image for the polymorphic warnings, and in the case of static warnings, the same warning was repeated ten times. The images were selected at random. To ensure that participants were fully engaged in the examination, they were asked to analyze each warning message and determine whether the dialog is new, similar to a previous image, or even identical [3].

As the static warnings were repeated, the warning repetition, which indicates if it is the first, second, or nth repetition of a warning, declined and became significantly negative [3]. A negative effect indicates a decrease in eye fixation and gaze duration upon repetitive interactions. On the other hand, the warning repetition of the polymorphic warnings was significantly positive, indicating that the eye gaze duration decreases much more slowly for the polymorphic warnings. Figure 1 shows the decrease in the level of fixation for both warning types, static and polymorphic, across the number of reputations.

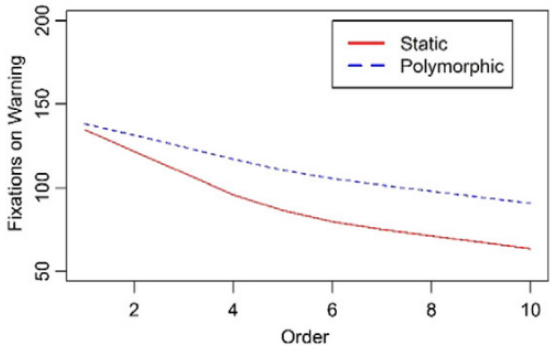


Figure 1. Fixation variation along the repetition number for both warning types [3]

3.2 Experiment 2: fMRI experiment

fMRI uses the same equipment as clinical MRIs [4]. It measures the neural activity in the whole brain on a grid. It identifies the activated brain areas by tracking changes in metabolic demands. The experiment requires participants to lie down on their backs on the fMRI scanner [8]. The image of the security warnings is viewed in the mirror attached to the head coil. This is a reverse reflection from a large monitor outside the scanner that displays the images.

Polymorphic warnings were designed for the experiment [4]. From a pool of 40 warning images, 20 were selected at random for polymorphic treatment, and the other 20 were given for static treatment. 13 polymorphic warnings were made visible one after the other, and the same static warning was repeated 13 times. Each participant was presented with 560 images in five blocks of 6.5 minutes each with a 2 minute break in between, and each image was shown for 3 seconds with a 0.5 second interval in between [8].

Participants were provided with an input device with buttons that enabled them to rate each warning message [8] as new, similar, or identical to a previous image [4], to ensure that participants were engaged in the task.

Four regions of the brain were found that are involved with warning type and repetition number interaction [4]. Those regions reacted in a different way when the warnings were repeated. The left and right superior parietal cortex, which are involved in attentional processing, showed higher activation when polymorphic warnings were presented than static warnings. The regions also showed attention consistency and reduced repetition suppression for polymorphic warnings.

On the other hand, bilateral medial prefrontal cortex and the left retrosplenial cortex, which are involved in the memory retrieval process and default mode activation, show higher activation for static warnings than polymorphic warnings [4]. Default mode network is the area of the brain that tends to decrease while paying attention to a stimulus and rise when the brain is not involved in a mental exercise. The activation in those areas of the brain was consistently higher for static warnings due to the increased memory processing in the default mode network, which is lower for polymorphic warnings. Figure 2 shows the four areas of the brain with their level of activation for the two warning types.

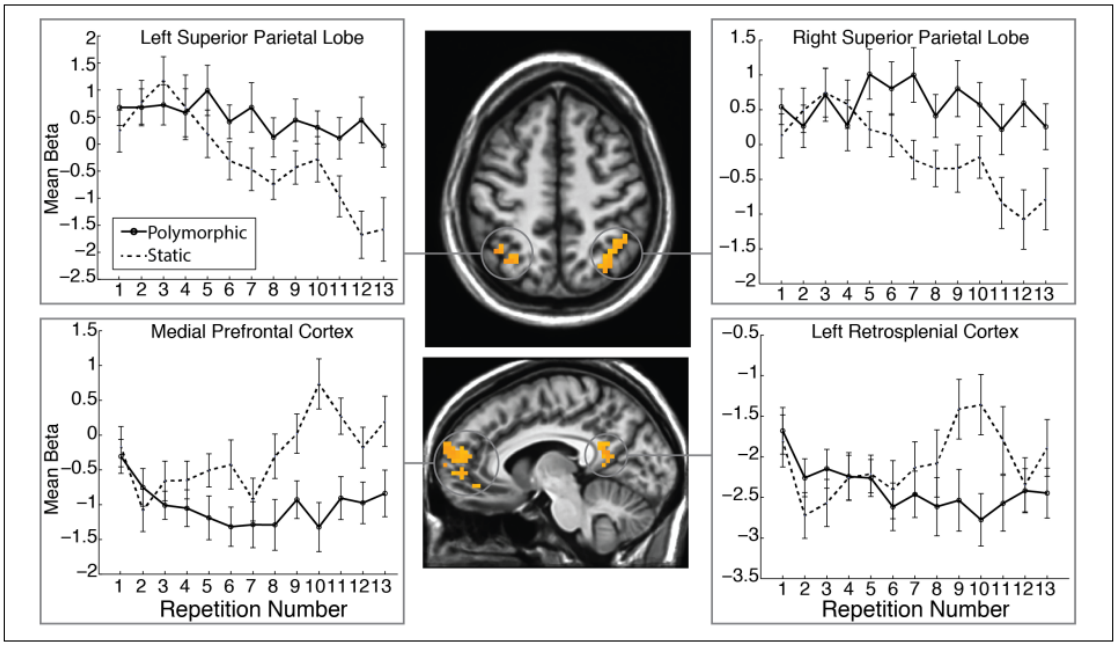


Figure 2. Activated brain regions for both types of warnings. the upper two, left superior and right superior parietal lobe were more activated for polymorphic warnings that the static ones, while medial prefrontal and left retrosplenial cortex shows higher activation for static warnings than polymorphic [4].

3.3 Discussion and limitation

According to the results obtained from the eye tracking experiment, the eye fixation and duration decrease with static warnings, which indicates a reduction in the attention of participants with the repetitive exposures. On the contrary, the slight reduction in eye fixation for polymorphic warnings shows that attention is well maintained and the formation of habituation is rather slow.

The fMRI result reveals that the brain regions that are responsible for attention processing showed higher activation when participants were presented with polymorphic warnings. This emphasizes the occurrence of sensitization and the reduction of habituation. In contrast, brain regions responsible for activities that are reliant on memory retrieval and require less mental processing were activated when participants saw static warnings. The same regions showed less activation when polymorphic warnings were presented. These results indicate that users interact with static warnings by relying on their memory and prior experience, while they tend to examine and focus more on dynamic warnings.

The eye tracking and fMRI experiments empower a better understanding of habituation. However, certain limitations could be mentioned that apply to the study. The fMRI experiment requires participants to lie on

their backs and stay still throughout the experiment [2], with their heads fixed, in order to avoid head movement [8]. In addition, the sound that is generated from the fMRI scanner may cause possible distraction. Therefore, the fMRI experiment could be regarded as artificial as it creates an unfamiliar working environment for participants.

Furthermore, the experiment was conducted within a limited time frame. Habituation could result in a different output over a long period of time [3]. In order to fill this gap, researchers held a longitudinal five-day work week [9] and a three-week field experiment [8]. The output of those experiments supports the idea that polymorphic warnings are more resistant to habituation.

4 Further improvements of polymorphic warnings

Users tend to habituate to dialogs that are static. Although the iconic elements of polymorphic warnings are dynamic, the informational elements remain static [1]. Therefore, users could still create a mental model for the text that leads to habituation to the wording in the polymorphic warnings. Varying the text content could reduce the habituation process and result in sensitization. For this reason, researchers introduced dynamic text in addition to the dynamic design of polymorphic warnings. Constantly differing both in the representation and the semantics of the text content can result in increased sensitization.

In addition, the messages that the security warnings convey can be improved. Often, security dialogs are ineffective in communicating the actual problem, the level of risk associated with it and clear avoidance mechanisms [6]. They frequently use vague language and jargon that an average user would find difficult to comprehend. As a result, users choose to disregard the warnings. Text variation that clearly communicates the amount of risk can persuade the user to comply with the warnings.

5 Implication for further studies

One of the factors that contribute to habituation is the false alarm effect, the reduction in the reliability of a system as a result of a warning that does not result in the anticipated consequence [6]. This effect decreases the reliance of the user on the system and the associated risk. Often,

security warnings are presented even when the activity has no security issue. This greatly decreases the reliance on the warnings, and users learn to disregard and ignore the warnings even in risky scenarios. As a result, presenting security indicators when they are actually necessary could help to reduce habituation by improving reliance. Further research could be done on the methods that automatically and algorithmically solve most of the security-related issues and only present a security warning prompt when the decision of the user is absolutely crucial.

According to this reasoning, immediate consequences after acting on a security warning may increase sensitivity and reduce habituation. For instance, unable to comply with a browser update should result in a slower browser right away. Such kind of immediate costs train the user to read and analyze the warning dialogs before responding. When the user fails to comply with the first security warning, presenting a confirmation dialog that briefly describes the actual consequence and requests the user to reanalyze their decision gives them a second chance to think and adds one more step to the habituation process. Those discussions are left open for future research.

6 Conclusion

Habituation, as a phenomenon occurring at the neurobiological level, has been difficult to measure. NeuroIS methods, such as eye tracking and fMRI experiments, empower the habituation study by providing tangible evidence. The results of the experiment showed that both eye fixation and brain activity associated with attention increase when polymorphic warnings are presented and decrease with static warnings. Thus, the habituation process is much slower for polymorphic warnings than for static warnings. To advance the resistance towards habituation, dynamic text could be incorporated into the warnings. It was also advised that warnings should only be presented when they are absolutely essential and should be followed by swift consequences to resist habituation; however, this requires further investigation.

References

[1] Pranith Abbaraju, Kevin Harmon, and Jaeki Song. Effect of dynamic text on habituation to polymorphic warnings. 2019.

[2] Bonnie Anderson, Anthony Vance, Brock Kirwan, David Eargle, and Seth Howard. Users aren't (necessarily) lazy: Using neurois to explain habituation to security warnings. 2014.

[3] Bonnie Brinton Anderson, Jeffrey L Jenkins, Anthony Vance, C Brock Kirwan, and David Eargle. Your memory is working against you: How eye tracking and memory explain habituation to security warnings. *Decision Support Systems*, 92:3–13, 2016.

[4] Bonnie Brinton Anderson, C Brock Kirwan, Jeffrey L Jenkins, David Eargle, Seth Howard, and Anthony Vance. How polymorphic warnings reduce habituation in the brain: Insights from an fmri study. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 2883–2892, 2015.

[5] Bonnie Brinton Anderson, Anthony Vance, C Brock Kirwan, Jeffrey L Jenkins, and David Eargle. From warning to wallpaper: Why the brain habituates to security warnings and what can be done about it. *Journal of Management Information Systems*, 33(3):713–743, 2016.

[6] Cristian Bravo-Lillo. *Improving computer security dialogs: an exploration of attention and habituation*. PhD thesis, Carnegie Mellon University, 2014.

[7] Bonnie Brinton Anderson, Anthony Vance, C Brock Kirwan, David Eargle, and Jeffrey L Jenkins. How users perceive and respond to security messages: a neurois research agenda and empirical study. *European Journal of Information Systems*, 25(4):364–390, 2016.

[8] Anthony Vance, Jeffrey L Jenkins, Bonnie Brinton Anderson, Daniel K Bjornn, and C Brock Kirwan. Tuning out security warnings: A longitudinal examination of habituation through fmri, eye tracking, and field experiments. 2018.

[9] Anthony Vance, Brock Kirwan, Daniel Bjornn, Jeffrey Jenkins, and Bonnie Brinton Anderson. What do we really know about how habituation to warnings occurs over time? a longitudinal fmri study of habituation and polymorphic warnings. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2215–2227, 2017.

[10] Ricardo Mark Villamarín Salomón. *Improving computer-system security with polymorphic warning dialogs and security-conditioning applications*. PhD thesis, University of Pittsburgh, 2010.

[11] Zarul Fitri Zaaba, Christine Lim Xin Yi, Ammar Amran, and Mohd Adib Omar. Harnessing the challenges and solutions to improve security warnings: A review. *Sensors*, 21(21):7313, 2021.

Using Deep Reinforcement Learning to solve the Service Placement Problem of the Fog Computing system

Samath Lenaduwa Lokuge
samath.lenaduwalokuge@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

This research survey paper will focus on the Service Placement Problem in the Fog Computing system due to growing demand of complicated applications used by "Internet of Things" devices. Our target is to focus on Deep Reinforcement Learning methods that have been proposed by several researches and how they solve the Service Placement Problem, the potential to combine these methods together and concluded if Reinforcement Learning and Deep Neural Networks are viable tools to solve Fog Computing problems.

KEYWORDS: Fog Computing, Deep Reinforcement Learning, Service Placement Problem, Quality of Service, Internet of Things

1 Introduction

The rise of Internet of things (IoT) devices in our lives has prompted the advancement of mobile networks. Devices like virtual reality, augmented reality, multimedia delivery, and artificial intelligenc demand broad bandwidth, low response latency, and high computational power [1]. IoT data is mostly generated in a distributed way, sent to the cloud for processing and then delivered to distributed user or IoT devices, often located very near to the data sources. Hence processing IoT data in central tends to high communication delays and affects the QoS(quality of service) perceived by the users [2]. Fog Computing (FC) systems are designed to bring the Cloud closer to the user by localizing smaller Cloud nodes called Fog nodes. Due to FC being a more complex and distributed system, the Service Placement Problem (SPP) has being defined as the main challenge when forming an efficient and cost-effective system. This paper will discuss the Service Placement Problem and how Deep Reinforcement Learning (DRL) can be used to solve certain issues.

2 Background

2.1 Fog Computing

Fog Computing (FC) is a digitilized platform that aims to distribute the Cloud resources into local areas as Fog Nodes [3]. These Nodes interact with the end-users and collect localized data to analyze. Only the necessary data is forwarded to the central Cloud, making this system faster than an single Centralized Cloud network. The FC system can be visualized as seen in Fig. 1:

FC system can be split into three layers, which are the End-, Fog-, and Cloud Layers [3]. End layer is where most of the end-users and IoT devices are located. Service requests from end-users are collected in this layer and then forward to the upper layer of the FC paradigm. The Fog layer is in between the other two layers. It contains devices called Fog Nodes (FNs) that process, store, and handle requests. FNs consist of routers, gateways, switches and specific Fog serves that are linked to the highest layer. Cloud layer is the upper-most layer in the structure. The

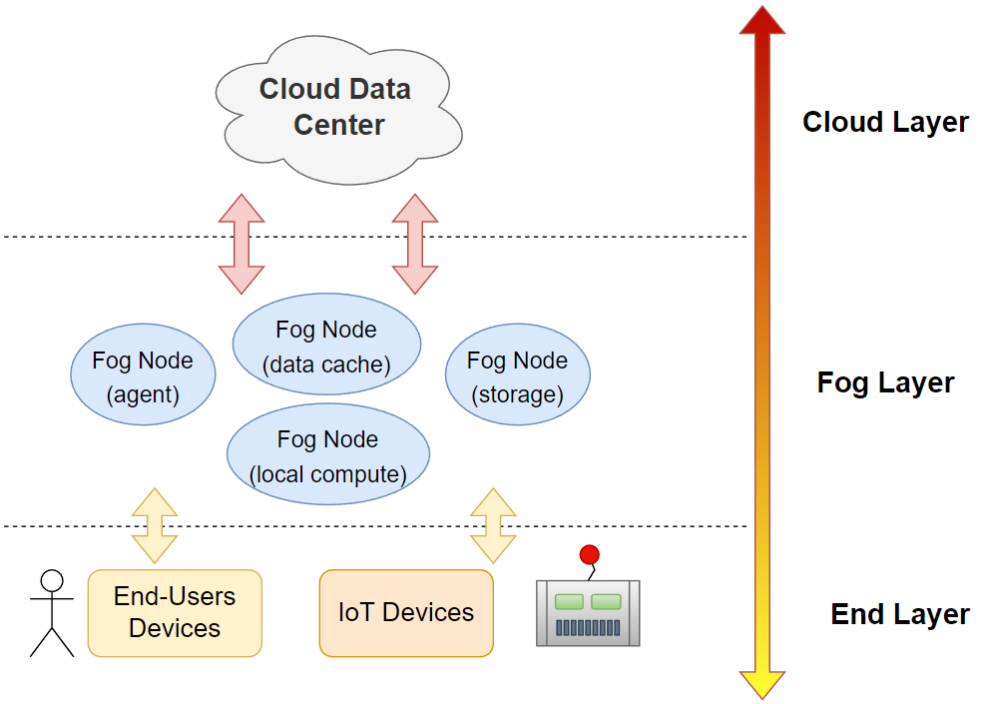


Figure 1. Generic Fog computing architecture

Cloud servers and Data Centers reside here handling analyses and storing data. Most Machine Learning algorithms are performed in this layer.

FC system consists of several advantanges compared to the Cloud Server, such as lower latency, bandwidth saving, scalability or mobility [3]. However, due to FC being newer and more complex, emerging issues need to be dealt in the form of a Service Placement Problem.

2.2 Service Placement Problem

The Service Placement Problem (SPP), as defined by Apat et al [2], is the process of allocating service request from multiple cllients to specific FNs, while maintaining Service Level Agreement (SLA) and end-user Quality of Service (QoS). SPP's primary target is distribute services on FNs so that it improves the QoS level experienced by end users and the maintains the requirements of the FC system [4]. To do this we can divide the SPP in to different objectives as shown in Fig. 2:

Each individual objective provides support for the main target of SPP as stated above. For example, the services should have the closest proximity to end users [4]. End users can be served by a FN that is in there

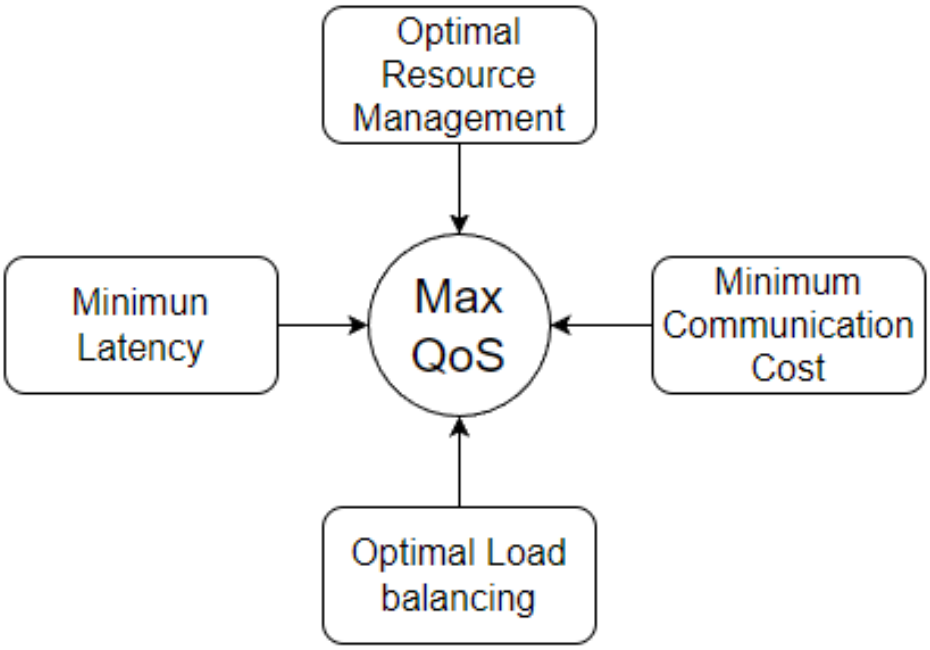


Figure 2. Service Placement Problem objectives

area. Every FN must know the status of data in its fog cluster [2]. This can achieved by focusing on real-time adaption of the Fog Node topology. Communication cost, energy, and service time can be optimized by allocating resources so that it fulfills the QoS expectations of the end-users. If a specific FN cluster is overloaded, traffic should be distributed among neighboring FN clusters via load balancing. If this process of balancing doesn't meet the expected QoS, then the end-users will receive high latency while using the services. Many methods are used to solve the SPP and maximize QoS. This paper will focus on the DRL method that we will go more in depth on.

2.3 Deep Reinforcement Learning

To understand Deep Reinforcement learning (DRL), Reinforcement learning (RL) has to be explained first. The foundation of RL can be seen in Markov Decision Process (MDP); a mathematical framework often used in decision making problems.[5] Using enviroments based on MDP models, RL can use previous mistakes to learn an optimal strategy. Unlike Supervised and Unsupervised learning, which targets a direct short-term goal, RL focuses on a long-term increamenting goal. A fundemental charasteristic of RL is that it learns from the consequences of its actions, by interacting with the enviroment, and then alter its future actions based

on the reward it acheives [6]. Deep neural networks (DNN) automates the composition of higher-level features using lower-level ones, such as images and local combination of edges, using raw input data [7]. DNN uses Distributed representation, where several features can represent a input or vice versa.

DRL is RL with the guidance of DNN. This type of RL has being prominent in modern machine learning methods, as deep learning helps RL to perform end-to-end learning with a gradient descent, meaning that the RL doesn't depend on domain knowledge as much [7]. This means that traditional RL methods can be automated, which saves time, simplifies tasks and makes sure that they would be completed.

3 Literature

3.1 Actor-Learner framework

In order to handle larger datasets without sacrificing a lot of training time, a distributed agent called "Importance Weighted Actor-Learner Architecture" (IMPALA) was developed by Espeholt et al [8]. IMPALA uses multi-task reinforcement learning and uses actors to interact with a centralized learner using experiences, such as status, actions and rewards. It's orginal function is to conduct accurate machine training without significant losses in data efficiency or resource utilization. Goudarzi et al [9] propose to use this framework to reduce costs of DRL agents, which implement placement techniques for IoT devices in the Fog Computing System. Here, several actors are also used to interact with various enviroments to form experiences, which are sent to the learner to train for the optimal policy. The actors will then reset all there values during every policy update. However, these actors can learn from their previous experiences that results in a reinforcement method, which reduces exploration costs drastically. The Actor-Learner framework (ALF) helps to identify the shortest path between FNs using IMPALA. This can be example of minimizing communication cost. As stated in the SPP background, communication cost is resource that has be optimized in order to achieve a expected QoS later on the Service Placement.

3.2 Mixed-integer linear programming

FC system requires to maintain a higher scalability and mobility compared to a tradition cloud computing system. Service Function Chaining (SFC) is a network softwarization that creates virtual chains of connected Micro-Services [10]. Due to its dynamic behaviour, SFC allocation mechanism results in poor use of resource usage and scalability. Santos et al [10] have proposed a DRL enviroment known as Mixed-Integer Linear Programming (MILP), in which agents learn the SFC allocation in the FC system without prior infomation. MILP can handle the dynamic behavior of SFC allocation but at the expense of higher execution time. This improves the scability of the network and connection between nodes and edges by optimizing resource management.

3.3 Experience-based replay RL method

Multi-acess Edge Computer (MEC) paradigm is used to ensure lower execution time for edge computation on devices. MEC has challenge regarding it's computation offloading scheduling in a dynamic network. Li et al [11] proposes Experience-based replay RL algorithm (EBRL) that can improve convergence performance using collected transformation knowledge from it's experience pool. To solve the challenge, the offloading problem is first converted to a MDP model and then EBRL method is used. The result shows that EBRL method effectively increases stablity and training speed compared to other methods used in the MEC computation offloading. This, like MILP, is an example of resource management.

3.4 Multi-Update Deep Reinforcement Learning method

Multi-update Deep Reinforcement Learning algorithm (MDRL) was originally proposed by Hao et al [12] when discussing the improving long-term mean reduction of delay, which is formulated as a Markov decision process. Generally, Traditional methods have large action spaces making the problem harder to solve. Action space can be greatly reduced by this algorithm, which uses a novel exploration strategy and update method. MDRL is used in the End layer, where edge nodes have limited capabilities when dealing with the number of requests they receive from IoT devices. The application of MDRL allows optimum load balancing of storage space and compute resoureces between FNs.

3.5 Parameterized Deep Q Network

The Parameterized Deep Q Network (PDQN) method was proposed by Liu et al [13] that can be used to handle online service placement and computation resource allocation issues of a FC system. The PDQN is a extension of DQN that is computationally tractable with a continuous action space by using two sections to make decisions; action generation and network training. Action generation picks actions based on decisions that networks compute. Then the Network Training is responsible of training these networks with recorded experiences. The method was evaluated by several simulations for its performance and PDQN performed better than other approaches using proper parameters due to its significant lower total latencies. Hence, this is an example of minimizing latency.

4 Discussion

The SPP objectives serve to maximize QoS, which is a measure of satisfaction the user has when using a application that is served by the FC system. Therefore, the objectives can be reflected by how they would maximize this measurement of satisfaction. The resources have to be optimized to have a service that maximizes QoS. The failure of resource management can negatively affect the end-user in two ways. One way is directly effecting the user experience by making services slow and unresponsive. The other way is indirectly effecting the user by high resource costs that results in users having to pay higher rates for their services. The resources in FC system are dynamic and have to adapt to current service requirements. Fortunately, the concept of DRL is to continuosly adapt its MDP modeled strategy. RL learns from prior mistake so its ideal to be used in optimization problems and DNN can help to model FC system with large number of FNs and services. These are important in load balancing as the FN clusters adapt to different quanities of traffic and learn from their previous failures of data distribution between neigboring clusters. The latency of services depends on all of the factors mentioned and directly affects the satisfaction of the user experience. This too can be minimized with the support of these specific methods.

The DRL methods described in the Literature section offer solutions that maximize QoS by optimizing one of the objectives of Service Placement Problem as stated in Fig. 2. These methods are versatile since they solve different objectives of the SPP target. It's possible to further maximize QoS by using multiple DRL methods at the same time. Some combinations of methods would be more effective than others. It would be clear to assume that having multiple methods working on the same goal like MILP and EBRL would cause problems as they would try to solve resource management idependently (though it is possible to create a new method that makes these methods cooperate with each other). For focusing solely on resource optimization, ALF and either MILP and EBRL would be ideal as they can optimize the resources and maximize the QoS without complicating the other objectives. Likewise, for focusing on the Fog Clusters and the distribution of traffic among them, MDRL and PDQN work together in allocation the FNs, the former with balancing the storage space and the latter with minimizing total latency. However, these are speculations of possible combiantions of DRL methods and further research using experiments have to be conducted to confirm these hypotheses.

5 Conclusion

The research has show that Deep Reinforcement Learning methods are used in different areas of the Service Placement Problem and increase the end-user satisfaction of the FC system. It can be speculated whether these methods can cooperate with each other to make a multi-DRL method. This theory can be further investigated and tested to see if it works. Either way, Deep Reinforcement Learning will remain a prominent Machine Learning method for the field of Fog Computing.

References

[1] Jianyu Wang, Jianli Pan, Flavio Esposito, Prasad Calyam, Zhicheng Yang, and Prasant Mohapatra. Edge cloud offloading algorithms: Issues, methods, and perspectives, January 2020.

[2] Hemant Kumar Apat, Bibhudatta Sahoo, and Prasenjit Maiti. Service placement in fog computing environment. 2018 International Conference on Information Technology, December 2018.

[3] Farah Ait Salaht, Frederic Desprez, and Adrien Lebre. An overview of service placement problem in fog and edge computing, June 2020.

[4] Hani Sami, Azzam Mourad, and Hadi Otrok. Demand-driven deep reinforcement learning for scalable fog and service placement, September 2022.

[5] Filippo Poltronieri, Mauro Tortonesi, Cesare Stefanelli, and Niranjan Suri. Reinforcement learning for value-based placement of fog services. University of Ferrara, May 2021.

[6] Kai Arulkumaran, Marc Peter Desenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey, September 2017.

[7] Yuxi Li. Deep reinforcement learning: An overview, January 2017.

[8] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, February 2018.

[9] Mohammad Goudarz, Marimuthu Palaniswami, and Rajkumar Buyya. A distributed deep reinforcement learning technique for application placement in edge and fog computing environments, October 2021.

[10] Jose Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. Resource provisioning in fog computing through deep reinforcement learning, May 2021.

[11] Kexin Li, Xingwei Wang, and Bo Yi. Experience-based computation offloading by deep reinforcement learning for multi-access edge computing network. ACM Digital Library, July 2022.

[12] Hao Hao, Changqiao Xu, Lujie Zhong, and Gabriel-Miro Muntean. A multi-update deep reinforcement learning algorithm for edge computing service offloading. ACM Digital Library, October 2020.

[13] Tong Liu, Shenggang Ni, Xiaoqiang Li, Yanmin Zhu, Linghe Kong, and Yuanyuan Yang. Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing, February 2022.

Adversarial attacks and defenses on neural networks

Samu Kähkönen
samu.kahkonen@aalto.fi

Tutor: Blerta Lindqvist

Abstract

Neural networks are one of the most prominent machine learning technologies, and they achieve state-of-the-art performance on many types of tasks. However, they are vulnerable to attacks by an adversary. Adversarial attacks consist of generating adversarial examples that cause the target model to mislabel the input data, e.g misclassify images. Numerous studies have been conducted on adversarial attacks and on ways to defend from them (adversarial defenses). This paper discusses the current state of adversarial machine learning and provides an overview of many of the most common types of adversarial attacks, and a brief discussion on adversarial defenses.

KEYWORDS: Machine learning, Neural networks, Adversarial attacks, Adversarial defenses,

1 Introduction

In recent years machine learning has been applied in various fields, especially with the advent of neural networks. Neural networks have enabled great progress in areas such as computer vision, paving the way for, e.g. image classification. However, neural networks have some curious

counter-intuitive properties, that leave them vulnerable to attacks by malicious third parties [13], [7]. Thus the robustness of neural networks has been a highly studied topic of interest in recent years.

Various studies have been conducted on adversarial attacks that abuse these weaknesses in many applications, such as self-driving cars [4] and voice interfaces [1]. These vulnerabilities have been even seen as an inherent problem of neural networks, especially in safety critical contexts [11].

This paper discusses the vulnerabilities in neural networks, methods to abuse these vulnerabilities and ways to defend from such attacks. The most common types of adversarial attacks can be divided into four categories: Evasion attacks [7] [11], [5], [2], [3], data poisoning attacks [10], [15], [16], [8], [14], model extraction attacks [9] and inference attacks [6]. In this paper we focus on the first two. Evasion attacks target neural networks in the inference phase, after the model has been trained. Data poisoning attacks on the other hand target the training phase of the model, or more accurately, the training data set.

This paper discusses the theory behind these adversarial attacks as well as briefly discusses the ways to resist them. Chapter 2 will go over the basic theory behind adversarial attacks. In chapter 3 and 4 we will discuss the most common types of adversarial attacks, evasion and poisoning attacks respectively. Chapter 5 gives an overview of the theory of adversarial defenses. Finally, chapter 6 provides a conclusion on the current state of adversarial attacks.

2 Basic theory behind adversarial attacks

Let x denote a data input to a neural network model, with a correct label of l_1. The goal of an adversarial attack is to generate a data input x' such that it will be virtually indistinguishable from the original data x (by both humans and the machine learning systems), but changed in minor ways such that the model C will misclassify the input with some label $l_2 \neq l_1$ [13]. Or as a formula

$$C(x) = l_1$$

$$C(x') = l_2$$

Optimized over

$$min(D(x, x')), \text{Where} \quad D(x, x') \quad \text{is some distance metric}$$

If the attacker is not concerned what exactly this l_2 label is, only that it is incorrect, the attack is called an untargeted attack, and if the attacker wants to achieve misclassification into a specific label, the attack is targeted.

The goal of the attacker is to generate an adversarial example that will misclassify the data sample with as small perturbations as possible. It is not trivial to decide what kind of parameter is the best for measuring the distance between the original and the modified example, that would be as close to human perception as possible. regarding image classification, there are three metrics that are used for estimating this distance that are widely used in the literature [2], and those are different L_p norms. The L_0 norm measures the amount of pixels that are changed, the L_2 norm is the Euclidean distance between the images, and the L_∞ norm measures the maximum change to any pixel between the images. Most existing studies use some or all of these metrics.

One important property of adversarial attacks is transferability. This means that adversarial examples that manage to fool some model will often also cause misclassification in other models, even ones that are trained differently with different parameters [2].

Various studies in recent times have discovered that generating adversarial examples is in fact quite easily achieved [7]. Especially for neural networks that do not employ any kind of defensive measures, it is almost trivial to find adversarial examples. The potency of adversarial defenses is discussed in more detail in a later chapter.

The next two chapters will go over different kinds of evasion and data poisoning attacks respectively.

3 Evasion attacks

Evasion attack is a type of adversarial attack where the attacker does not need to have the ability to modify the training data set of the model, and instead creates small worst-case perturbations so that the model will misclassify data with the wrong label and high confidence [7]. There are a few different methods to achieve this misclassification. Some of the important attacks are fast gradient sign method (FGSM) [7], projected gradient descent (PGD) [11], Carlini-Wagner (CW) [2] and Zeroth order optimization (ZOO) [3].

The attack methods usually are chosen depending on whether the attacker is targeting a white-box system, where the attacker knows how the model are trained and what parameters are used. For these kinds of systems, FGS, PGD, and CW are widely used. Usually in real-world scenarios however, the configurations and parameters of neural networks are not released to the public and it is not practical to assume the ability to conduct white box attacks. This kind of closed system where the attacked is unable to retrieve this information from the system, is called a black-box system. The only information available for the attacker in this case will be the input data and the output labels. ZOO is a powerful attack for black-box systems.

The basic principle behind evasion attacks is to find the gradient of the classification model and iteratively take steps toward the opposite direction, known as gradient descent.

3.1 Fast Gradient Sign Method

FGSM [7] is a technique used in adversarial machine learning to create adversarial examples - modified versions of input data that are designed to cause a machine learning model to make incorrect predictions or classifications.

FGSM works by calculating the gradient of the loss function of a neural network with respect to its input data, and then modifying the input data by a small amount in the direction of the sign of the gradient. The size of the modification is determined by a hyperparameter called the epsilon value, which controls the magnitude of the perturbation. By adjusting the epsilon value, an attacker can control the severity of the attack and the degree of similarity between the original and adversarial inputs.

The FGSM attack is designed to be a fast method of generating adversarial attacks, and not necessary the optimal [2].

3.2 Projected Gradient Descent

The PGD attack [11] is a stronger and more effective attack than the simpler FGSM, which only makes a single-step perturbation in the direction of the gradient. In contrast, PGD iteratively perturbs the input, taking multiple steps to make smaller perturbations, and projecting the perturbed input back onto the constraint set after each step.

By iteratively adjusting the perturbation in this way, PGD can create ad-

versarial examples that are optimized to fool the target machine learning model. PGD is particularly effective at generating adversarial examples that are robust to various types of defenses, including those designed to detect or mitigate adversarial attacks. [11]

PGD can be used with a wide range of machine learning models, including deep neural networks, and can be applied to a variety of input data types, including images, audio, and text. However, PGD attacks can be computationally expensive, as they require multiple iterations of the optimization process, and they can also be more difficult to implement than simpler attacks like FGSM. Nonetheless, PGD is a powerful and effective tool for testing the robustness of machine learning models to adversarial attacks.

Auto-PDG is a further improvement on PGD introduced by Croce et al. [5]. The idea behind the improvements it to have a parameter-free vesion of PGD, as improper tuning of parameters has been a problem in evaluating the robustness of neural networks. Auto-PGD takes into account the available number of iterations, and divides them into an exploration phase, intended to find good starting points for the algorithm, and an exploitation phase, to then maximize the function locally. Unlike PGD, Auto-PGD does not have a fixed size. Instead it starts with a larger step-size in the exploration phase, and reduces it iteratively. The reduction is also not scheduled in advance, and instead follows the trend of the optimization.

3.3 Carlini-Wagner attack

The CW-attack [2] is an optimization based attack that will search for the smallest possible distance between the original and modified data, that will cause a misclassification by the target model. The attack is implemented for the three L_p norms discussed before, L_0, L_2 and L_∞.

The L_2 attack is the simplest of the attacks, as the distance metric is differentiable. The goal is to minimize the distance with regard to the best objective function defined in the paper. The problem is then solved with multiple starting-point gradient descent, to reduce the probability of getting stuck in a bad local minimum. [2]

The L_0 metric is non-differentiable, which means the problem cannot be solved with gradient descent. Instead the L_0 attack uses an iterative algorithm that identifies pixels with little effect to the classifier output, and fixes them such they will never be changed. The L_2 attack is used

to identify which pixels will not be modified. As the set of fixed pixels grow, eventually the algorithm will have identified a minimal subset of modifiable pixels that can generate an adversarial example. [2]

The L_∞ attack is also solved with an iterative algorithm that has a gradually decreasing threshold, such that any term exceeding that value gets penalized.

According to the tests conducted in the paper, the L_2 performed the best with nearly all adversarial examples being visually indistinguishable from the originals on the MNIST-dataset. The L_∞ attack was the second best, having a few distinguishable adversarial examples, while most of the examples generated by the L_0 attacks were visually distinguishable.

3.4 Zeroth order optimization

The ZOO attack [3], unlike the previously mentioned attacks, is a black box attack. In other words, it only requires access to the input and the output. The attack takes inspiration from the previously discussed CW-attack, however because of the black box constraint, back propagation cannot be applied. Chen et al. [3] solved this problem by a few modifications to the CW attack. Firstly, they used a modified loss function that only takes into account the output of the neural network and the target label. Secondly, they showed that the gradient can be sufficiently estimated by only using zeroth order methods (i.e. without derivatives) by evaluating the objective function at very close points. The estimated gradients are then used in coordinate descent to find the best perturbations at the chosen points.

The remaining problem is that estimating the gradients is extremely expensive in the black box scenario. ZOO solves this by using hierarchical attack-space dimension reduction to balance optimization efficiency and attack complexity. Importance sampling is also applied to find the most important coordinates to use in the coordinate descent [3]

The paper states that ZOO achieves performance comparable to the CW attack, and significantly outperforms existing black box attacks that are based on model substitution.

4 Poisoning attacks

Data poisoning attacks differ from evasion attacks in that they attack the model already during the training period. This naturally is more difficult to achieve than evasion attacks, since it requires the attacker to have access to the training data of the target model, as well as the ability to modify that data. Some prominent data poisoning attacks that will be discussed in this chapter are label flipping attacks, backdoor attacks and clean label backdoor attacks and [10], [15].

4.1 Label flipping attack

Label flipping attacks are one of the more simple forms of data poisoning attacks. Their goal is to taint a training data set by changing labels of data points, so that a model trained on the set will be less accurate. Xiao et al. [16] demonstrated this kind of attack that uses gradient ascent to identify the labels that will cause the largest classification error on the target model. The paper showed that this kind of attack can be highly effective. However, the weakness of label flipping attacks is that the bad performance of affected models can be detected by evaluating the model on a test data set [14].

4.2 Backdoor attack

Backdoor attacks are a more insidious type of data poisoning attack, which aim to plant a backdoor into models trained on the poisoned data set [8]. The backdoor is implemented with embedding a backdoor trigger to data points of the original data set, which is some kind of pattern that the model will be trained to classify with the label of the backdoor target. In image classification, a simple backdoor trigger could for example be a geometric shape planted in the corner of an image. The backdoor is then activated during the inference of the model, by modifying the input with the backdoor trigger.

Backdoor attacks are also considerably harder to detect than label flipping attacks, because the poisoned model can perform as well as a clean model on clean samples. The poisoning only affects examples that include the backdoor trigger. This type of attack relies on the assumption that the adversary can make arbitrary poisoned inputs, even ones that are clearly mislabeled [14]. This means that the attack can be detected by a filtering

process, or even by just human evaluation of the training data set.

4.3 Clean label backdoor attack

The clean label backdoor attack is an improvement over the previously discussed backdoor attack introduced by Turner et al. [14]. The goal of this attack is to plant a backdoor the target models similarly to the backdoor attack. Unlike the backdoor attack, this version tries to keep the poisoned input data and labels consistent, even to human inspection, thus removing the main weakness of the backdoor attack. In their paper they discovered that using a restricted version of the backdoor attack by Gu et al. [8], where true labels are not changed and only the backdoor trigger is added to the target class, is not effective.

According to their analysis, this is caused by the fact that the poisoned samples can be correctly classified without relying on the backdoor trigger. To solve this problem, they perturb the poisoned samples such that learning the salient characteristics of the inputs becomes more difficult. As many widely used datasets contain numerous low quality images, it turned out to be possible to perturb the inputs in a way that the data-label pairs remain plausible to humans in manual inspection.

5 Adversarial defenses

Studies on adversarial defenses have shown that it is possible to resist at least certain types of adversarial attacks by using adversarial samples in the training set of the model [7], [11]. This kind of defense is called adversarial training, which is shown to be quite resistant to the type of attack it is trained on, however it comes with a cost of accuracy on natural benign samples. The major disadvantage of adversarial training is that it requires adversarial samples during training, and it will not be able to be robust against unknown attacks, as you will not be able to prepare for them during the training phase.

One type of adversarial defense in defensive distillation [12]. Defensive distillation is an approach to attempt to reduce a larger model to a smaller distilled model. First, you train the model on the training data as is usually done. This model will be called the teacher in defensive distillation Then the teacher model is used to obtain soft labels on the training data set. Finally, a new model is trained, but instead of using the training

data's hard labels, we use the soft labels provided by the teacher model to train this new distilled network. Initial reports showed that this method was a promising defense to adversarial attacks, however Carlini et al. [2] showed in their paper that defensive distillation has only a marginal benefit against stronger, improved attacks.

Madry et al. [11] explore the robustness of neural networks through robust optimization techniques. In the paper they seek to identify methods to specify a certain guarantee of security, that would be universal against all kinds of attacks. They use the previously discussed PGD as the strongest first-order adversary, and manage to train a robust neural network on the MNIST-dataset that achieves close to 90% accuracy against the strongest PGD adversaries. However, their model on the more complex CIFAR10-dataset did not achieve the same level of robustness, misclassifying over half of the adversarial examples.

6 Conclusion

As a result of neural networks becoming the state-of-the-art in many machine learning tasks, it is increasingly important to achieve robustness against adversarial attacks. It is especially important to have guarantees of security in safety-critical use cases. Through studying the current state of the research it is clear that there is still much to do in order to achieve that, as there still is no conclusive proof of defending against the strongest adversaries. However, there are studies that show promising results indicating that defending against adversarial attacks is possible. Undoubtedly more research is still needed before neural networks can be used in their full potential.

References

[1] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David A Wagner, and Wenchao Zhou. Hidden voice commands. In *Usenix security symposium*, pages 513–530, 2016.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[3] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM*

workshop on artificial intelligence and security, pages 15–26, 2017.

[4] Alesia Chernikova, Alina Oprea, Cristina Nita-Rotaru, and BaekGyu Kim. Are self-driving cars secure? evasion attacks against deep neural networks for steering angle prediction. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 132–137. IEEE, 2019.

[5] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[9] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1345–1362, 2020.

[10] Jing Lin, Long Dang, Mohamed Rahouti, and Kaiqi Xiong. Ml attack models: Adversarial attacks and data poisoning attacks. *arXiv preprint arXiv:2112.02797*, 2021.

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[12] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016.

[13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[14] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

[15] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[16] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI 2012*, pages 870–875. IOS Press, 2012.

A Survey on Participant Selection for Mobile Crowdsensing

Sandeep Aryal
sandeep.aryal@aalto.fi

Tutor: Zhanabatyrova Aziza

Abstract

Mobile Crowdsensing is a cost-efficient way of collecting private and public data. This has been possible due to the advancement and availability of sensing devices. Similarly, the pool of sensing users have been rising which has increased the challenges of selecting effective users. The objective of this paper is to understand the MCS challenges and solutions. In addition, this paper summarizes three different participation selection models. Recent research works were used for studying and summarising the topic.

KEYWORDS: crowdsensing, mobile crowdsensing MCS, MCS types, MCS challenges, MCS participation selection model, Willingness-based participation selection, User characteristics aware participation selection, quality aiming participation selection

1 Introduction

Crowdsensing refers to collecting data from sensors on mobile devices, including smartphones, wearables, and IoT devices, for analysis to extract insights and improve system performance. Its increasing popularity is attributed to the extensive utilization of mobile devices and low-cost sensing technologies. The use of advanced sensing technologies, machine learning, and blockchain can enhance the efficiency and security of data collection and analysis. The increasing demand for more intelligent and responsive systems is driving the development of crowdsensing applications in various domains. [20] At the same time, it is costly to store and process all the data collected from crowdsensing. Therefore, it is important to effectively identify and select the well-suited participant from the large user pool. [9, 1]

This document focuses mainly on mobile crowdsensing (MCS) types and their challenges. It provides an overview of MSC challenges and summarizes three different participation selection methods/models. In addition, possible solutions to MSC challenges are defined and future research direction is suggested. Recent research documents are used for the study of the mentioned topic. This document is divided into seven sections and the second section introduces the MCS and its types. The third section highlights MCS challenges and solutions. Similarly, section four, five and six describes the three different participation selection model. In the end, a discussion and conclusion are provided.

2 Crowdsensing and Types

Crowdsensing is a good option for organisations, as it enables them to gather valuable data with minimal investment. This is made possible through tools like Mobile Campaign Designer which makes creating crowdsensing platforms simple. This tool enables organizations to set the specifications for their crowdsensing campaigns, and it generates the appropriate source code and executables required for a personalized crowdsensing initiative. This indicates that even individuals without technical expertise can establish a crowdsensing application in just five minutes. [4]

Five types of crowdsensing can be categorized based on two criteria: the level of user involvement in the crowdsensing process and the type of phenomenon being measured. The first criterion separates crowdsensing into

two categories: participatory and opportunistic. Participatory crowdsensing necessitates users to actively transmit sensor data to a server. On the other hand, opportunistic crowdsensing operates automatically and entails minimal user engagement to transmit the collected information. [4]

2.1 First Criterion

Participatory crowdsensing involves users voluntarily contributing information or participating in exchange for rewards. There is thus a greater variety of data available and this leads to more localized and precise knowledge. For example, it has made it possible for a local administrator to maintain a service like water drainage lines and road conditions. In addition to taking pictures, users can also use their smartphone's GPS sensor to pinpoint the location of an issue, allowing MCS to better address the issue. The Google Crowdsource application is another example, which requires users to provide input so that Google's service can be improved. Users get rewarded for submitting information and in turn, Google gets to collect correct data from users. [12]

Opportunistic crowdsensing depends on utilizing devices such as smartphones that come with different sensors to gather data from the environment in proximity. For example, a fitness tracker sends statistics such as heart rate, body movements and sleep data to the company's server which analyses them amongst all users when connecting to your mobile device. This allows users to compare their data with the community using the same fitness band, improving health conditions, road traffic situations and more. For example, the company will analyse the data collected continuously by the fitness band and provide the user with comparisons and health advice. [12]

2.2 Second Criterion

The second criterion used to categorize crowdsensing is the type of phenomenon being measured. This results in three categories: *physical infrastructure, environmental* and *social crowdsensing.* Infrastructure crowdsensing is used to measure public infrastructure, such as traffic congestion and road conditions. Environmental crowdsensing is used to measure aspects of the natural environment, such as the number of habitats affected by wildfire, air pollution levels and water levels. Social crowdsens-

ing is used to gather data about people's social lives, such as the movies or super-market visited by an individual. [4]

3 Crowdsensing Challenges

Most Mobile Crowdsourcing (MCS) systems depend on people's voluntary involvement and support. Engaging in a crowdsourcing initiative through their smartphones can expose users to potential security and privacy threats as they not only deplete their resources but also subject themselves to risks such as sharing their data along with location markers. Additionally, MCS providers aim to get high-quality contributions from the crowd, but they may not always be trustworthy, accurate, or well-intentioned. In conclusion, there are four significant challenges in MCS systems: incentive, task management and privacy protection and security, and quality assurance. [16]

3.1 Incentive

An incentive is a form of impetus or inspiration that motivates an individual to take action or exert more effort. This is especially significant in situations where the available resources, such as storage and power capacity, are limited in devices like wearables and mobile phones, or when the data collected is sensitive. To implement an MCS system on a large scale, a substantial number of participants are needed. Unless the benefits outweigh their expectations, participants may stop participating in data collection. [3].

Research has investigated the factors that motivate users to participate in crowdsourcing communities [14]. Positive feedback and a sense of community identity are influential in increasing contributions and retention. Monetary compensation and flexible task schedules have been identified as primary motivators for workers in platforms like TaskRabbit and GigWalk [6]. Studies have shown that a small percentage of "power users" complete the majority of tasks [13]. Incentives in MCS can be categorized into three types: financial, entertainment, and social. Financial incentives, such as real money, are the most straightforward motivator but can raise concerns about quality control. Entertainment incentives can include enjoyment or game-based crowdsourcing, but may not be feasible for all tasks. Social incentives involve gaining reputation or recognition,

while non-monetary intrinsic factors like mental satisfaction and personal skill development can also motivate participation. Social psychological incentives, like social facilitation and ranking of contributions, have been utilized to increase contributions, but identity management and reputation measurement can be challenging in dynamic MCS systems. To effectively motivate individuals in MCS, a combination of above mention incentives should be used. [8, 10]

3.2 Task Management

The task design in MCS includes a framework established by the crowdsourcer to describe its work, which is composed of several components. Specific criteria to determine who can be engaged in the task and their respective evaluation and compensation policies may be laid down by the crowdsourcer. Three key factors for creating the task are its definition, UI (user interface) and granularity. [2]

Task definition: It is a critical component of MCS as it determines the goals, objectives, and data that will be collected in a crowdsensing campaign. There are several challenges related to task definition in MCS, it includes the complexity and heterogeneity of data sources, context awareness, and the trade-off between data quality and cost. These challenges can be overcome by participatory sensing techniques, user-centric task design, and task decomposition and delegation. In addition, it is important to engage the crowd in the process of task definition to ensure that tasks are engaging, relevant, and aligned with participants' interests and capabilities. [17]

User Interface: Similarly, good UI is important in MCS because it can impact worker productivity, task accuracy, and worker satisfaction. In general, MCS UI lacks context and the cognitive load required to complete tasks. Participants may have difficulty understanding the task requirements or may become fatigued from completing tasks, leading to decreased accuracy and participant satisfaction. [11]

Effective UI can be made with context-driven design, reducing cognitive load, providing real-time feedback and incorporating participants' preferences. Context-driven includes providing relevant information about the task and its purpose, which can help them to better understand the requirements and perform the task accurately. Reducing cognitive load

involves simplifying the task and its interface, such as breaking the task into smaller subtasks or using a conversational interface. Similarly, providing real-time feedback can help participants correct errors and improve their performance. [11]

Granularity: It allows more fine-grained approach allocation, which can improve the efficiency and effectiveness of MCS. A fine-grained approach to task allocation can ensure that tasks are assigned to the most suitable users based on their characteristics, such as location and historical behaviour. This approach can also enable the more complex and heterogeneous tasks to be completed, which can lead to more accurate and reliable data collection. However, if the task is too fine-grained, it can result in increased computational complexity and communication overhead. Spatiotemporal model, which predicts the probability of a mobile user completing a particular task based on their current location, movement patterns, and historical behaviour can be used to estimate the granularity. The model also considers the diversity and heterogeneity of the task and mobile users. [15]

3.3 Privacy and Security

The security and privacy concerns of MCS are much more complex and difficult compared with the past computing paradigms, because of several factors. First, the fact that people are involved in crowdsourcing sensing and intelligence means there could be sensitive information from mobile devices and the personal data of participants present in sensed data. Similarly, crowdsourcing raises security concerns, particularly when sensitive information is involved. It is more challenging to protect private information when the task gets distributed to a dynamic group of people, as the size of the group cannot be predetermined. Another problem is the constant change in the network topology of mobile users accepting crowdsourced tasks according to their location, interests or device conditions. The development and implementation of effective security and privacy solutions will become even more challenging as the topology changes with time, due to people's mobility and evolving user behaviour. [19]

4 Willingness-based Participant Selection

Article [7] proposes a participant selection strategy for crowdsensing that takes into account users' willingness to participate. The authors develop a theoretical framework that models user willingness as a function of various factors, such as *task type, task reward, and user experience,* and use this framework to design an algorithm for selecting participants that maximize the number of willing users and the quality of the collected data. The paper also presents an implementation of the proposed strategy in a real-world crowdsensing system and evaluates its effectiveness through both simulation experiments and real-world experiments. The results show that the proposed strategy can improve the efficiency and effectiveness of crowdsensing, by increasing the number of willing participants and improving the quality of the collected data. This model is based on a greedy algorithm that selects users with the highest predicted willingness score. The algorithm sorts the users in descending order of their willingness scores, and then iteratively selects the top-ranked users. In each iteration, the algorithm evaluates the marginal utility of adding a new user based on the expected contribution of the user to the overall quality of the collected data. If the expected contribution is high, the user is selected; otherwise, the algorithm moves on to the next user in the ranked list. The model also includes a reinforcement learning component that learns from the outcomes of past tasks and adjusts the willingness scores of the users based on their actual participation and contribution to the tasks. This helps to refine the predictions of user willingness over time and improve the accuracy of the participant selection algorithm. [7] The steps are listed below :

User-Specific Features: The willingness function takes as input a set of user-specific features, which include demographic information, location, and past participation history.

Feature Selection: The willingness function uses a correlation-based feature selection (CFS) algorithm to select the most relevant features for predicting user willingness. The CFS algorithm selects a subset of features that are highly correlated with the target variable (user willingness) while being minimally correlated with each other.

Logistic Regression Model: The willingness function uses a logistic regression model to estimate the probability of a user's willingness to participate in a crowdsensing campaign. The logistic regression model is

trained on the selected features using a supervised learning approach, where the model learns to predict the target variable (user willingness) based on the input features.

Willingness Score Calculation: Once the model is trained, the willingness function uses it to calculate a willingness score for each user. The willingness score is calculated by applying the logistic regression model to the user-specific features. The output of the model is a value between 0 and 1, which represents the user's probability of participating in a crowdsensing campaign.

Participant Selection: The willingness score is used to select the top N users with the highest scores for participation in the crowdsensing campaign. Users with higher willingness scores are more likely to participate in the campaign. [7]

5 User Characteristic Aware Participant Selection

MCS is a promising approach for collecting data from a large number of mobile devices. However, selecting appropriate participants for MCS is a challenging task, as the characteristics of potential participants can vary widely and can affect the quality and efficiency of the data collection. In [18], the authors propose a participant selection method for MCS that takes into account the user characteristics of potential participants, such as their mobility patterns, location, and availability. The proposed method aims to improve the quality of the data collected in MCS by selecting the most appropriate participants based on their characteristics. The authors first review the existing participant selection methods for MCS and highlight their limitations, such as the lack of consideration of user characteristics or the use of simple heuristics that do not capture the complexity of the MCS task. They then introduce the proposed method, which uses a scoring system to evaluate the suitability of potential participants for the MCS task. [18]

The scoring system takes into account three user characteristics, i.e. *mobility patterns, location,* and *availability*, and three task requirements, i.e. *accuracy, timeliness,* and *cost*. This uses a fuzzy inference system to convert the user characteristics and task requirements into fuzzy sets and then apply a set of fuzzy rules to calculate the score for each potential participant. The fuzzy rules are based on expert knowledge and are designed to capture the relationship between the user characteristics and

task requirements and the suitability of a potential participant for the MCS task. [18]

Similarly, it evaluates the proposed method using real-world datasets and compares it to other participant selection methods. In terms of the quality of the data collected and the effectiveness of the MCS task, the evaluation results show that the proposed method is superior to the other methods. The authors also conduct a sensitivity analysis to investigate the effect of the weightings of the user characteristics and task requirements on the participant selection process. The results show that the weightings can be adjusted to balance the trade-off between the quality of the data and the cost and timeliness of the MCS task. [18]

The paper discusses the implications of the proposed method for future research and practical applications. They suggest that the method can be extended to include other user characteristics and task requirements and that it can be integrated with other techniques, such as incentive mechanisms and task allocation strategies. They also note that the method can be applied in various domains, such as environmental monitoring, transportation, and healthcare, where MCS can provide valuable insights and services. [18]

6 Quality Aiming Participation Selection

Article [5] proposes a participant recruitment method for MCS that takes into account the quality of data collected. The authors argue that traditional recruitment methods for mobile crowdsensing often prioritize the number of participants over the quality of data, which can result in low-quality data. The proposed method, on the other hand, uses a two-stage recruitment process that involves pre-screening potential participants based on their reliability and experience with mobile sensing and then selecting the most suitable participants for the task at hand. [5]

The first stage of the recruitment process involves pre-screening potential participants based on their reliability and experience with mobile sensing. The paper proposes a *Reliability Index (RI)* that can be used to evaluate a participant's reliability based on their past performance in similar tasks. The RI is calculated based on three factors: *completion rate, accuracy rate,* and *response time.* Participants who meet a minimum RI score are then eligible to participate in the second stage of the recruitment process. [5]

The second stage of the recruitment process involves selecting the most suitable participants for the task at hand. The paper proposes a *Suitability Index (SI)* that can be used to evaluate a participant's suitability based on their characteristics and experience. The SI is calculated based on four factors: *age, gender, education level,* and *previous experience with the specific task*. Participants who meet a minimum SI score are then selected to participate in the mobile crowdsensing task. [5]

The paper argues that this two-stage recruitment process can improve the quality of data collected in mobile crowd sensing by selecting the most reliable and suitable participants for the task at hand. They also note that the proposed method can be used in a variety of applications, such as environmental monitoring, traffic monitoring, and health monitoring. In addition to the proposed method, the paper also discusses several challenges and limitations of mobile crowd sensing. One of the main challenges is ensuring the quality of data collected, as participants may not always provide accurate or reliable data. The authors suggest that the proposed method can help address this challenge by selecting more reliable and suitable participants. [5]

7 Discussion and Conclusion

This paper used recent research papers to summarise the MCS challenges and focused on three different participation selection methods. It was found that participation selection was one of the least researched topics in MCS. As three different participation selection methods were summarized, this helps to understand the overall participation selection model and its recent trend. As this is a summarization, it might not provide detailed insight into the model itself but it provides the platform to reach there.

To conclude, All three participant selection model has the common goal to improve the efficiency and effectiveness of MCS. It can be seen that traditional participant selection methods, such as random sampling or incentive-based approaches, may not always result in high-quality data or reliable participants. In all three research papers, it was noticed that they are combining multiple variables and algorithms to define the model. With things in common, they differ in their specific approaches and methodologies, such as whether to consider user characteristics, willingness to participate or the participant's skills, experience and reputation.

References

[1] Tariq Ali, Umar Draz, Sana Yasin, Javeria Noureen, Ahmad Shaf, and Munwar Zardari. An efficient participant's selection algorithm for crowdsensing. *International Journal of Advanced Computer Science and Applications*, 9, 01 2018.

[2] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, Mar 2013.

[3] Shanila Azhar, Shan Chang, Ye Liu, Yuting Tao, and Guohua Liu. Privacy-preserving and utility-aware participant selection for mobile crowd sensing. *Mobile Networks and Applications*, 27(1):290–302, 2020.

[4] Daniel Dimov. Crowdsensing: State of the art and privacy aspects, Oct 2020.

[5] Weijin Jiang, Junpeng Chen, Xiaoliang Liu, Yuehua Liu, and Sijian Lv. Participant recruitment method aiming at service quality in mobile crowd sensing. *Wireless Communications and Mobile Computing*, 2021:1–14, Apr 2021.

[6] Yonghang Jiang, Bingyi Liu, Ze Wang, and Xiaoquan Yi. Start from scratch: A crowdsourcing-based data fusion approach to support location-aware applications. *Sensors*, 19(20):4518, 2019.

[7] Wu Jiaying, Zhang Xiaoyu, Miao Xingxing, Chen Zhen, and Kang Wenshan. User willingness-based participant selection strategy of crowdsensing. *2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML)*, page 809–816, Aug 2022.

[8] Robert E. Kraut, Paul Resnick, and Sara Kiesler. *Building successful online communities evidence-based Social Design*. MIT Press, 2011.

[9] Hanshang Li, Ting Li, Weichao Wang, and Yu Wang. Dynamic participant selection for large-scale mobile crowd sensing. *IEEE Transactions on Mobile Computing*, 18(12):2842–2855, 2019.

[10] Yefeng Liu, Vili Lehdonvirta, Todorka Alexandrova, Ming Liu, and Tatsuo Nakajima. Engaging social medias: Case mobile crowdsourcing, Jan 2011.

[11] Panagiotis Mavridis, Owen Huang, Sihang Qiu, Ujwal Gadiraju, and Alessandro Bozzon. Chatterbox: Conversational interfaces for microtask crowdsourcing. *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*, Jun 2019.

[12] Umang Mehta, Parth Soni, and Jinan Fiaidhi. Mobile crowd sensing (mcs) [preprint]. 2020.

[13] Mohamed Musthag and Deepak Ganesan. Labor dynamics in a mobile micro-task market. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013.

[14] Jose Mauricio Nava Auza, Jose Roberto Boisson de Marca, and Glaucio Lima Siqueira. Design of a local information incentive mechanism for mobile crowdsensing. *Sensors*, 19(11):2532, 2019.

[15] Liang Wang, Zhiwen Yu, Daqing Zhang, Bin Guo, and Chi Harold Liu. Heterogeneous multi-task assignment in mobile crowdsensing using spatiotemporal correlation. *IEEE Transactions on Mobile Computing*, 18(1):84–97, 2019.

[16] Yufeng Wang, Xueyu Jia, Qun Jin, and Jianhua Ma. Mobile crowdsourcing: Framework, challenges, and solutions. *Concurrency and Computation: Practice and Experience*, 29(3), 2016.

[17] Zhibo Wang, Jiahui Hu, Qian Wang, Ruizhao Lv, Jian Wei, Honglong Chen, and Xiaoguang Niu. Task-bundling-based incentive for location-dependent mobile crowdsourcing. *IEEE Communications Magazine*, 57(2):54–59, Feb 2019.

[18] Dapeng Wu, Haopeng Li, and Ruyan Wang. User characteristic aware participant selection for mobile crowdsensing. *Recent Advances in Crowdsensing and Its Security, Privacy and Trust Challenges*, 18(11):3959, 2018.

[19] Kan Yang, Kuan Zhang, Ju Ren, and Xuemin Shen. Security and privacy in mobile crowdsourcing networks: Challenges and opportunities. *IEEE Communications Magazine*, 53(8):75–81, Aug 2015.

[20] Zhiwen Yu, Huadong Ma, Bin Guo, and Zheng Yang. Crowdsensing 2.0. *Communications of the ACM*, 64(11):76–80, 2021.

Succinct Non-Interactive Arguments

Shuto Kuriyama
shuto.kuriyama@aalto.fi

Tutor: Russell W. F. Lai

Abstract

Ever since succinct non-interactive argument of knowledge (SNARK) was introduced as a class of proof systems, its formalisation and many applications have been explored recently. SNARK is the extended version of one type of proof systems known as succinct non-interactive argument (SNARG). This paper explains several fundamental concepts required to comprehend SNARK including a variety of proof systems for $\mathcal{NP}$ language and shows a construction method of SNARG from another type of proof system known as probabilistic checkable proof (PCP). Proof systems are listed in historical order so that the relationship among them can be recognised easily.

KEYWORDS: SNARK, SNARGs, PCP, IP

1 Introduction

With the widespread use of internet, cryptographic protocols have played a fundamental role in securing communication between two parties or even within multiple parties. Due to the growing concern about privacy protection in the big data paradigm, proof systems (one of the classes of cryptographic protocol) has recently got even more attention for their ap-

plications in privacy preserving systems [8, 20, 13].

1.1 Proof Systems

Proof systems are in general consist of a prover and a verifier that are probabilistic Turing machines with common (shared) input and private input [9]. Given a mathematical statement as common input and a proof of the statement as private input, referred to as *witness*, the prover wants to convince the verifier that they knows a proof of the veracity of the statement. The verifier should be convinced with an overwhelming probability if the statement is actually true while he should not be convinced with more than a negligible probability if the statement is false. These properties are known as *completeness* and *soundness* respectively.

1.2 zk-SNARK

Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK) is one kind of proof systems that are widely applied in privacy preserving systems including decentralised money, smart contract, supply chain, and self-sovereign identity [6, 17, 11, 18, 14, 19]. Self-sovereign identity enables identification without revealing one's identity [19]. For example, an question "Are you over 20 years old" can be answered and verified without disclosing one's identity (one's age).

An argument system is a restricted proof system whose soundness holds only for a prover bounded by a polynomial time. An argument system is *succinct*, if its proof size is small, namely, bounded poly-logarithmically in the size of witness [12]. A non-interactive proof consists of a single non-interactive message that can be verified by any verifier [5]. Argument systems comprising these above features are known as *succinct non-interactive arguments* (SNARGs) [7, 16]. SNARK is SNARG with the extended soundness - *knowledge soundness* - that additionally guarantees the prover knows the witness of a statement if the verifier is convinced [4]. Lastly, a proof system (argument system) is *zero-knowledge* if the verifier does not learn any information except for the fact that the prover knows a statement is true [9].

Both SNARG and SNARK can be constructed from probabilistic checkable proof system (PCP) that is one type of proof systems. A verifier in PCP is probabilistic and it can verify a statement just by looking at a couple of bits of its proof without checking the entire proof.

1.3 The purpose of this paper

There are many articles regarding zk-SNARK and other relevant proof systems, however they often do not comprehensively define them with a consistent style and explain how each of the proof systems are related. This complicates learning proof systems especially for a new learner.

The aim of this paper is to showcase the development of proof systems in historical order starting from interactive proof systems to zk-SNARK along with their theoretical concepts. A computer science student who has no prior knowledge in proof systems should be able to gasp the main concepts in the field before they starts reading recent articles.

1.4 Paper Structure

This paper first defines interactive proof systems and PCP in the chapter 3-4. Next it describes SNARG and SNARK in the chapter 5. In the chapter 6-7, a construction method of SNARG with two steps by Micali and Kilian is explained.

2 Preliminary

A language class $\mathcal{NP}$ contains a collection of language L for which there exists a Turing machine (TM) V that takes an instance x and a witness w (a proof of the instance) as input and verifies the statement $x \in L$ in a polynomial time.

Definition 2.1 (Class $\mathcal{NP}$)**.** A language L is in the $\mathcal{NP}$ class if there exists a polynomial time TM V such that

$$L = \{x \mid \exists w,\ |w| = poly(|x|) \wedge V(x, w) = 1\}.$$

A collision-resistant hash function is a member of a function family for which it is hard to find two inputs that are mapped to the same output.

Definition 2.2 (Collision-resistant Hash Function Family)**.** A function family $\mathbb{H} = \bigcup_\lambda \mathcal{H}_\lambda$, where $\mathcal{H}_\lambda = \{h : \{0,1\}^{\ell_{in}(\lambda)} \to \{0,1\}^{\ell_{out}(\lambda)}\}$ and λ is a security parameter, is collision-resistant hash function (CRHF) family if it satisfies the following

- Efficiency: The functions ℓ_{in} and ℓ_{out} are bounded by a polynomial in λ. Given $x \in \{0,1\}^{\ell_{in}(\lambda)}$ and λ, $h(x)$ can be computed in a polynomial time

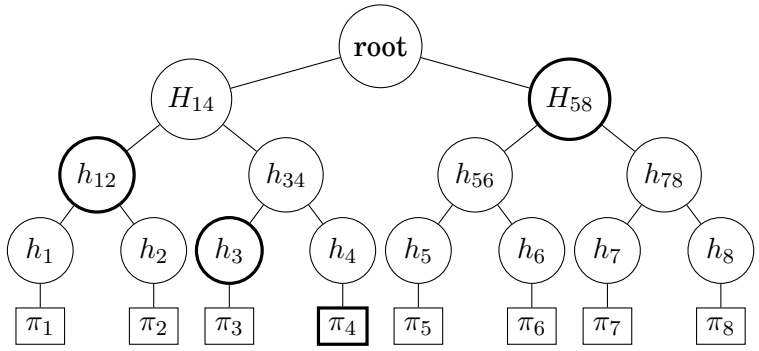


Figure 1. A Merkle tree with 8 leaf nodes for data $\pi = \pi_1||\pi_2||\cdots||\pi_8$ ($|\pi_i| = \frac{|\pi|}{8}$) and a CRHF h. $h_{ij} = h(\pi_i||\pi_j)$, $H_{ik} = h(h_{ij}||h_{j+1k})$, $i, j, k \in [8]$.

in λ.

- Compression: For all security parameter λ, $\ell_{in}(\lambda) > \ell_{out}(\lambda)$
- Collision resistance: For all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, it holds that

$$\Pr\left[\begin{array}{c|c} h(x) = h(x^{'}) & h \leftarrow\!\!\$\ \mathbb{H} \\ x \neq x^{'} & (x, x^{'}) \leftarrow \mathcal{A}(1^{\lambda}, h) \end{array}\right] = negl(\lambda)$$

2.1 Merkle Tree

A Merkle tree is a binary tree whose leaf nodes are labelled with of a data block and other nodes are labelled with the hash of their child nodes (See Figure 1). The structure enables efficient and secure verification of large amounts of data. By the nature of a hash function (compression), the size of the root of a Merkle tree is smaler than the data blocks that is the concatenation of the leaf nodes. Given a Merkle tree, verifying that a leaf node (data block) is a part of the tree only requires computing a number of hashes proportinal to the logarithm of the number of leaf nodes hash values. The list of nodes is also known as the authentication path of the leaf node. In the Figure 1, for example, the authentication path for the data block π_4 is the list of the hash with the bold outline $l = [h_3, h_{12}, H_{58}]$, and $|l| = \log_2 8 = 3$. The root can be computed via the CRHF h, π_4, and l. It can be verified that π_4 is in the given tree if the computed root matches the original root of the given tree.

3 Interactive Proof System

An interactive proof system comprises two parties: a prover and a verifier. Both of them have a instance x and a mathmatical statement consisting of a tuple (x, L) as input, where L is a language. The statement is described as "$x \in L$". The prover is additionally given the proof of the statement, known as "witness" w. The verifier may or may not know about the witness. In the setting, the prover tries to convince the verifier that the prover knows whether the statement is *true* or *false*. More formally, a language L is defined with regards to a relation R.

Definition 3.1 (Relation)**.** A relation R is a collection of tuples (x, w) that is the sepecification of the relationship between a instance x and a witness w.

Definition 3.2 (Language)**.** A language L_R is a set of instances that appear in a relation R

$$L_R := \{x \mid \exists w,\ (x, w) \in R\}.$$

An interactive proof system for a language L is a interactive protocol (P, V), where a prover P is a probabilistic Turing machine (TM) with unbound computation time and a verifier V is a probablistic polynomial-time (PPT) TM whose computation is bounded by a polynomial time in its input [9]. Both of them have an internal random tape containing infinite sequences of random bits and access to it. The operation of reading a random bit is known as *cointoss*. Random bits read by P and V are kept private inside each machine. Such protocol is so called a private coin protocol.

Definition 3.3 (Interactive Proof (IP))**.** An interactive proof system for a language $L \in \{0, 1\}^*$ is an interactive protocol (P, V) (for communication between a prover P and a verifier V) that satisfies the followings

- Completeness: If $x \in L$, then P should be able to convince V with overwhelming probability (the probability is taken over the *cointoss* of P and V).
  $\forall\, k > 0,\ \exists\, N,\ \forall x \in L,$

$$|x| > N \rightarrow \Pr[V \textit{ accepts in } \langle P, V\rangle(x)] > 1 - \frac{1}{|x|^k}$$

- Soundness: If $x \notin L$, any prover $\tilde{P}$ can convince V with only negligible probability (the probability is taken over the *cointoss* of P and V).
  $\forall \tilde{P} \, \forall \, k > 0, \, \exists \, N, \, \forall x \in L,$

$$|x| > N \rightarrow \Pr[V \; accepts \; in \; \langle P, V \rangle(x)] > \frac{1}{|x|^k}.$$

The class of language that have an interactive proof system is defined as IP.

Definition 3.4 (Class IP)**.**

$$IP := \{L \mid L \; has \; an \; interactive \; proof\}$$

Babai defined a similar language class AM (Arthur–Merlin) that has an interactive proof system where random bits read by a verifier is public, known as public coin protocol [3]. In other words, the *cointoss* of verifier V is visible to a prover P. Any language that has interactive proofs with private coin also has interactive proofs with public coins [10].

4 Probablistically Checkable Proof (PCP)

Probablisitically Checkable Proof (PCP) was introduced as another complexity class that gives a novel representation of $\mathcal{NP}$. A language in the $\mathcal{NP}$ can be verified in a constant time with PCP by looking at a couple of bits in the proof without checking the entire proof.[1, 2].s

Definition 4.1 (PCP)**.** A PCP system for a language L is a PPT Turing machine M that uses public coins (internal random strings) with input x and the proof string Π (an array of bits), satisfying

- Completeness: If $x \in L, \; \exists \, \Pi$ such that $\Pr[M^{\Pi}(x) = 1] = 1$
- Soundness: If $x \notin L, \; \forall \, \Pi \quad \Pr[M^{\Pi}(x) = 1] < \frac{1}{2}$.

M can be interpreted as a verifier, and it has random access to each bit of proof string Π. A *query* is the operation of reading a bit of Π. The operation that access to an internal random bit is referred to as *coin toss*. A restricted version of PCP was considered for the sake of exploring the trade-offs of among the running time of the verifier and the size of internal random bits and query bits [1].

Definition 4.2 (PCP$(r(\cdot), q(\cdot))$)**.** Let $r : \mathbb{N} \to \mathbb{N},\ q : \mathbb{N} \to \mathbb{N}$. A PCP$(r(\cdot), q(\cdot))$ system for a language L is a PCP M defined above that additionally satisfies

- Romdomness Complexity: On input x, M can make at most $\mathcal{O}(r(|x|))$ coin tosses.
- Query Complexity: On input x, M can query at most $\mathcal{O}(q(|x|))$ times.

5 SNARGs Definition

Kilian introduced a proof size efficient interactive argument system based on PCP: that is *succinct interactive argument systems* [12]. The prover in the argument system uses a Merkle tree that is a binary tree whose nodes are labelled as the hash of their children. It enables the verifier to have access to a virtual PCP [15]. Subsequently, Micali showed the method to transform it to non-interactive arguments systems, known as succinct non-interactive argument systems (SNARGs), by applying Fiat-Shamir heuristic. [7, 16]. Fiat-Shamir heuristic transforms public coin interactive argument systems into non-interactive argument system by simulating the coin toss of the verifier by a random oracle.

A SNARG protocol consists of an argument system expressed as a triple of TMs (G, P, V). G generates a common input crs and a private input of the verifier vrs. They are referred to as *common reference string* (CRS) and *verification state* respectively. P takes the crs, the instance x, and the witness w as input and outputs the proof π. V takes the vrs, the statement x, and the proof π and outputs 1 or 0 corresponding to accepts/rejects.

Definition 5.1 (SNARG)**.** A SNARG is an argument systems $\Pi = (G, P, V)$ for a relation R, where

- (crs, vrs) $\leftarrow$ G(1^λ) $\quad \lambda$ is a security parameter.
- $\pi \leftarrow$ P(crs, x, w)
- $b \leftarrow$ V(crs, x, π) $\quad$ If $b = 1$, the verifier is convinced. $b = 0$ otherwise.

and satisfies the following properties

- Succinctness: Both cocommunication complexity between P and V de-

noted by $|\langle P, V\rangle|$ and the running time of V are bound by poly($\lambda, o(|x|)$). In non-interactive setting, $|\langle P, V\rangle| = |\pi| = o(poly(\lambda, |x|))$

- Completeness: If $x \in L_R$, then P should be able to convince V with overwhelming probability
  $\forall\, \lambda \in \mathbb{N},\ \forall (x, w) \in R$

$$\Pr\left[V(\mathbf{crs}, x, \pi) = 1 \,\middle|\, \begin{array}{c} \mathbf{crs} \leftarrow G(1^\lambda) \\ \pi \leftarrow P(\mathbf{crs}, x, w) \end{array}\right] = 1 - negl(\lambda)$$

- Soundness: If $x \notin L_R$, any adversary $\mathcal{A}$ (including both honest provers and malicious provers) can convince V with only negligible probability
  $\forall\, \lambda \in \mathbb{N},\ \forall x \notin L_R,\ \forall PPT\ \mathcal{A}$

$$\Pr\left[V(\mathbf{crs}, x, \pi^{'}) = 1 \,\middle|\, \begin{array}{c} \mathbf{crs} \leftarrow G(1^\lambda) \\ \pi^{'} \leftarrow \mathcal{A}(\mathbf{crs}, x) \end{array}\right] = negl(\lambda).$$

Succinct non-interactive argument of knowledge (SNARK) is an argument system satisfying above with an additionally stronger soundness - *knowledge soundness* - instead of soundness. Knowledge soundness guarantees that, if the verifier accepts a proof, the prover can compute the witness w corresponding to the instance x for the proof.

Definition 5.2 (Knowledge Soundness)**.** An argument system $\Pi = (G, P, V, \mathcal{E}_\mathcal{A})$ for a relation R is knowledge sound if there exists a public machine (extractor) that can compute the witness w for the instance x. Formally, let $\mathcal{A}$ be an adversary and $\mathcal{E}_\mathcal{A}$ a PPT TM.
$\forall\, \lambda \in \mathbb{N},\ \forall\ PPT\ \mathcal{A}\ \ \exists\ PPT\ \mathcal{E}_\mathcal{A}$

$$\Pr\left[\begin{array}{c} V(\mathbf{crs}, x, \pi^{'}) = 1 \\ \wedge\ (x, w) \notin R \end{array} \,\middle|\, \begin{array}{c} \mathbf{crs} \leftarrow G(1^\lambda) \\ (\pi^{'}; w) \leftarrow \mathcal{A}||\mathcal{E}_\mathcal{A}(\mathbf{crs}, x) \end{array}\right] = negl(\lambda)$$

Under the above definition, soundness obviously holds as $x \notin L_R \rightarrow (x, w) \notin R$.

Definition 5.3 (SNARK)**.** A SNARK is an argument system $\Pi = (G, P, V, \mathcal{E}_\mathcal{A})$ satisfies the aforementioned properties of SNARG as well as knowledge soundness defined above.

6 Succinct Interactive Argument System from PCP (Kilian)

PCP (explained in the chapter 3) is verifieable with polylogarithmic time in the size of the proof for a deterministic polynomial TM. Kilian applied PCP to construct interactive argument systems with succinctness enabling the size of the proof to be polylogarithmic in that of $\mathcal{NP}$ proof. In the construction, the prover uses a Merkle tree to shorten the proof string of PCP and provide the access to the proof string (indirect access). The prover is provided a collision resistant hash function (CRHF) and compute the root of the Merkle tree setting its leaves as the split of the proof string. The interactions between the prover P and the verifier V are defined as following.

1. The verifier V sends a CRHF h to the prover
2. The prover P generates the proof string π of PCP given a witness and instance. Afterwards, P compiles π into the root rt of a Merkle tree using the CRHF h and sends it to V.
   $\pi \leftarrow P(x, w)$
   rt $\leftarrow$ Merkle(π)
3. V tosses coin a polynomial number of times and generate series of random bits $r_1, r_2, ..., r_n$ and sends them to P. Both P and V internanlly computes the PCP queries $q_1, q_2, ..., q_m$ depending on the instance x, rt, and the series of random bits.
4. P sends the corresponding bits to those queries as well as their proofs: that enable V to confirm the those answers are consistent with the root rt. The proofs, known as authentication paths, are the values of the minimum number of nodes enables V to re-compute the root of the tree and confirm the result is the same as rt.

7 SNARG from Succinct Interactive Argument System (Micali)

Micali applied Fiat-Shamir heuristic to Kilian's construction to make it non-interactive. In Kilian's construction, the verifier V sends a CRHF h to the prover and generates a sequence of random bits $r_1, r_2, ..., r_n$ and sent it to the prover P. By applying Fiat-Shamir heuristic, the prover has access to the CRHF h and another hash function h' simulated by random

oracles. The prover computes the root of Merkle tree rt using h and a series of queries $r_1 = h'(\mathbf{rt}), r_2 = h'(\mathbf{rt}, r_1), ..., r_n = h'(\mathbf{rt}, r_1, r_2, ..., r_{n-1})$.

8 Conclusion

The paper explained theoretical concepts of various proof systems that are underlying concepts of (zk-)SNARK including interactive proof, PCP, SNARG. A construction method of SNARG from PCP was achieved by applying Fiat-Shamir heuristic (random oracles) to succinct interactive arguments constructed by Kilian's method using a Merkle tree. The construction method of (zk-)SNARK from PCP as well as its example could be further described.

References

[1] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[2] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32, 1991.

[3] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[4] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349, 2012.

[5] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. 2019.

[6] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[7] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86: Proceedings 6*, pages 186–194. Springer, 1987.

[8] Uriel Fiege, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 210–217, 1987.

[9] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[10] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68, 1986.

[11] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, 4:220, 2016.

[12] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[13] Wanxin Li, Hao Guo, Mark Nejad, and Chien-Chung Shen. Privacy-preserving traffic management: A blockchain and zero-knowledge proof inspired approach. *IEEE access*, 8:181733–181743, 2020.

[14] Mediledger. Mediledger 2019 progress report. https://assets.chronicled.com, 2019. Accessed: 2023-04-12.

[15] Ralph Charles Merkle. *Secrecy, authentication, and public key systems*. Stanford university, 1979.

[16] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[17] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[18] Nick Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.

[19] Andrew Tobin and Drummond Reed. The inevitable rise of self-sovereign identity. *The Sovrin Foundation*, 29(2016):18, 2016.

[20] Jiannan Wei, Tran Viet Xuan Phuong, and Guomin Yang. An efficient privacy preserving message authentication scheme for internet-of-things. *IEEE Transactions on Industrial Informatics*, 17(1):617–626, 2020.

Migration from Monolithic Architecture to Microservices: Challenges and Opportunities

Shweta Jaiswal
shweta.jaiswal@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

The shift from monolithic to microservices architecture has become popular among software development teams due to its potential to improve scalability, flexibility, and maintainability. However, migrating from monolithic to microservices architecture can present various challenges and opportunities for organizations. This paper investigates the challenges organizations face during the migration process. We explore the technical, infrastructural, security, and organizational challenges of migrating to microservices architecture and provide insights into the best practices and strategies for overcoming these challenges. Additionally, we highlight the opportunities that microservices architecture can offer for increased agility, scalability, and employability as compared to monolithic architecture. Overall, this paper provides valuable insights for organizations considering migrating from monolithic to microservices architecture.

KEYWORDS: monolithic, microservices, migration challenges.

1 Introduction

In recent years, microservices as an architecture have advanced to a point where every company wants to migrate towards it. The primary reason

for shifting to microservices from monolithic architecture is the benefits provided by microservices, such as increased flexibility, scalability, and resilience.

"Micro web services" was first introduced in 2005 by Dr. Peter Rogers [12]. According to him, a micro web services architecture is a well-organized platform that applies the principles of REST and web services, along with Unix-like pipelines and scheduling. It is designed to simplify service-oriented architectures and enhance flexibility. The term "microservices" was first coined at a software architecture conference in 2011 to describe a type of architecture that many attendees were testing at the time [9]. Among the early adopters of microservices were prominent tech companies such as Amazon and Netflix.

A monolithic architecture combines all business logic in a single computing network with a single source code. This application requires changing the complete network and tech stack, which involves modifying the code base and updating and deploying the service-side interface. As a result, updates become complicated and exhausting [15].

Multinational companies face issues with massive legacy systems based on the monolithic architecture that has grown over time. Such complex systems become challenging to upgrade and maintain. The solution to this problem is microservices, which divide complicated activities into smaller processes that operate independently and can be upgraded individually [15].

The microservices style of architecture creates sophisticated applications from small, standalone apps that communicate using APIs (Application Programming Interface(s)), which are independent of language. The approach of structuring an application as a group of services (standalone apps) that can be deployed separately and have minimal interdependencies is commonly known as microservices architecture. These services are typically aligned with specific business functionalities and managed by a dedicated team. Due to these features of microservices, the code is highly maintainable and testable [16].

This paper examines the critical differences between the two architectural styles and explores the challenges of migrating to microservices. By evaluating the benefits of microservices and challenges in migration, this paper aims to provide a compelling argument for why organizations should consider using microservices over monolithic architecture for their next project.

This paper is organized as follows. Section 2 presents the background literature on monolithic and microservices architecture. Section 3 discusses the main differences between these two architectures. Section 4 reviews the challenges of migration to microservices from monolithic architecture. Section 5 provides a consolidated discussion. Finally, Section 6 presents some concluding remarks.

2 Background

This section contextualizes microservices and monolithic architectures in terms of existing literature and compares their performance.

Multiple studies have been conducted on these two architectures in different environments based on various parameters. Gos and Zabierowski [6] compared a web application's performance in Java, concluding that microservices architecture is more efficient in handling a larger number of requests than monolithic architecture. The code is of high quality, easy to scale, more reliable, and convenient to maintain. Another experiment [17] was done in the cloud using the Play web framework, which showed that one of the major benefits of microservices is developing, scaling, deploying, monitoring, and operating an application independently.

Al-Debagy and Martinek [1] give a detailed comparison between microservices and monolithic architecture. They compare these two architectures based on different parameters, such as concurrency testing, load testing, and discovery testing. Their results exhibit that microservices architecture performs better in concurrency testing and discovery testing in terms of throughput, whereas the results did not present a significant difference in load testing.

In another research [3], the performance and scalability of microservices and monolithic architecture were compared with two different implementation technologies (Java vs. C#.NET). The experiments were performed on different environments (local, Azure App Service, and Azure Spring Cloud). The results showed that although monolithic architectures scale better on a single machine, microservices outperform them in distributed systems and are more cost-efficient.

Almost all of the existing literature implies that microservices are a better architecture than monolithic ones. However, significant issues arise when migrating from monolithic to microservices, especially in legacy systems. The cost, deployability, and team organization are a few factors that

can affect migration [2].

Although companies are migrating from monolithic in-house facilities to cloud microservices, there is a lack of understanding and expertise in adopting microservices and the tools and techniques to use for the migration. It has been shown that different techniques should be followed for different aspects of migration: orchestration, storage, and deployment settings [14].

At present, organizations need to scale in an adaptable way that impacts their production and meets their business needs. This involves using containers with microservices. Research [15] shows that using microservices with some orchestration can improve overall performance tremendously, including response, development, and deployment time.

Many enterprises often demand tailored software products that meet their particular requirements before deploying them on-premises. Adapting such products becomes more straightforward if the migration is towards Cloud-native SaaS that employs microservices, thereby capitalizing on the Cloud and multi-tenancy's economies of scale [7].

In conclusion, the existing literature suggests that microservices architecture is a better option than monolithic architecture for various reasons, including better scalability, reliability, and ease of maintenance. However, migrating from a monolithic system to a microservices architecture can be challenging and costly, particularly for legacy systems. Organizations looking to adopt microservices should carefully consider the various factors that can impact their migration, such as team organization, cost, deployability, and the appropriate tools and techniques for different aspects of the migration. Overall, microservices architecture, when properly implemented, can provide several benefits to organizations, particularly in terms of agility and scalability.

The upcoming section further explores the architectural differences between microservices and monoliths in the context of migration.

3 Architectural Differences between Monolithic and Microservices

When migrating from a monolithic architecture to microservices, it is essential to consider the differences between these two architectures. Table 1 highlights the differences between monolithic and microservices architecture w.r.t. different parameters. Microservices architecture enables independent development, easy scalability, higher consistency, and avail-

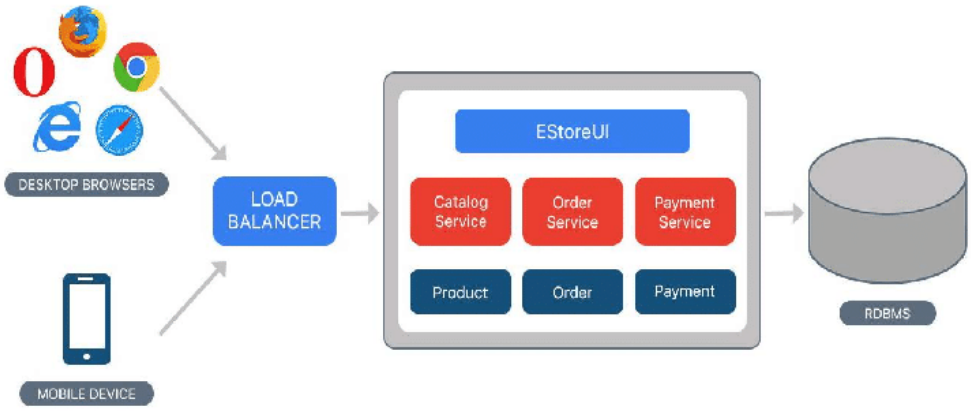


Figure 1. Monolithic Architecture (Source: Figure 1 from [6])

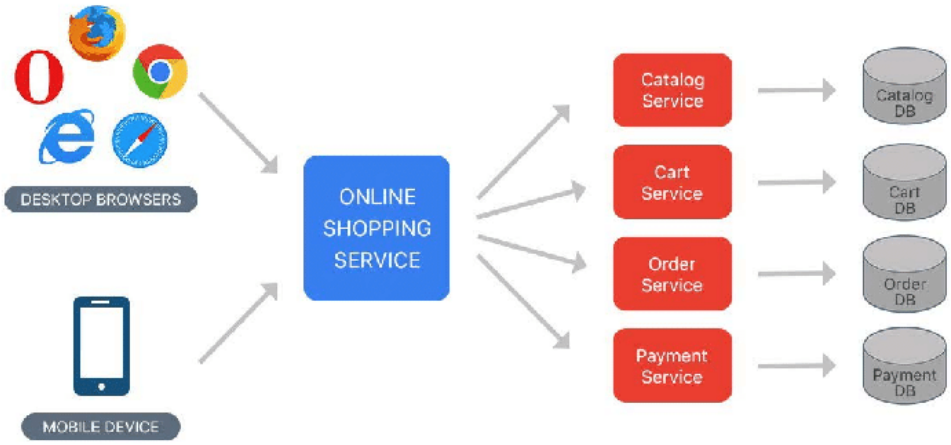


Figure 2. Microservices based Architecture (Source: Figure 2 from [6])

ability than monolithic architecture. Figure 1 presents the monolithic architecture, and Figure 2 shows the microservices-based architecture.

Monolithic architecture is built as a single extensive system with one code base, while microservices architecture is built as several small independent modules that communicate with each other. The modular nature of microservices makes them easily scalable, and each module can have its own database. This architecture also significantly reduces the overall build and development time compared to monolithic architecture.

4 Challenges in Migration: Monolithic to Microservices

The conventional monolithic architecture is inadequate for fulfilling the rapidly increasing demands of modern organizations. Due to the achieve-

Parameters	***Monolithic***	***Microservices***
Programming Language	Completely developed in a single programming language.	Each service can be developed independently using different programming languages.
Codebase	One codebase for the entire application	Each service has its separate codebase.
Understandability	It is complicated to understand and confusing.	It has high understandability and is very easy to maintain.
Application Scaling	It is complicated as it requires scaling the entire application.	Scaling is easier as each service can be scaled independently without scaling the entire application.
Development and deployment	Continuous development and deployment are complex.	Continuous development and deployment are smooth.
Service Startup	It takes time to build and start the service.	Building and starting up service is quick.
Data Model	It has a centralized data model.	It employs a federated data model, enabling each service to implement its unique data model.
Consistency and Availability	Consistency and availability are relatively low as any update to the application requires developing the complete application from scratch.	Highly consistent and readily available.

Table 1. Difference between Monolithic and Microservices Architecture

ments of some prominent enterprises in creating and implementing services using a microservices architecture, other organizations have a significant incentive to explore adopting this approach. Nevertheless, migrating to this architecture is a complex undertaking. In numerous ex-

isting systems, the components are tightly interconnected, with multiple dependencies that make it extremely challenging to separate them sensibly [11].

One of the most significant challenges with monolithic architecture is its tight coupling, which makes it extremely difficult to change the technology or framework of the code [6]. Organizations can overcome the challenge related to tight coupling and improve the flexibility of their system by migrating to microservices architecture. However, the team should assess the efforts and potential challenges before migrating and carefully plan and execute the process to ensure a successful transition.

A stepwise process could be adopted, which can involve migrating from a monolith to a modular intermediate state and then to microservices [4]. It is observed that significant effort is involved in migrating to a modular state. The modular state can be a transitional step in migrating to a microservices architecture. This provides a more agile software development paradigm while allowing for a stepwise approach to address the challenges of a complete migration to a microservices-based architecture.

The amount of work required for the migration is determined by the number of connections among the domain entities of the new modules, as well as the number of interfaces that exist between the modules [4]. The various challenges associated with migration to microservices predominantly occur because all enterprise-level applications start as monolithic ones. The subsequent section presents a detailed discussion of these challenges.

4.1 Technical Challenges

This section discusses the technical challenges associated with the implementation of microservices. The challenges discussed include modularization, integration testing, deployment, and transaction management.

Modularization: Refactoring the services from a monolithic architecture to make them modular is an extensive process. It may take a long time, especially if it is difficult to find a starting point from where the modularization of the application can take place. This can result in additional operational and infrastructural burdens, such as managing configurations, deployment, integration, security, provisioning, and monitoring [13]. One viable approach to mitigating modularization's complexities is containerization and provisioning [10]. This approach can effectively manage and reduce challenges related to modularization.

Integration Testing: Integrating several microservices together presents difficulty, primarily if the services are written in different languages. These services are asynchronous, making them challenging to do integration testing. Furthermore, the test engineer needs to understand all the services in-depth to cover all the test scenarios [18]. To overcome testing difficulties, it is recommended to utilize diverse testing techniques and tools, such as automated continuous integration, and follow standard agile methodologies. If the automation test coverage is substantial, it helps release and validate the microservices frequently [8].

Deployment: Deploying an application based on microservices can present challenges. Unlike a monolithic application typically deployed on one host/machine and scaled vertically, microservices-based applications offer various deployment patterns, such as multiple service instances per host, one service instance per host, or one service instance per host container [10]. Running multiple services and managing versions can lead to increased operational overhead. It is crucial to have an infrastructure management strategy and regularly implement automated and proactive infrastructure monitoring [18].

Transaction management: In a monolithic architecture, transaction management is uncomplicated as it involves in-process calls. However, in a microservices architecture, transaction management is more complex. Each microservice might maintain its own private database, and since microservices are stateless, several remote invocations may be necessary to fulfill a request. One solution to this challenge is implementing event-based programming, triggering events, and notifying relevant microservices of state changes [13].

4.2 Infrastructure Challenges

One of the key challenges in transitioning to microservices-based architecture is in understanding and handling the infrastructure for new services.

Performance: A microservices-based application may experience slower execution due to increased resource usage from a large number of APIs. This can be overcome by increasing the infrastructural resources, such as additional servers, or increasing the capacity of the current servers [10].

Database Migration: A monolithic service usually has a single database, whereas a microservice has multiple databases, usually one for each service. Breaking down a monolithic data model into independent data models for each microservice can be challenging, mainly due to the require-

ment of preserving data and transactional consistency. Database Migration is one of the open challenges during migration to microservices architecture [13].

4.3 Security Challenges

The migration from a monolithic architecture to microservices presents various security challenges that must be considered. With the use of several modular services in microservices, the attack surface is potentially more extensive than in monolithic architecture, increasing the likelihood of potential attacks. Additionally, even if microservices reduce the surface area for attack compared to monoliths, they increase the number of authentication steps, increasing authentication latency and impacting user experience. Furthermore, the increased traffic between services at a more granular level can make it difficult to automatically enable application observability for metrics, logs, and tracing, which is essential for both operational health and security posture tracking [13]. In this context, implementing the API Gateway pattern effectively enhances the security of microservice access by creating an abstraction layer that shields the underlying microservices from external clients. Other strategies, such as SSL, OAuth, and containerization, can also be implemented to improve the security of microservices-based applications.

Possible security challenges during migration from a monolithic architecture to microservices include the following:

Data breaches: With microservices, multiple entry points exist into the system, making it easier for attackers to exploit vulnerabilities and gain access to sensitive data [5].

Distributed denial of service (DDoS) attacks: Microservices can make it easier for attackers to launch DDoS attacks by targeting specific services and overwhelming them with traffic [5].

Authentication and authorization challenges: Microservices can increase the number of authentication steps, leading to longer authentication latencies. This can be a particular challenge for organizations that require real-time access to data [5].

Service discovery: In a microservices architecture, it is typical to have many services communicating with one another. This can make it challenging to track and manage service dependencies and vulnerabilities [13].

Containerization security: Containerization is a common approach

to deploying microservices but introduces new security challenges. Containers can be compromised if they are not correctly configured or secured, and containerized applications can be vulnerable to attacks that exploit container-specific vulnerabilities [5].

Compliance and regulatory challenges: Microservices can make it more difficult to comply with regulatory requirements, mainly if sensitive data is being processed and stored across multiple services. Organizations may need to take additional steps to ensure they meet compliance requirements.

Security testing challenges: With a large number of services in a microservices architecture, testing can become more complex and time-consuming. It can be challenging to test each service individually, and it can be difficult to simulate real-world attack scenarios [13].

These are just a few examples of the security challenges that organizations may face during migration from a monolithic architecture to microservices. Each organization will have its own unique set of challenges and must develop its own strategies for addressing them.

4.4 Organizational Challenges

Migrating from a monolithic to a microservices architecture is accompanied by several organizational challenges, including the lack of necessary skills within the team. Another challenge is the impact on the team's culture, as successful implementation requires effective communication among team members and stakeholders. According to Conway's law, a system's structure reflects the communication structure within an organization. Consequently, organizations that lack effective communication structures may develop systems with architectures that impede development cycles due to the delays involved during project transfers [8].

5 Discussion

Before transitioning from a monolithic architecture to microservices, an organization should carefully assess the costs and benefits of the transition. Although microservices can often facilitate faster application development, continuous integration, and quicker releases, it is crucial to consider that the challenges of transitioning may outweigh the potential benefits for some organizations.

This paper has explored the challenges and opportunities of migrating from a monolithic architecture to microservices. Through the literature review, we have found that microservices offer many benefits, such as scalability, flexibility, and agility, which can help organizations achieve their business objectives. However, this migration process comes with several challenges, including technical, infrastructural, security, and organizational challenges.

The technical challenges organizations might face during migration were discussed in this study. These challenges include breaking down the monolith into smaller, more manageable services, ensuring service compatibility and communication, and maintaining data consistency. One solution to these challenges is the use of containerization, which can provide better isolation and enable easier deployment and scaling. In the infrastructural challenges, we discuss the impact of resources on performance, while database migration remains an open challenge.

Furthermore, the security challenges related to migration were highlighted. Microservices increase the surface area for attacks, making them more prone to security threats. The implementation of SSL, OAuth, and containerization can help mitigate these risks, along with using an API Gateway pattern to secure access to the microservices.

One of the organizational challenges discussed is the unavailability of certain skills in the team. Migrating from monolithic to microservices also changes the culture of the team. For microservices architecture to be successful, communication among stakeholders and developers is crucial. This requires a shift in the organization's culture, which can be daunting.

Overall, migrating from a monolithic architecture to microservices offers several opportunities for organizations, including increased scalability, flexibility, and agility. The use of containerization, API Gateway pattern, and other security measures can help organizations maintain the security of their microservices-based applications. However, this process requires careful planning and execution to mitigate the challenges discussed. Organizations must ensure they have the necessary skills and culture in place to transition to microservices architecture successfully.

6 Conclusion

In conclusion, transitioning from a monolithic architecture to a microservices architecture can be challenging. However, it also presents several

opportunities for organizations to improve their software development practices. While the benefits of microservices are numerous, the challenges of transitioning cannot be ignored, especially the technical, infrastructural, security, and organizational challenges. Therefore, through careful planning and execution, organizations can overcome these challenges and reap the benefits of microservices architecture. The use of containerization, security measures, and cultural change management can help organizations make a successful transition. As software development practices continue to evolve, it is essential for organizations to consider microservices architecture as a viable option for achieving their business objectives.

References

[1] Omar Al-Debagy and Peter Martinek. A comparative review of microservices and monolithic architectures. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 000149–000154. IEEE, 2018.

[2] Florian Auer, Valentina Lenarduzzi, Michael Felderer, and Davide Taibi. From monolithic systems to microservices: An assessment framework. *Information and Software Technology*, 137:106600, 2021.

[3] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 10:20357–20374, 2022.

[4] Diogo Faustino, Nuno Gonçalves, Manuel Portela, and António Rito Silva. Stepwise migration of a monolith to a microservices architecture: Performance and migration effort evaluation. *arXiv preprint arXiv:2201.07226*, 2022.

[5] Hemanth Gopal, Guanqun Song, and Ting Zhu. Security, privacy and challenges in microservices architecture and cloud computing-survey. *arXiv preprint arXiv:2212.14422*, 2022.

[6] Konrad Gos and Wojciech Zabierowski. The comparison of microservice and monolithic architecture. In *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pages 150–153. IEEE, 2020.

[7] Sindre Grønstøl Haugeland, Phu H Nguyen, Hui Song, and Franck Chauvel. Migrating monoliths to microservices-based customizable multi-tenant cloud-native apps. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 170–177. IEEE, 2021.

[8] Miika Kalske, Niko Mäkitalo, and Tommi Mikkonen. Challenges when moving from monolith to microservice architecture. In *Current Trends in Web Engineering: ICWE 2017 International Workshops, Liquid Multi-Device*

Software and EnWoT, practi-O-web, NLPIT, SoWeMine, Rome, Italy, June 5-8, 2017, Revised Selected Papers 17, pages 32–47. Springer, 2018.

[9] James Lewis and Martin Fowler. Microservices: a definition of this new architectural term. *MartinFowler. com*, 25(14-26):12, 2014.

[10] Guozhi Liu, Bi Huang, Zhihong Liang, Minmin Qin, Hua Zhou, and Zhang Li. Microservices: architecture, container, and challenges. In *2020 IEEE 20th international conference on software quality, reliability and security companion (QRS-C)*, pages 629–635. IEEE, 2020.

[11] Francisco Ponce, Gastón Márquez, and Hernán Astudillo. Migrating from monolithic architecture to microservices: A rapid review. In *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–7. IEEE, 2019.

[12] Peter Rodgers. Service-oriented development on netkernel-patterns processes & products to reduce system complexity web services edge 2005 east: Cs-3". *CloudComputingExpo*, 2005.

[13] Mehmet Söylemez, Bedir Tekinerdogan, and Ayça Kolukısa Tarhan. Challenges and solution directions of microservice architectures: A systematic literature review. *Applied Sciences*, 12(11):5507, 2022.

[14] Davide Taibi, Valentina Lenarduzzi, and Claus Pahl. Continuous architecting with microservices and devops: A systematic mapping study. In *Cloud Computing and Services Science: 8th International Conference, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018, Revised Selected Papers 8*, pages 126–151. Springer, 2019.

[15] Freddy Tapia, Miguel Ángel Mora, Walter Fuertes, Hernán Aules, Edwin Flores, and Theofilos Toulkeridis. From monolithic systems to microservices: A comparative study of performance. *Applied sciences*, 10(17):5797, 2020.

[16] Markos Viggiato, Ricardo Terra, Henrique Rocha, Marco Tulio Valente, and Eduardo Figueiredo. Microservices in practice: A survey study. *arXiv preprint arXiv:1808.04836*, 2018.

[17] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *2015 10th Computing Colombian Conference (10CCC)*, pages 583–590. IEEE, 2015.

[18] Muhammad Waseem, Peng Liang, Mojtaba Shahin, Amleto Di Salle, and Gastón Márquez. Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software*, 182:111061, 2021.

Kubernetes Approach to Public Key Infrastructure

Huong Pham Thi Song
songhuong.phamthi@aalto.fi

Tutor: Matti Siekkinen

Abstract

A Kubernetes cluster has control plane and worker nodes, each of which has many components. In order to increase the availability of the cluster, the control plane could span over multiple nodes, spreading its components to different physical machines. Additionally, the api-server which is core component of a Kubernetes cluster is also exposed to outside of the cluster. In order to secure the communication between components inside control plane and communication from worker nodes to api-server, Kubernetes requires internal Public Key Infrastructure PKI. This paper explores the details of internal Kubernetes PKI.

KEYWORDS: Kubernetes, Cloud Computing, Public Key Infrastructure

1 Introduction

One of the reasons for Kubernetes success is its ability to be highly scalable. In order to achieve this, communication between components needs a secure channel and api-server also needs to be exposed through Transport Layer Security TLS for outside connection. The Kubernetes internal PKI plays a big role in maintaining these channels, hence directly affecting Kubernetes success.

This paper is organized as follows. Section 2 presents the overview of Kubernetes, namely the design principles and description of core components. Section 3 goes into the internal PKI, which operations require certificates, and which components have server and client certificates. Section 4 provides concluding remarks.

2 Kubernetes overview

In the recent years, Kubernetes has become the standard for container orchestration. Many organizations have successfully leveraged Kubernetes to gain better control over their containers. Before Kubernetes, containers are deployed directly on the environment which is hard to manage, monitor and maintain. As the demand for running containers not only for development purposes but also for production increases, so is the need for a tool to manage these containers in large scale manner. Kubernetes is designed to solve this problem and while there are several approaches to this problem such as Docker Swarm, Kubernetes is a success because of its design principles that allow easy integration (migrating to Kubernetes doesn't require code rewrite) and customisation (users could write operators for their own applications). [2]

2.1 Kubernetes design principles

1. Kubernetes API are declarative rather than imperative

   In Docker, the deployment process is done by the developers gaining remote access to server, starting the container and deploying it by himself. Hence, the process has to be repeated by the developer when he wants to deploy the container again. This process is called imperative. In Kubernetes, the developer only needs to define the desired state and lets Kubernetes find out how to reach that state. This style of configuration is called declarative and is similar to the way infrastructure-as-code tools such as Ansible and Terraform works. This not only saves developers a lot of time but also better guarantees that the container is deployed correctly.

2. There are no internal hidden APIs in Kubernetes control plane.

   We tend to think the control plane is the brain of the cluster, so it requires a lot of complex logic in order to make all decisions. However,

giving the control plane too many responsibilities will introduce a central point of failure as the cluster couldn't function correctly if the master node isn't available. Instead, what they do in Kubernetes is to store only the desired states of pods inside master node. Worker nodes constantly watch the API server to see compare between their current state and the desired state. If the desired state is different, the worker node will work to reach that state. Additionally, all components monitor their health and report to the master node.

This approach where system operates to reach a state is called level-triggering. Compared to edge-triggering, where a system responds to events so when the system doesn't work correctly, developers are responsible to send the events again, level-triggering has self-healing benefits. If a component crashes, when coming back up, it can look at the API server to restore the current state. Hence, there is no missing events issues.

Especially in distributing system, where components are expected to fail, system needs to be designed for reliability and tolerance. By using level-triggering methods, the responsibility for keeping the cluster healthy is distributed among all components. Hence, there is no central point of failure. Components continue to function independently in the event of failure in other components. This independence is a reason making Kubernetes composable and extensible. It's possible to write operators that support custom applications by making it interactable with API endpoints. Some interesting information that kubernetes API provide are:

- secrets: sensitive information such as passwords and certificates that shouldn't be inside containers need a way to be injected to the container at run time. Secret API object is what Kubernetes provides to accomplish that.

- configmaps: configuration information such as application startup parameters.

- downwardAPI: API that contains pod information, allowing containers to consume information about itself without consulting Kubernetes API.

The same API is used by internal components, is exposed to the outside. Hence, when writing application, developers could fetch secret object, configmap information, or information about the pod that application is running on from API server.

3. Meet the user where they are

   Kubernetes creators understand that not every organization is open to make modifications to their applications in order to accommodate new technologies. For example, for applications that are old but critical to the organizations such as PKI, requiring modifications in order to integrate with Kubernetes API is a major hinder to adoption because the cons of breaking the application are much greater than the pros of successfully migrating it. Hence, in order to encourage transition, it's possible to consume objects such as secrets, configmaps, downward API as files or environment variables within the containers. Information about dependent objects could be specified under pod definition, which Kubernetes will make sure to be mounted as a file or as an environment variable to the containers and hence the applications only need to be able to read from files or from environment variables.

4. Workload portability

   Kubernetes is comparable to an operating system OS in infrastructure level. In the old days, applications were written to work on specific hardware. Then OS came along to remove the trouble of considering hardware specification for developers by providing interfaces. The similar case is happening in distributed system world, where instead of deploying applications to fit certain cluster implementations, Kubernetes exposes its API so that applications could be deployed against it, making Kubernetes an abstraction layer for application deployment. This approach successfully separates application development from cluster implementation.

2.2 Kubernetes components

Core components of a Kubernetes cluster:

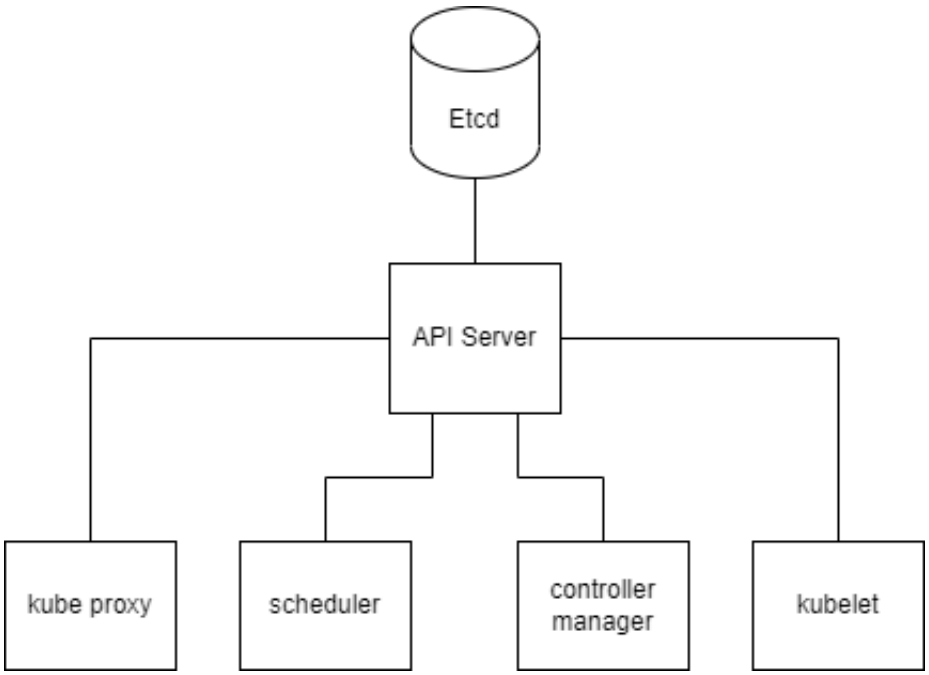


Figure 1. simple view of core Kubernetes components

1. etcd: key-value store inside control plane to store the components' states.

2. api-server: the most important component of the cluster because all interactions from outside or from other components to the cluster go through api-server.

3. scheduler: the scheduler monitors the kube api server looking for unscheduled pods. These are API objects of type pods that don't have the Node field filled in. So it starts to look for the best placement for that object then update the API server accordingly.

4. controller-manager: there needs to be a component to continuously watch the API server for the shared states in order to bring the current state towards the desired ones. Controller manager is such component, that operates as a non-terminating loop to regulate the state of the system. [1]

5. kubelet: exists in every nodes to act like a communication point between node and the cluster. Kubelet is responsible to self-report node health and current states to API server for monitoring.

6. kube-proxy: component that runs on each node for maintaining the network rules. These rules allow communication to pods from inside or

outside network.

API server is the central component and is the only component that talks to Etcd to store and retrieve cluster states. Other components such as kube-proxy, scheduler, controller manager and kubelet are API server clients. APIserver, scheduler, controller manager, and Etcd are control plane components while kube-proxy and kubelet belong to worker. [5]

3 Secure communication inside Kubernetes

As control plane components could spread between multiple nodes, we need a secure channel for the communication between the components [3]. Hence, we need an internal CA to verify each parties in the communication. The CA will play the role of the trusted third party that both client and server trust. When communicating, the client could ask the server for its certificate to verify and vice versa, before responding to a request, a server could also ask for its certificate.

3.1 PKI requirements for a Kubernetes cluster:

Kubernetes requires certificates for the following operations [1]:

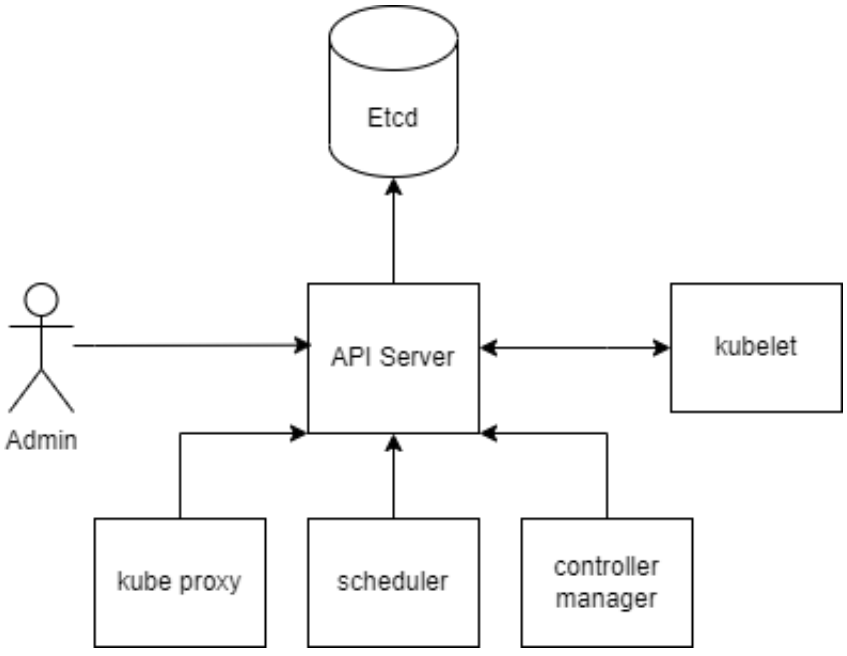


Figure 2. Kubernetes PKI communication flow

- server certificate for api-server endpoints

- server certificate for kubelet endpoints
- server certificate for etcd
- client certificate for administrators to communicate with api-server
- client certificate for api-server to communicate with etcd
- client certificates for controller-manager, scheduler, kubelet to communicate with api-server
- client certificates for api-server to communicate with kubelets
- client and server certificates for front-proxy

As api-server and kubelet expose their own API, both have server certificates. Other components need to have client certificates in order to communicate with servers. Because most components (except for etc-d) need to interact with api-server, they all have client certificates to api-server. API-server has client certificates to talk to kubelets on nodes (this is the case when api-server wants to retrieve logs and metrics information from the containers [3] and etc-d in order to read and retrieve cluster states. The diagram above shows the client-server relationship between components.

As the diagram shows, api-server and kubelet have both server and client certificates. While etc-d only has server certificate and only api-server has client certificate to talk to etc-d, the remaining components such as controller-manager, scheduler and kubelet are all api-server clients. Administrator certificates are specified in the user section of configuration files kubeconfig used by kubectl command line clients when connecting to cluster.

From the server side of view, any requests that present the clients signed by cluster CA are authenticated. Using the Common Name CN field and Groups field from the certificate, different access levels could be granted to different components, ensuring least privilege access to api resources.

3.2 Submitting certificate signing requests

When building a cluster, the operator needs to create cluster CA key and certificate then distribute the certificate to all control plane nodes. For worker nodes, the operator also needs to create key and certificates for each kubelets then send CSR certificate signing requests to the api-server. In order to simplify kubelet certificate signing process, the api-server provides a signing api where the kubelets could automatically submit their CSR on starting up.

For approving the requests, the operator could either configure the controller manager to automatically approve the CSR or wait for the CSR to be manually approved by an administrator. After successfully retrieve the signed certificate, the certificate is embedded inside the kubeconfig. The kubelet continues to operate normally [4].

The same process applies to users if they want to get access to the api-server. They create their own keys and certificates, sending CSR to api-server and wait for a cluster admin or controller manager to approve the requests. The process of sending CSR and receiving certificates of kubelets and users could be visualized using the diagram:

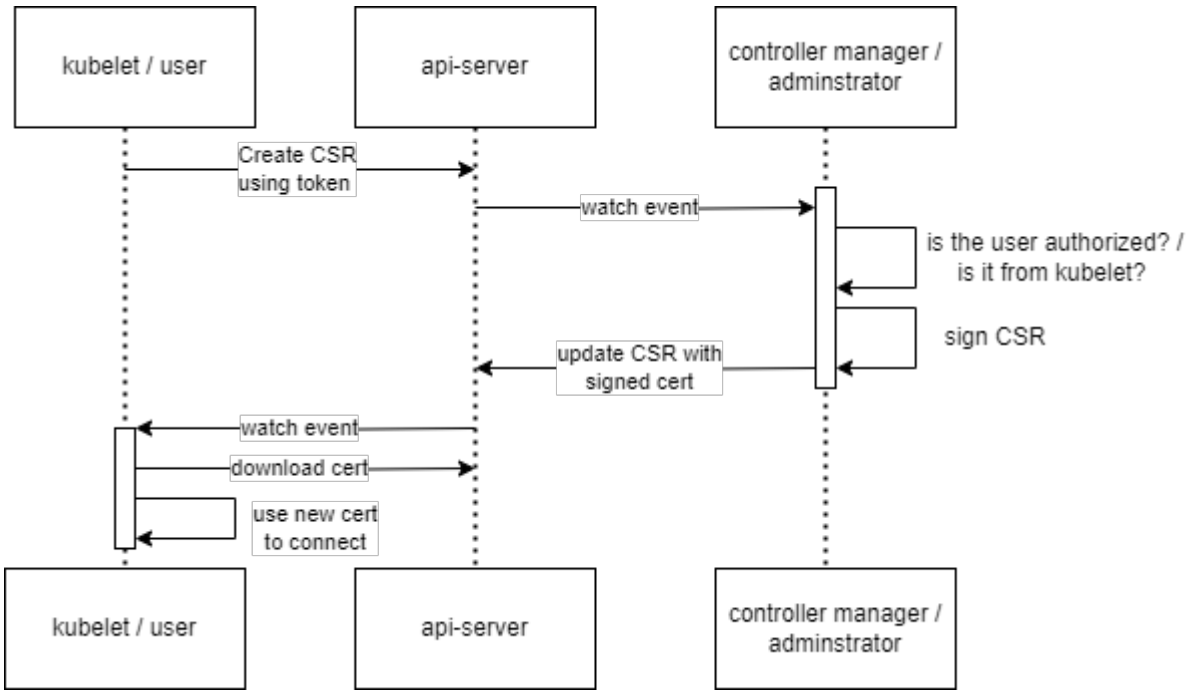


Figure 3. CSR signing process

3.3 Kubelet certificate rotation

There are limitations when it comes to certificates as the kubelets might fail to talk to api-server because of certificate expiration. To mitigate this, the kubelet could either self-request certification extension once near expiration date or rotate the serving certificate. Kubelet certification rotation is enabled by default and is done by creating new CSR only the old ones nearly expire.

3.4 TLS for service

Ingress has the ability to expose services through TLS. By defining ingress with key and certificate as secret, the service endpoint could be securely exposed with TLS. However, the CA in this case isn't cluster CA but it should be the an external CA so that the service could be trusted publicly on the internet.
The process of signing this ingress certificate could be also automated by cert-manager, which is certificate controller that allows certification management for container workload inside a Kubernetes cluster. This is where Kubernetes extensibility shows because cert-manager has the same work principle with a controller manager. Just as the controller manager that has specific controllers watching the api-server for specific objects in order to instruct the scheduler to schedule their creation, the cert-manager watches he api-server for ingress and certificate objects to see if they need to request new certificates or extend the current ones. The certificate object here is also an extended object from cert-manager as it doesn't belong to Kubernetes api.

4 Conclusion

The paper discusses the details of internal Kubernetes PKI, the creation of kubelet and user CSR and sign process for these requested certificates. The paper also takes the closer look of how Kubernetes is designed to easier understand the way it operates. Last but not least, the paper goes through how services could automatically obtain their certificates from external CA using cert-manager.

References

[1] Kubernetes documentation.

[2] Saad Ali. Kubernetes design principles: Understand the why. 2019.

[3] Alexander Brand. Certifik8s: all you need to know about certificates. 2017.

[4] Loïc Miller, Pascal Mérindol, Antoine Gallais, and Cristel Pelsser. Towards secure and leak-free workflows using microservice isolation, 12 2020.

[5] Piotr Tylenda Nassim Kebbani and Russ McKendrick. *The Kubernetes Bible*. Packt Publishing, 2022.

Implementing a Virtual Network System among Containers

Songlin Jiang
songlin.jiang@aalto.fi

Tutor: Tuomas Aura

Abstract

This paper investigates the method of implementing a virtual network system among containers. We first implement a testbed environment for Virtual Private Network (VPN) systems in IPv4 (site-to-site, host-to-host) and IPv6 (site-to-site). Then we compare VirtualBox-based Vagrant with Docker Compose in realizing the same networking features. The result shows that migrating the test from the Virtual Machines (VM) to the Docker containers can save nearly 90% of CPU time and 94% of memory for the VPN system. At the same time, the host machine still has no security risk increase.

KEYWORDS: Container, Network, Cloud, VPN

1 Introduction

In recent years, container technologies, such as Docker, have received much attention from the industry and academia as applications are moving to the cloud. Containers are much more efficient and lightweight than virtual machines because containers share the Linux kernel with the host. In contrast, virtual machines employ hardware virtualization and have their own kernel instance, which consumes more resources [16].

It is still a common practice [11] to build and test network systems configurations using virtual machines, which allows the network engineer to experiment in a virtual environment before setting up the physical system. In addition, virtual networks are sometimes also needed for automated integration tests, as developers may want to test the usability and performance of their software under some specific network topology.

However, creating a virtual computing environment can be slow and troublesome. The problems can worsen when testing virtual network systems with a large number of nodes on one host machine, as each network component and host needs a virtual machine instance. It can be memory-consuming to simultaneously run many virtual machine instances on one host machine to simulate the network environment, which significantly troubles testers working on personal computers short of memory. Moreover, Mac M1 / M2 chips are based on the ARM64 architecture, and virtual machine hypervisors currently have limited support for ARM64 hardware virtualization. In contrast, ARM64 is well supported by container runtimes [12]. Furthermore, Containers are easy to launch on demand in the cloud, and the cost is low because they can run within one virtual machine. Running with virtual machines can significantly increase the cost of Continuous Integration / Continuous Delivery (CI/CD) implementations in the cloud, as using nested virtualization with limited resources is hard. Most importantly, building, storing, and managing multi-platform virtual machine images in the cloud is also challenging. In contrast, Docker supports building multi-platform images [8] and uploading them to Docker Hub.

Due to little research on implementing network systems using containers, this paper investigates the possibilities of implementing virtual network systems based on Docker containers to overcome the disadvantages of virtual machines mentioned above. This paper also analyzes the functional and security limitations when virtual networks are implemented this way.

This paper is organized as follows. Section 2 reviews the current technologies used for container networking. Section 3 explains our case study of implementing a VPN system using Docker containers, while Section 4 explains the details of our implementation. Section 5 compares the VPN system performance when implemented with the virtual machines and containers. Finally, Section 6 provides the concluding remarks.

2 Docker Networking System

This section discusses the choice of the Docker container network driver for implementing the virtual network system. It also discusses how to enable routing, firewalls and IPv6 inside a container.

2.1 Network Drivers

Docker employs a pluggable networking subsystem. Several drivers can provide the Docker network functionality. The default drivers include the bridge, host, overlay, IPVLAN, MACVLAN, and none. We can choose one of them to implement the virtual network system.

The none, host, and overlay drivers do not cater for our needs. Firstly, the none driver disables all networking. Secondly, the host driver is unnecessary as it can have security implications due to its nature of sharing the same network stack with the host machine. In addition, the overlay is not an appropriate option as we are simulating the network system only on one host machine.

As a result, the possible candidates are bridge, IPVLAN, and MACVLAN. When examining the details, the bridge learns Media Access Control (MAC) addresses by checking the frame headers sent by the communicating hosts. On the other hand, the MACVLAN is a trivial bridge that does not need to learn as it already knows every MAC address it can receive [4]. IPVLAN is similar to MACVLAN, except IPVLAN assigns the same MAC address to all containers attached to it. In contrast, MACVLAN assigns a different MAC address to each attached Docker container [5].

MACVLAN is the best choice for our needs. As we only use the driver to implement the internal network, there is no need to use advanced flood control and forwarding database manipulation that are specific to the bridge driver. MACVLAN bridge mode allows the testbed network to run layer 2 (data link layer) protocols, such as the Address Resolution Protocol (ARP) and Link Layer Discovery Protocol (LLDP). MACVLAN also supports address configuration and discovery protocols, as well as other multicast protocols, such as Dynamic Host Configuration Protocol (DHCP) [17] and Precision Time Protocol (PTP) [1]. Moreover, according to Docker documentation related to networking [9], MACVLAN networks are the best choice when migrating from a VM setup, as MACVLAN makes the container appear as a physical device with its own MAC address. Furthermore, Gundall et al. [10] conduct benchmarks for different virtualiza-

tion technologies for the networking overhead. The result shows that the MACVLAN driver performs best in throughput while requiring the least CPU resources compared to the other network drivers.

2.2 Routing and Firewall

By default, Docker do not allow manipulating container network devices and setting routing tables or firewalls inside the containers. These features can be enabled by assigning the net_admin capability to the container.

According to the capabilities man page [13], assigning the net_admin capability to containers allows the following network-related operations:

1. Making interface configuration;
2. Administrating IP firewall, masquerading, and accounting;
3. Modifying routing tables;
4. Binding to any address for transparent proxying;
5. Setting the type-of-service (TOS);
6. Clearing driver statistics;
7. Setting the promiscuous mode;
8. Enabling multicasting;
9. Using setsockopt(2) to set the following socket options: SO_DEBUG, SO_MARK, SO_PRIORITY (for a priority outside the range 0 to 6), SO_RCVBUFFORCE, and SO_SNDBUFFORCE.

As we use the MACVLAN to build the internal network, all the network manipulations mentioned above only work for the network components that belong to the corresponding network namespace. There are no security implications to the host network if we give net_admin capability to the container with its network namespace.

2.3 IPv6

Configuring IPv6 networking in Docker [7] is also possible. Docker disables the IPv6 support by default. We can add the following content to the daemon configuration file (default location at: `/etc/docker/daemon.json`) or corresponding settings in Docker Desktop:

```
{ "ipv6": true, "fixed-cidr-v6": "fd00::/80" }
```

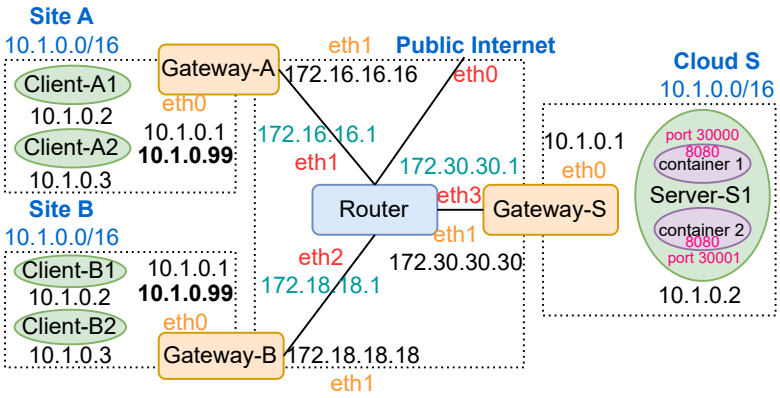


Figure 1. Host to Host

3 Case Study: VPN

This paper simulates a scenario where the IoT devices (clients) in two sites, A and B, would like to connect to the server in the cloud S. The topology is based on the Aalto University CS-E4300 Network Security 2022-2023 instance Project 2 [3], where site A, B, and cloud S both use the private IP addresses to improve the security and save the IPv4 addresses. Gateways A, B, and S connect sites A, B, and S to the public Internet. The router in the topology represents the Internet between the sites and the cloud. The address space between the gateway and the router simulates public, routable IPv4 addresses, although they are all private. Site A, B, and cloud S use the router to access the Internet.

In order to make the clients in both site A and site B connect to the cloud server safely, this paper attempts to implement a virtual network testbed for this networking exercise based on Docker containers. We experiment with two types of VPN: site-to-site and host-to-host, using strongSwan, a VPN implementation based on Internet Protocol Security (IPsec).

3.1 Host to Host

A host-to-host VPN connects different gateways together. IP packets then get routed to different clients within the corresponding site according to the routing table of the gateway [14].

Figure 1 shows the topology and corresponding address spaces under such circumstances.

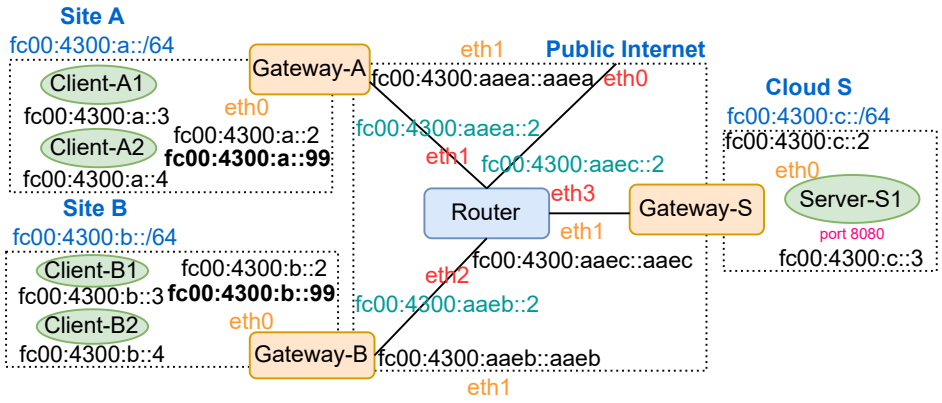


Figure 2. Site to Site in IPv6

3.2 Site to Site

A site-to-site VPN connects different network systems located at different sites together directly [2]. In this case, the address spaces of sites A, B and the cloud network S should not overlap [15].

Our site-to-site VPN topology is almost the same as figure 1, except that the address space for site B is 10.2.0.0/16, and cloud S is 10.3.0.0/16.

3.3 Site to Site in IPv6

Site to Site in IPv6 is similar to Site to Site VPN in IPv4 but replaces all the address space in IPv6.

Figure 2 shows the topology and corresponding address space under such circumstances.

4 Implementation

To manage the Docker containers, we use Docker Compose for orchestration. Based on the figures above, we make each network component a separate container, assign the net_admin capability to each container, and implement the network with the MACVLAN driver. We only connect the router to the Docker default bridge network to enable connections to the public Internet. Finally, we set up the routing and firewall and configured the strongSwan IPsec with certificates.

We use Network Address Translation (NAT) masquerade for the gateway interface to prevent leaking local IP addresses outside their subnets for routing. We bind the preconfigured local server IP address 10.1.0.99 or 10.2.0.99 to the interface eth0 of corresponding gateway A and B accord-

ing to their subnets.

We use strict firewall rules on the clients, assuming there should be no need for the clients (IoT devices) to visit the Internet. We use iptables to set up gateway A and B firewall rules for input and output. We accept Internet Key Exchange (IKE) and Encapsulating Security Payload (ESP) traffic (port 500 and 4500 in UDP) from and to the cloud. Finally, we drop everything else, including the connection from and to the Internet.

As explained in the following sections, there are also some differences between the two kinds of VPN setups, and between IPv4 and IPv6.

4.1 Host to Host

About routing, for gateway A and B, we redirect the traffic from the original local server address 10.1.0.99 of port 8080 to cloud gateway S of port 8080 with Destination NAT. For gateway S, we redirect the traffic from the client gateway A and B of port 8080 to corresponding ports (30000 and 30001) on the server s1 address.

There are overlapping network address spaces for host-to-host VPN. Suppose we specify the IP address and subnet directly through Docker Compose. In that case, there will be errors notifying us that 'Pool overlaps with another one on this address space'. Although technically this should not be a problem, as we are creating a separate internal network without direct routing, Docker still forbids us.

We have addressed the issue of overlapping address space through a method that can be likened to IP address spoofing. When we do not specify the IP address of the network in Docker Compose, it will assign a random address from the Docker address pool to the network interface. Now we can modify the IP address and subnet of the interface to our desired one, and no error will be thrown now.

In addition, for server S1, we use the Docker-in-Docker image to run Docker containers inside the server S1 container. Running that image requires the container to be privileged according to the documentation [6]. As a result, in practice, we must ensure the software running in server S1 is benign and poses no risk to the host machine. It can be acceptable in a testbed network but not in production. Otherwise, we recommend having a separate VM for the servers.

4.2 Site to Site

Concerning routing, for gateways A and B, we redirect the traffic from the local address (10.1.0.99 and 10.2.0.99) of port 8080 to cloud server S1 on port 8080 with Destination NAT.

For the firewall on gateways A and B, we also have to accept the post-routing traffic to the cloud server 10.3.0.3.

There is no overlapping network address space within the site-to-site VPN. Thus, we can specify the IP address and subnet directly through Docker Compose. Additionally, we avoid using the IP address ending in ".1", as Docker does not allow us to assign that address to any container. These addresses are reserved for gateways or routers on a particular network (although we are simulating the Gateway and Router).

4.3 Site to Site in IPv6

Site to Site in IPv6 is similar to Site to Site in IPv4. Just replacing all the IPv4 addresses with IPv6 would complete the job. The only difference is that, in addition to the existing configurations, we also have to allow ICMPv6 traffic at the gateways for firewall rules. In IPv6, Neighbor Discovery is a necessary component, replacing Address Resolution Protocol (ARP) in IPv4. This way, IPv6 Neighbor Discovery can work, and different containers can communicate within their subnet. We also need ICMPv6 for Destination Unreachable messages.

5 Evaluation

This section summarizes the result of our experiment. It makes a comparison between the virtual-machine-based testbed and our container-based testbed.

5.1 Usability and Portability

We can use Docker Buildx [8] to create the testing environment images for multiple platforms with only one machine, then upload them to Docker Hub for reusing. Developers do not need to be aware of the different processor architectures when starting the container-based testbed. In contrast, VM monitors, such as VirtualBox, usually do not have a centralized image hub. The developer must choose the right image based on the ar-

Table 1. Performance Test Result in Average

Solution	Boot Time[1]	Memory[2]
Docker Compose	75 s	278 MB
Vagrant + VirtualBox	689 s	4.5 GB

chitecture before running the VM testbed.

Most importantly, the Docker setup can run in M1/M2-based macOS with Docker Desktop. In contrast, the VM ones cannot be run due to the limited support of VirtualBox for ARM64.

The shell script commands for all the routing and firewall configurations in Docker containers and VMs are identical. Hence migrating configurations from the VMs to the Docker containers is easy. We can simultaneously start the site-to-site and host-to-host VPN setup in Docker without interfering with each other.

5.2 Performance

We use Docker Engine 23.0.1 to run the containers and VirtualBox 7.0.6 to run the Virtual Machines. Table 1 reflects the average situation for all the three implementations. Table 1 shows that our container solution significantly reduces the fresh boot time[1] for the whole VPN system by nearly 90%. It dramatically reduces the memory consumption[2] by nearly 94% as well.

5.3 Security

We can check the current virtual network devices with the command `ls /sys/devices/virtual/net -l`. It shows different results when executing from the host machine and inside the container, indicating that the network stacks inside containers are entirely isolated from the host machine.

5.4 Limitations

There are also some limitations of the Docker networking model, which cause several observable differences compared to VMs. However, these limitations generally have workarounds to bypass and will not stop us from adopting container solutions.

[1]Also include the running environment building time for the host platform
[2]Maximum value during the whole running process

1. We cannot have overlapped IP address ranges in different virtual network interfaces assigned by Docker. The only way to do that is to configure the IP addresses manually inside containers.
2. We are not allowed to assign the IP address ending in ".1" to a Docker container, and we will only get a warning if we do that in a VM. Similar to the previous limitation, a workaround is to assign the IP addresses ending in ".1" manually inside containers.

In addition, if we want to test the scalability of the network software and run Docker containers inside a Docker container, that container needs to be privileged. Such requirements may cause some security risks, but this is unnecessary for implementing virtual networks.

6 Conclusion

This paper investigates how to construct the testbed network environment through the case study of container-based and VM-based VPN configurations. Containers are much more lightweight than virtual machines. As a mature virtualization technology, containers can realize every functionality we require, similar to virtual machines, while increasing no security risk to the host machine.

We hope this paper can inspire researchers and engineers to migrate their testing environment related to network systems from VMs into containers.

Source code for this paper: *https://github.com/HollowMan6/Implement-VPN-System-with-Containers/tree/main/src*

References

[1] Giuliano Albanese, Robert Birke, Georgia Giannopoulou, Sandro Schönborn, and Thanikesavan Sivanthi. Evaluation of networking options for containerized deployment of real-time applications. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021.

[2] Aung, Si Thu and Thein, Thandar. Comparative Analysis of Site-to-Site Layer 2 Virtual Private Networks. In *2020 IEEE Conference on Computer Applications(ICCA)*, 2020.

[3] Aura, Tuomas and Peltonen, Aleksi and Bui, Thanh. Tuomaura/CS-e4300_testbed: TESTBED network setup for Student Projects, Dec 2022. GitHub Repository.

[4] Cha, Jae-Geun and Kim, Sun Wook. Design and Evaluation of Container-based Networking for Low-latency Edge Services. In *2021 International Conference on Information and Communication Tech Convergence (ICTC)*, pages 1287–1289, 2021. IEEE.

[5] Claassen, Joris and Koning, Ralph and Grosso, Paola. Linux containers networking: Performance and scalability of kernel modules. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 713–717, 2016.

[6] Docker contributors. Docker-in-Docker image README, February 2023. Docker Hub.

[7] Docker contributors. Enable ipv6 support, February 2023. Docker Documentation.

[8] Docker contributors. Multi-platform images, March 2023. Docker Documentation.

[9] Docker contributors. Networking overview, February 2023. Docker Documentation.

[10] Gundall, Michael and Reti, Daniel and Schotten, Hans D. Application of Virtualization Technologies in Novel Industrial Automation: Catalyst or Show-Stopper? In *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 283–290, 2020.

[11] Hauser, Frederik and Häberle, Marco and Schmidt, Mark and Menth, Mich. P4-IPsec: Site-to-Site and Host-to-Site VPN With IPsec in P4-Based SDN. *IEEE Access*, 8:139567–139586, 2020.

[12] Kaiser, Shahidullah and Haq, Md. Sadun and Tosun, Ali Saman and Korkmaz, Turgay. Container Technologies for ARM Architecture: A Comprehensive Survey of the State-of-the-Art. *IEEE Access*, 10:84853–84881, 2022.

[13] Linux contributors. *Capabilities(7)*, February 2023. Linux Man Page.

[14] Du Meng. Implementation of a host-to-host vpn based on udp tunnel and openvpn tap interface in java and its performance analysis. In *2013 8th International Conference on Computer Science & Education*, pages 940–943. IEEE, 2013.

[15] Oğuzhan Akyıldız and İbrahim Kök and Feyza Yıldırım Okay and Suat Özdemir. A P4-assisted task offloading scheme for Fog networks: An intelligent transportation system scenario. *Internet of Things*, 22:100695, 2023. Elsevier.

[16] Sharma, Prateek and Chaufournier, Lucas and Shenoy, Prashant and Tay, Y. C. Containers and Virtual Machines at Scale: A Comparative Study. In *Proceedings of the 17th International Middleware Conference*, Middleware '16. ACM, 2016.

[17] Arne Wendt and Thorsten Schüppstuhl. Proxying ros communications — enabling containerized ros deployments in distributed multi-host environments. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 265–270, 2022.

Virtual reality toward the internet of senses

Tenho Korhonen
tenho.korhonen@aalto.fi

Tutor: Nassim Sehad

Abstract

Existing virtual reality (VR) and augmented reality (AR) applications combined with next-generation networking technologies propose a wide range of possible new applications. Leveraging these technologies, the Internet of Senses denotes a concept which aims to enable immersive multisensory interaction over a network with a remote real environment or a virtual environment.

Implementing the IoS contains a variety of challenges regarding communication networks, user experience and cybersecurity. This paper introduces these chalenges and concepts related to the IoS, and examines two challenges more closely: networking technologies and multisensory synchronization. The most prevalent challenges in these domains include the requirement for low latency, great bandwidth and high reliability, as well as the lack of standardized methods for multisensory synchronization.

KEYWORDS: Internet of Senses, mulsemedia, latency, multisensory synchronization

1 Introduction

Current virtual reality (VR) technology is able to effectively simulate the real world in terms of vision and sound, thus enabling the user to experience high levels of immersion and embodiment. However, in the real world, we observe our surroundings with more senses than only hearing and vision. In particular, tactile sensation – the sense of touch – is an essential way for humans to interact with their environment.

The fifth generation of networks has emerged, and building onto it, the International Telecommunication Union (ITU) has introduced a concept known as tactile internet: the Internet that combines "extremely low latency" with "high availability, reliability and security" [1]. These technologies have opened a wide range of applications, and combined with existing technologies, such as Extended Reality (XR), holoportation, and remotely controlled machines, they have potential to revolutionize the future of factories, smart cities, and digital healthcare [1].

The Internet of Senses (IoS) aims at transforming the digital world into a fully immersive one by delivering multisensory experiences over the network; experiences that affect more senses than hearing and vision. Providing these experiences has two requirements. The first requirement is multisensory user interfaces that can stimulate additional senses of the user, such as haptic and smell. The second requirement is ultra-low latency networks that support synchronization and delivery of multisensory data. The data used by these systems is referred to as multiple sensorial media (mulsemedia). Currently, the development of haptic interfaces is still in its infancy, and additional advancement in networking technologies is also needed to enable mulsemedia.

The goal of this paper is to review literature on technologies that are needed to enable effective multisensory interaction, and provide adequate background information about Mulsemedia and the IoS. This paper, in particular, examines two topics relating to this domain: networking technologies for multisensory interaction, and multisensory synchronization. Another topic that is closely related to the IoS is multisensory interfaces, but these are left outside the scope of this article.

This paper is structured in the following way. Section 2 presents the general idea and concepts of IoS. Section 3 presents existing research about the technologies of IoS, and Section 4 provides conclusions based on the findings in the previous sections.

2 Background

2.1 Heuristics and concepts

Mulsemedia and the Internet of Senses are both parts of a central concept known as the Metaverse. It is defined as "the post-reality universe, a perpetual and persistent multiuser environment merging physical reality with digital virtuality" [2]. In other words, the Metaverse is a network of virtual environments, and it is especially focused on immersive social interaction between users by leveraging virtual reality (VR) and augmented reality (AR) technologies, and enabling users to dynamically interact over the Internet with low latency [2]. This definition of the Metaverse suggests that interaction in the Metaverse should resemble interaction in the real world.

The Internet of Senses is a concept that addresses this demand for a realistic interaction. The IoS is a fairly modern concept which aims to provide a more realistic user experience by leveraging mulsemedia in addition to audio and visuals, enabling a more "real" interaction. While the concept of Metaverse usually includes social aspect of interaction, the IoS refers to any multisensory interaction in real-time, and it includes a variety of other applications in addition to social interaction. For example, the IoS could be leveraged in virtual tourism, enabling the user to feel as if they were actually present in a distant location, or in healthcare, where surgery could be performed without the surgeon being physically present [3].

Existing research has been conducted about mulsemedia. Falk et al. [3] argue that producing immersive experiences requires stimulating more senses than hearing and vision. They also discuss the general heuristics of the IoS in more detail and elaborate on some existing problems with the IoS, such as the requirement for low latency. The authors focus mainly on the human and environmental aspects of IoS.

A recent advance has been made in *affective computing* [3]. Affective computing is defined as "the use of devices and algorithms to recognize, interpret, process, and simulate different human affective states" [3]. Affective computing aims to monitor the user's mental state in real time [4], which instantly allows producing a better quality of experience (QoE) for the user. The mental state can be followed by observing the user externally, including the user's facial expression, eye gaze, posture, body move-

ment, tone of speech, heart rate and skin perspiration [4]. Affective computing plays a significant role in communication through the Metaverse, since a large portion of real-life human interaction is based on non-verbal communication [3], and an adequate Metaverse experience should also be able to produce this sensation.

While affective computing can be practiced by externally monitoring the user, an even deeper understanding of the user's state and perception can be achieved by measuring the neural activity in the user's brain. The tools for this task are referred to as brain-computer interfaces (BCI), and they can be implemented using neuro-imaging tools, such as electroencephalography (EEG) [5]. A BCI enables computer software to monitor the user's cerebral activity, thus enabling the computer to interpret instructions from the user's thoughts [6]. BCI technologies are typically complex, but recent advancements have been made in BCI technologies both in laboratory and real-world settings that could bring BCI technologies closer to consumers [3, 7].

As seen this section, Mulsemedia can contain very sensitive private information. Therefore, privacy and cybersecurity should be vital factors when handling such data. Possible solutions for these problems include quantum computing and quantum cryptography, and existing research also displays federated learning [8] and blockchain [9] as promising solutions [3]. This cybersecurity aspect, however, will not be the focus of this article.

2.2 Challenges of the IoS

Implementing the IoS proposes a variety of challenges, and this subsection summarizes the most prevalent challenges in the topic. We will comply with existing categorization [3] by diving the challenges in two domains: *communication networks and AI*, and *user experience monitoring*. The final category, *cybersecurity*, is intentionally left outside the scope of this article, and it remains as a topic for future articles to explore.

The first category, *communication networks and AI* focuses on the communication between IoS users. The greatest technical challenges include the low latency and high transmission rates required for IoS. This category also contains the possible environmental problems caused by the energy consumption of deep neural networks, since the other problems in this category are addressed with the help of these networks. Additionally, this category contains the practical difficulty of integrating various dif-

ferent technologies together and the various problems arising from this integration process [3].

The problems with *user experience monitoring* are focused on affective computing, mobile brain-computer interfaces, and multisensory QoE. Affective computing requires context-aware methods, and the computing device must be durable and reliable. A possibility for a non-contact affective computing method was also mentioned, which would not need cameras to observe the user's actions. Another problem in this category is investigating how faulty multisensory feedback, for example due to a network error, affects the user experience [3].

3 Topics of research

This section reviews existing research on the technologies needed for implementing the IoS, and introduces additional terminology of the field.

3.1 Minimising latency, maximising reliability

As stated in Section 2.2, the IoS requires very low latency to function – something which is not attainable by traditional networking and cloud computing conventions. Despite the demanding nature of the challenges, a variety of technologies exist for solving these problems.

Mobile edge computing (MEC) is a vital aspect of minimising network latency. Edge computing refers to on placing computation, storage and network control of applications to the edges of the network in contrast to cloud computing [10], where such activities are performed in cloud-based clusters in a centralized manner. MEC is a subcategory of edge computing, where computation occurs both on local devices and at the edge of the network [11]. In MEC, the computation units close to the user are referred to as *base stations*, and performing the computation in these stations reduces the workload on the users' end devices [12]. MEC is a fairly modern concept and a key technology in 5G networks and IoT applications, because it reduces latency for end users [12].

Artificial intelligence technologies (AI) have seen recent improvement that has produced powerful tools applicable to implementing the Metaverse [3]. In combination with edge computing, AI has introduced a new concept: *edge intelligence* [13]. Edge intelligence introduces new network concepts such as edge offloading, edge caching, local machine learning and

model training, and edge resource optimization [13].

Powerful AI tools also have a drawback, namely, they consume a great amount of energy, which poses an environmental problem. A possible solution for this problem would be optical neural networks that consume significantly less energy than conventionally implemented neural networks [3].

Chaccour et al. [14] investigate the potential of terahertz Internet (referred to simply as THz) in wireless VR applications. Their results demonstrate that terahertz-magnitude bandwidths have potential to support high-performance VR needs, but in some conditions terahertz technology also had difficulty in providing a reliable QoE.

Van Hyunh et al. [15] research digital twin (DT) supported Metaverse by utilizing mobile edge computing and ultra-reliable and low latency communications (URLLC), which are examined more closely in the following subsection. DT is a design of producing a virtual replica of real-life objects and rendering these objects in real time, thus producing a "digital twin" of a real-life scene. In their mathematically-oriented article, the authors "formulate a latency minimization problem under stringent constraints of URLLC-based transmissions by optimizing edge caching strategies, task offloading policies, as well as computation and communication resources", and propose a solution to this minimization problem [15].

Joda et al. [16] investigate IoS-related edge intelligence. Edge intelligence refers to leveraging AI and machine learning (ML) tools in conjunction with edge computing. The authors propose a solution for organizing *semantic communication* between edge nodes.

Semantic communications denote a concept where increasing emphasis is placed on network reliability and the "meaning" of the transmitted data [17], i.e., the semantics of the data. These semantics – the intended purpose of the transmitted data – could then, for example, affect network scheduling policies in order to provide a more reliable connection. Lan, et al. [17] argue that the sixth generation of networks (6G) will require increasing attention to semantic communications, and mention three types of these communications: human-to-human, human-to-machine and machine-to-machine.

Lopez et al. [18] research channel state information (CSI) on the transmitter side (CSIT) in a 6G network context. CSI refers to information related with wireless network channels, which can be used to monitor the surrounding wireless environment [19]. The authors argue that tradi-

tional CSIT might not be capable of meeting the stringent requirements of 6G networks, and propose possible solutions and directions for future research in this topic. The study reveals that aqcuiring instant CSI with a high quality-of-service (QoS) demand is highly expensive. Based on this, the authors discuss various approaches to this problem, ranging from "efficient allocations of the pilot sequences up to schemes relying on statistical CSIT or location-based ML/AI-enabled predicted CSIT" [18].

3.2 Multisensory synchronization

Handling Mulsemedia data involves handling multiple input streams of data for different senses. These data streams must be synchronized in order to provide the user with high QoE, which requires thorough understanding of the user interface [20]. Synchronizing the data streams is a difficult and error-prone task [21], and no standard way of synchronizing these data exists yet, albeit some research has been conducted on the subject. Thus, multisensory synchronization remains as a topic for future research to explore. Yuan et al. [20] explore Mulsemedia synchronization with two multisensory data streams, haptics and air flow, and present guidelines for synchronization errors regarding these streams.

Bi et al. [22] present a solution for Mulsemedia streaming by adapting the Dynamic Adaptive Streaming over HTTP (DASH) protocol, which is commonly used for multimedia streaming, and expanding it to support Mulsemedia (DASH-MS). As the name of the protocol states, it delivers Mulsemedia content over a network with adaptive quality. The authors also noticed that experimental results demonstrated multisensory stimuli improving the satisfaction level of the user, even when out-of-sync multisensory effects were used.

Abreu et al. [21] present a semi-automatic approach for synchronizing mulsemedia streams using neural networks that detect information from video scenes. This synchronization technique is used in the context of a mulsemedia authoring tool to automatically create multisensory stimuli from the played-back audiovisual content. The presented algorithm recognizes the starting times for different multisensory effects, but does not consider the characteristics of the effects, such as intensity, type or position.

4 Discussion

While it may currently seem very distant to imagine a network of virtual realities and applications indistinguishable from the real world, it is clear that IoS could massively impact a large portion of the society. For this to happen in practice though, many issues would need to be solved perfectly: for example, the multisensory devices and interfaces should be unnoticeable for the user, the user's physical and mental state should be monitored extensively, and all this should happen in real time with minimal latency. Therefore, while some competent mulsemedia applications already exist, a flawless IoS experience may still lie in the distant future.

When minimising latency, one must keep in mind that the speed of light sets a boundary for transmitting information between distant location on the surface of the Earth. Therefore, the mulsemedia delivery system will always have some latency, unless some form of hypothetical prediction algorithm could be leveraged, which would pre-emptively detect both the user's actions and the remote environment's responses slightly before these events actually happen. This idea was not researched in the scope of this article, and exploring it will be left for future articles.

When examining multisensory synchronization, finding related research proved to be a rather difficult task. While limited information can be found about multisensory synchronization in the context of playing back multisensory content, even less information was found in the context of the IoS where multisensory stimuli occur in real time. Synchronizing mulsemedia streams remains extensively as a topic for future research. Section 3.2 introduced a way of automatically detecting multisensory stimuli from audiovisual content. This could hypothetically be leveraged in the context of IoS to obtain multisensory information from audiovisual content without the need for additional remote receptors for multisensory stimuli, such as smell. This idea is left for future articles to explore.

5 Conclusion

Recent development in networking technologies combined with existing technologies, such as VR and AR, has potential to create a variety of novel applications. One application domain is referred to as the Internet of Senses, a Metaverse-based multisensory form of interacting over the internet. Although concrete results have been obtained in the research

field and existing solutions for the challenges have emerged, a variety of challenges remain to be tackled in the future. This article presented an overview of current networking technologies needed for transmitting mulsemedia, and summarized current research on multisensory synchronization. Lastly, some discussion was provided based on the findings in this article.

References

[1] G. Fettweis, H. Boche, T. Wiegand, E. Zielinski, H. Schotten, P. Merz, S. Hirche, A. Festag, W. Häffner, M. Meyer, *et al.*, "The tactile internet-itu-t technology watch report," *Int. Telecom. Union (ITU), Geneva*, 2014.

[2] S. Mystakidis, "Metaverse," *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.

[3] T. H. Falk, L. B. Le, and R. Morandotti, "The internet of senses: A position paper on the challenges and opportunities of multisensory immersive experiences for the metaverse," in *2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, pp. 139–144, 2022.

[4] J. Tao and T. Tan, "Affective computing: A review," in *Affective Computing and Intelligent Interaction* (J. Tao, T. Tan, and R. W. Picard, eds.), (Berlin, Heidelberg), pp. 981–995, Springer Berlin Heidelberg, 2005.

[5] Y. Wang, W. Song, W. Tao, A. Liotta, D. Yang, X. Li, S. Gao, Y. Sun, W. Ge, W. Zhang, and W. Zhang, "A systematic review on affective computing: emotion models, databases, and recent advances," *Information Fusion*, vol. 83-84, pp. 19–52, 2022.

[6] L. F. Nicolas-Alonso and J. Gomez-Gil, "Brain computer interfaces, a review," *Sensors*, vol. 12, no. 2, pp. 1211–1279, 2012.

[7] P. Aricò, G. Borghini, G. Di Flumeri, N. Sciaraffa, and F. Babiloni, "Passive bci beyond the lab: current trends and future directions," *Physiol Meas*, vol. 39, 2018. Available at: http://dx.doi.org/10.1088/1361-6579/aad57e.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, pp. 50–60, may 2020.

[9] T. R. Gadekallu, T. Huynh-The, W. Wang, G. Yenduri, P. Ranaweera, Q.-V. Pham, D. B. da Costa, and M. Liyanage, "Blockchain for the metaverse: A review," 2022.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[11] T. Bai, C. Pan, Y. Deng, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Latency minimization for intelligent reflecting surface aided mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 11, pp. 2666–2682, 2020.

[12] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5g and internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[13] W. Y. B. Lim, Z. Xiong, D. Niyato, X. Cao, C. Miao, S. Sun, and Q. Yang, "Realizing the metaverse with edge intelligence: A match made in heaven," *IEEE Wireless Communications*, pp. 1–9, 2022.

[14] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Can terahertz provide high-rate reliable low-latency communications for wireless vr?," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9712–9729, 2022.

[15] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Wireless Communications Letters*, vol. 11, no. 8, pp. 1733–1737, 2022.

[16] R. Joda, M. Elsayed, H. Abou-zeid, R. Atawia, A. B. Sediq, G. Boudreau, M. Erol-Kantarci, and L. Hanzo, "The internet of senses: Building on semantic communications and edge intelligence," *IEEE Network*, pp. 1–9, 2022.

[17] Q. Lan, D. Wen, Z. Zhang, Q. Zeng, X. Chen, P. Popovski, and K. Huang, "What is semantic communication? A view on conveying meaning in the era of machine intelligence," *CoRR*, vol. abs/2110.00196, 2021.

[18] O. L. A. Lopez, N. H. Mahmood, H. Alves, C. M. Lima, and M. Latva-aho, "Ultra-low latency, low energy, and massiveness in the 6g era via efficient csit-limited scheme," *IEEE Communications Magazine*, vol. 58, no. 11, pp. 56–61, 2020.

[19] Y. Ma, G. Zhou, and S. Wang, "Wifi sensing with channel state information: A survey," *ACM Comput. Surv.*, vol. 52, jun 2019.

[20] Z. Yuan, T. Bi, G.-M. Muntean, and G. Ghinea, "Perceived synchronization of mulsemedia services," *IEEE Transactions on Multimedia*, vol. 17, no. 7, pp. 957–966, 2015.

[21] R. Abreu, D. Mattos, J. A. F. d. Santos, and D. C. Muchaluat-Saade, "Semi-automatic synchronization of sensory effects in mulsemedia authoring tools," in *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, WebMedia '19, (New York, NY, USA), p. 201–208, Association for Computing Machinery, 2019.

[22] T. Bi, A. Pichon, L. Zou, S. Chen, G. Ghinea, and G.-M. Muntean, "A dash-based mulsemedia adaptive delivery solution," in *Proceedings of the 10th International Workshop on Immersive Mixed and Virtual Environment Systems*, MMVE '18, (New York, NY, USA), p. 1–6, Association for Computing Machinery, 2018.

Analysis of GPU Architecture for High-Performance Stencil Computing

Tomi Molander
tomi.molander@aalto.fi

Tutor: Maarit Korpi-Lagg

Abstract

This study offers a comparative investigation of the parallel performance of NVIDIA and AMD HPC GPUs. The analysis compares the relative performance of NVIDIA and AMD GPUs for high-performance stencil computing applications and assesses the effect of GPU architecture on performance in the context of the CUDA-MPI library Astaroth. The architectures of the NVIDIA Tesla A100 and AMD MI100 GPUs, both of which are intended for use in data centers and high-performance computing applications, are examined in this study. The main hardware and execution model variations between the GPUs are outlined. The study concludes that the relative performance of Tesla A100 and AMD MI100 GPUs for stencil computations depends on the specific hardware and software configurations used.

KEYWORDS: NVIDIA, AMD, Graphics Processing Unit, GPU, Architecture, CUDA-MPI, Astaroth, Performance

1 Introduction

The field of computer graphics and high-performance computing has experienced tremendous growth in recent years, largely driven by advance-

ments in graphics processing units (GPUs). NVIDIA and AMD are two of the largest manufacturers of GPUs, and their products are widely used in a variety of applications, from video gaming to scientific simulations. In this context, it is important to understand the performance of these GPUs and their potential to handle complex tasks efficiently. This paper aims to estimate the effect GPU architecture has on multi-GPU performance of NVIDIA and AMD GPUs in the context of the CUDA-MPI (Message Passing Interface) library Astaroth.

CUDA [9] and its counterpart OpenCL [4], are parallel computing platforms that allow for the development of high-performance applications for GPUs. CUDA is a proprietary platform developed by NVIDIA and is specific to NVIDIA GPUs. However, OpenCL is an open standard for cross-platform, parallel programming that can run on GPUs from multiple vendors, including NVIDIA, AMD, and Intel. AMD also has its own open-source software platform known as ROCm that is designed for AMD hardware. Lastly, Heterogeneous-Compute Interface for Portability (HIP) is a software interface for developing portable parallel applications that run on both AMD and NVIDIA GPUs [3].

Astaroth is a high-performance computing library designed to utilize the parallel processing capabilities of GPUs [10]. It has a comprehensive set of tools for developing and running GPU-accelerated applications by providing GPU Application Programming Interfaces (APIs), domain-specific language (DSL) and a compiler for translating the DSL into CUDA/HIP subroutines. More specifically, Astaroth is using the HIPIFY tool of HIP for translating CUDA to HIP. This study evaluates the parallel performance of NVIDIA and AMD GPUs by comparing their underlying architectures in the context of stencil calculation.

The GPUs compared in this paper are NVIDIA Tesla A100 and AMD MI100. The paper aims to provide initial insights into the relative performance of NVIDIA and AMD GPUs for high-performance stencil computing applications. In summary, stencil computing is a numerical technique used to compute values at a point in a grid based on arbitrary number of neighbouring points, in a repetitive manner.

This paper is structured as follows. Chapter 2 discusses the architecture of GPUs. NVIDIA and AMD GPU architectures are examined in more detail, and the key differences between these platforms are summarized. Chapter 3 overviews the impact of GPU architecture on stencil calculations. Finally, Chapter 4 provides a review and conclusion of the

paper.

2 Graphics Processing Unit Architecture

The NVIDIA Tesla A100 and AMD MI100 are both high-performance GPUs designed for data center and high-performance computing applications. The Tesla A100 is based on NVIDIA's Ampere architecture, while the MI100 is based on AMD's CDNA architecture. The A100 and MI100 were released on May 21 and November 16 on 2020, respectively, which makes them prime candidates for comparison. As one might expect, the architectures share many similarities between them.

2.1 Similarities in Architecture Design

Both GPUs are built on the 7 nm lithography process technology and use modern high-bandwidth memory (HBM2) controllers, which allows extremely high memory bandwidth of 1.555 (A100) [8] and 1.23 (MI100) [2] terabytes per second. In addition, the Tesla A100 utilizes NVLink technology, which provides a direct link between the multiple GPUs. This connection is faster than a PCIe connection and effectively allows multiple GPUs to be used as a single processing unit [6]. The AMD CDNA architecture provides a similar technology known as Infinity Fabric [2]. The main distinction between the two being that Infinity Fabric is implemented using a chiplet design, which allows for simple scalability and modularity, while NVLink is implemented using a dedicated physical connector that is integrated into the GPU package. AMDs Infinity Fabric is also utilized in CPUs and accelerators. Thus, NVLink that was designed specifically for linking multiple GPUs, provides a higher total link bandwidth when comparing the A100s 600GB/s [8] to the MI100s 276GB/s [2] theoretical maximum peer-to-peer bandwidth. However, the single link transport rate bandwidth is higher on the MI100. A single MI100 contains three links, each with 92 GB/s bandwidth. In contrast, the A100 has 12 links, each yielding 50GB/s bandwidth.

Tesla A100 and MI100 GPUs incorporate specialized hardware components and instructions to accelerate complex data processing and compute-intensive workloads. For example, the NVIDIA Ampere architecture features third-generation Tensor Cores [8], which accelerate matrix multiplication operations commonly used in deep learning workloads, while the

AMD CDNA architecture incorporates Matrix Core technology for matrix operations [2]. In addition, the architectures support a range of programming models and software tools for high-performance computing, such as CUDA for NVIDIA and HIP for AMD. They also support popular machine learning and deep learning frameworks, such as TensorFlow, PyTorch and MXNet.

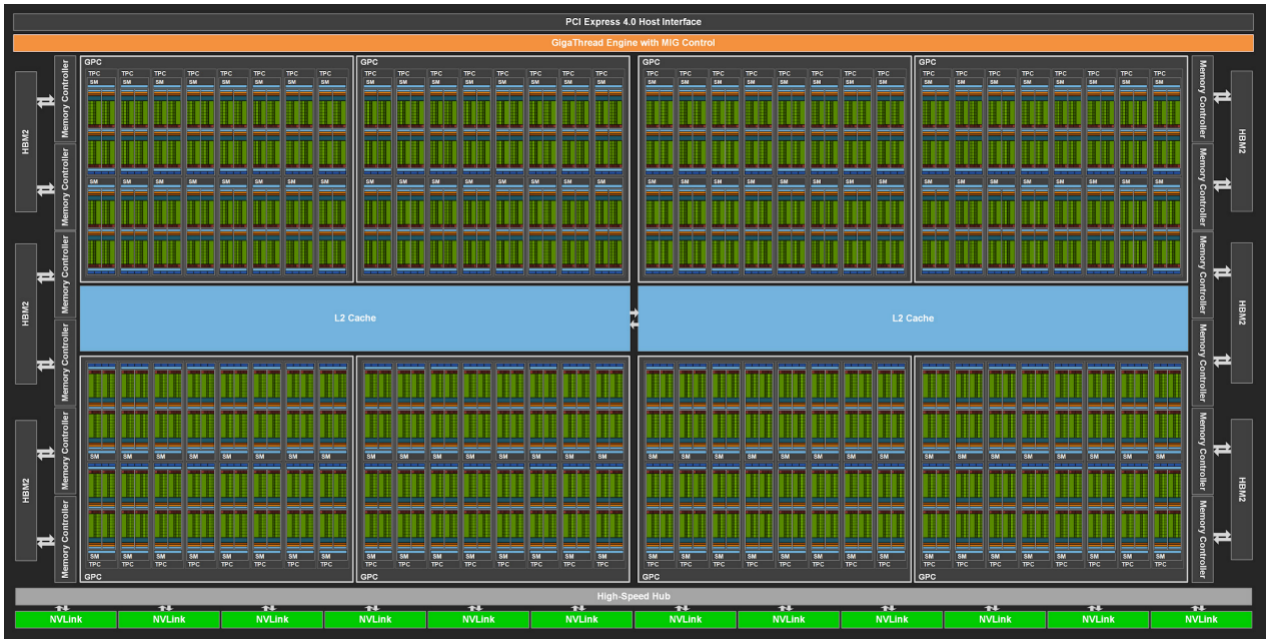


Figure 1. GA100 Full GPU with 128 SMs [8]

Figure 1 shows the full GA100 GPU design overview. A single unit consists of multiple Graphics Processing Clusters (GPC) that include multiple Streaming Multiprocessors (SM), Level 2 Caches, Memory Controllers and HBM2 memory. The unit also has NVLink as mentioned previously. The AMD MI100 has broadly speaking a similar structure, with different technologies, such as Infinity Fabric instead of NVLink. However, there are notable differences, such as Asynchronous Compute Engines (ACE), Hardware Scheduler (HWS) and Direct Memory Access (DMA) components, but discussion about these components is out of the scope of this paper.

2.2 NVIDIA Ampere Architecture

One notable difference between NVIDIA and AMD GPU architectures is how the processing cores are designed. The Ampere architecture has a multi-level cache hierarchy that includes L1, L2, and L3 caches, as well as a HBM module. NVIDIA has combined L1 data cache and shared memory into one. This was introduced with the Volta architecture in 2017 [7].

Shared memory is memory that can be accessed by multiple processing units simultaneously, such as threads within a GPU. Processing core and memory layout for the A100 is illustrated in Figure 1.

Figure 2. GA100 Streaming Multiprocessor (SM) [8]

In NVIDIA GPUs, a warp consists of 32 threads, which execute in lock-step. This means that all 32 threads execute the same instruction at the same time, which allows for efficient execution of parallel code as long as all threads within a warp follow the same path. In addition, Volta architecture introduced independently schedulable threads. This is because the A100's streaming multiprocessors (SMs) use a "single-instruction, multiple-thread" (SIMT) execution model. Under the SIMT model every single thread has its own program counter and stack, allowing the program execution to diverge and become task parallel.

2.3 AMD CDNA Architecture

On the compute unit level, the architectures begin to differ more distinctively. Unlike in Ampere architecture, each wavefront in AMD CDNA architecture has its own program counter, but individual threads within a wavefront do not have their own program counter. A wavefront is a group of threads that are executed together on a single compute unit (CU) [1]. A wavefront consists of 64 threads that execute in groups of 16

threads called a wavefront quad. Within a wavefront, all threads execute the same instruction in lockstep, thus they all follow the same program counter. The program counter is a register that keeps track of the address of the next instruction to be executed, and it is updated as the wavefront progresses through the program.

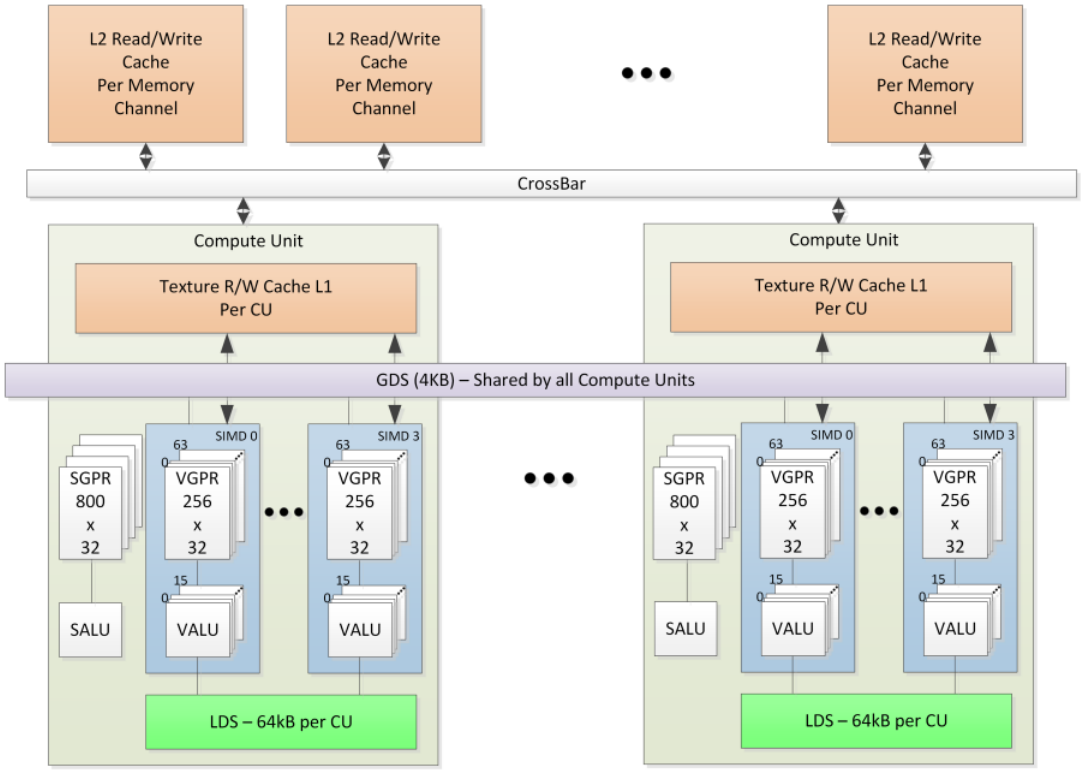


Figure 3. Shared Memory Hierarchy of AMD MI100 [1]

Figure 2 shows the shared memory hierarchy of the AMD MI100. Each compute unit has 64 kB of local data share (LDS) memory. Unlike, with Ampere architecture, the L1 Cache and shared memory are separate memory units.

3 The Impact of GPU Architecture on Stencil Calculations

Multinode GPU computing is a computing process that involves connecting multiple GPUs across different nodes to work together in parallel, with the aim of achieving higher performance gains than by single-node GPU computing. Stencil computations are ideal candidates for porting to the formidable GPU architecture because of the simple data structures and frequent memory access [5]. However, one important factor that can greatly impact the performance of multinode GPU computing is the com-

munication time delay between the GPUs. The authors of [11] discuss the use of multinode computing for high-order stencil computations using CUDA-aware MPI. The article explores the use of MPI to improve the scalability of stencil computations, which are widely used in scientific and engineering applications. The authors use CUDA-aware MPI to enable efficient communication between GPUs hosts, which allows for high-performance computing across multiple nodes. By using this approach, the authors demonstrate that they can achieve excellent scalability and performance for high-order stencil computations.

3.1 CUDA-MPI Astaroth

The CUDA-MPI library discussed in the previous paragraph is called Astaroth. The authors used Astaroth to enhance the performance of stencil computations [11]. Astaroth is an open-source library that is specifically designed for high-order stencil computations with a focus on the demands of multiphysics applications on modern high-performance computing (HPC) systems [12]. It offers both multi-GPU and single-GPU APIs, as well as a domain-specific language that enables stencil codes to be expressed in a high-level syntax. In addition, Astaroth includes an optimizing compiler that translates DSL sources into CUDA/HIP subroutines that exhibit performance that is similar to what would be achieved with hand-tuning. The authors of [11] state that their implementation achieved 20-60 times speedup and 9-12 times improved energy efficiency in compute-bound benchmarks on 16 nodes, when compared to a multi-core GPU solver.

3.2 GPU Memory

The performance of stencil calculations can be significantly impacted by the underlying GPU architecture. The memory hierarchy of the GPU is one key aspect. GPUs with greater on-chip memory and effective memory controllers can perform better by freeing up more memory bandwidth. The processing power of the GPU should also be taken into account. While stencil calculations can be parallelized across many GPU cores, where the memory bandwidth poses the largest bottleneck, GPUs with more cores or more powerful core architectures will perform better. Additionally, performance of stencil calculations can be further enhanced by GPUs with specialized compute units for particular operations, such as tensor cores

for matrix operations.

High performance on GPUs may be achieved in part through modifying thread granularity and utilizing data locality in on-chip storage [13], since stencil calculations require accessing neighboring data points. Storing data in a contiguous block of memory can achieve data locality. This allows for efficient access to neighboring data points without constantly fetching data from off-chip memory. This means memory bandwidth requirements can be significantly reduced, which improves performance. Data locality can also reduce the amount of data that is transferred between memory levels by minimizing cache misses and by maximizing data reuse.

4 Conclusion

From the previously discussed sections it can be concluded that both the Tesla A100 and AMD MI100 share many desirable characteristics for parallel stencil calculations. When examining just a singular GPU both of the models seem tied on the front of performance. However, since communication speed plays a crucial role in scaled stencil calculations, it could be argued that as the utilized number of GPUs increases the A100 would have a smaller drop in performance due to the good scalability of NVLink. On the contrary, the MI100 might perform better with lower number of GPUs due to the higher bandwidth on a singular link.

A important aspect to also consider is the price of the GPUs. The MSRP of the A100 was higher than that of the MI100 and it is still on average more expensive than the MI100. It shall be seen if the price difference persists as the price of GPUs in general has been quite volatile over the recent years. In addition, the AMD MI100 also comes cheaper in terms of the electricity bill as the typical power consumption is rated lower.

This paper has provided an overview of the architectures of NVIDIA Tesla A100 and AMD MI100, and evaluated their relative performance and role in high-performance stencil computing. Since both GPUs provide good performance on paper, more research and testing into the topic is needed in order to decide which GPU is more suited for this context.

References

[1] AMD. "AMD Instinct MI100" Instruction Set Architecture Reference Guide, 2020. Visited 20.2.2023. URL: https://developer.amd.com/wp-content/resources/CDNA1_Shader_ISA_14December2020.pdf.

[2] Advanced Micro Devices. AMD CDNA Architecture White Paper, 2020. Visited 2.2.2023. URL: https://www.amd.com/system/files/documents/amd-cdna-whitepaper.pdf.

[3] Advanced Micro Devices. ROCm Languages, HIP: Heterogeneous-Computing Interface for Portability, 2022. Visited 2.2.2023. URL: https://rocmdocs.amd.com/en/latest/Programming_Guides/LanguageInto.html.

[4] Khronos OpenCL Working Group. The OpenCL Specification, Version v3.0.12, 2022. Visited 2.2.2023. URL: https://registry.khronos.org/OpenCL/specs/3.0-unified/pdf/OpenCL_API.pdf.

[5] John Nickolls and William J. Dally. The gpu computing era. *IEEE Micro*, 30(2):56–69, 2010. URL: https://ieeexplore.ieee.org/document/5446251.

[6] NVIDIA. GP100 Pascal Whitepaper, 2016. Visited 16.2.2023. URL: https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf.

[7] NVIDIA. NVIDIA TESLA V100 GPU ARCHITECTURE, 2017. Visited 17.2.2023. URL: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

[8] NVIDIA. NVIDIA A100 Tensor Core GPU Architecture, 2020. Visited 16.2.2023. URL: https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.

[9] NVIDIA. CUDA C++ Programming Guide, Release 12.0, 2023. Visited 2.2.2023. URL: https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

[10] Johannes Pekkilä. Astaroth: A Library for Stencil Computations on Graphics Processing Units. Master's thesis, Aalto University. School of Science, 2019. URL: http://urn.fi/URN:NBN:fi:aalto-201906233993.

[11] Johannes Pekkilä, Miikka S. Väisälä, Maarit J. Käpylä, Matthias Rheinhardt, and Oskar Lappi. Scalable communication for high-order stencil computations using cuda-aware mpi. *Parallel Computing*, 111:102904, 2022. URL: https://www.sciencedirect.com/science/article/pii/S0167819122000102.

[12] Finland ReSoLVE Centre of Excellence, Espoo. Astaroth Repository, 2023. Visited 15.3.2023. URL: https://bitbucket.org/jpekkila/astaroth/src/master/.

[13] Huayou Su, Xing Cai, Mei Wen, and Chun-yuan Zhang. An analytical gpu performance model for 3d stencil computations from the angle of data traffic. *The Journal of Supercomputing*, 71:2433–2453, 02 2015. URL: https://www.researchgate.net/publication/276831218_An_analytical_GPU_performance_model_for_3D_stencil_computations_from_the_angle_of_data_traffic.

Dynamics of social interactions in social Mixed Reality

Touko Nurminen
touko.nurminen@aalto.fi

Tutor: Robin Welsch

Abstract

Mixed Reality (MR) technology has emerged as a promising tool for studying social interactions, offering a controlled and immersive environment for simulating various social situations. This paper provides a comprehensive review of the latest research on social interaction behavior within mixed reality environments. We discuss the development of MR technology in the context of social interaction research, the role of non-verbal behavior such as gaze and expression in social interaction research with MR, and the challenges associated with traditional social interaction research methods. Additionally, we explore the current limitations and future directions of MR technology in the context of social interaction research. Despite existing challenges, advancements in MR technology hold great promise for the field of social interaction research, offering valuable insights into human communication and interaction in the digital era.

KEYWORDS: Mixed Reality, Virtual Reality, Augmented Reality, Social Psychology, Social Interaction, Non-verbal Behavior

1 Introduction

In psychological science, social interaction has been studied with various methods to gain insight into topics such as communication, relationships, and cultural differences [18]. Questionnaires and self-reports, though frequently used, are not the most ecologically valid research methods for examining social interaction, since they rely on the participants' ability to accurately recall and report their behavior [8]. One of the dominant research methods for studying social interaction is behavioral observation, which allows for the direct observation of individuals within a social context [13]. However, this method presents its own set of challenges, including identifying suitable experimental settings, the inability to replicate test scenarios, and accounting for variability of traits among human subjects [10].

To combat this, researchers have begun simulating social interactions using Mixed Reality (MR) technology. A controlled, low-cost environment is achieved by embedding digital content, such as 3D objects and virtual agents into a virtual environment [16]. A real human participant can then interact with the virtual agent within the virtual environment and the interaction and its dynamics can be observed to gain insight into social interaction behavior.

This paper reviews the latest research on studying social interaction using MR technology, and discusses using MR to solve the issues regarding social interaction research. Additionally, the paper discusses the roles of various aspects of non-verbal behavior, and simulating those to enhance social interaction research. The goal of this paper is to review the state of MR technology in social interaction research and its potential for simulating non-verbal behavior, along with discussing the limitations and challenges faced in this field.

The paper is organized as follows. Section 2 describes the development of MR technology in the context of social interaction research. Section 3 presents the use of MR technology in social interaction research, including discussing the role of simulating non-verbal behavior such as gaze and expression, and additionally explores the challenges of MR technology in social interaction research. Section 4 discusses the future of MR technology in social interaction research. Finally, Section 5 provides concluding remarks by reviewing the results of the paper.

2 Background

MR technology has been used in a variety of fields, including entertainment, education, and healthcare [15, 17]. In recent years, it has also gained attention as a tool for studying social interaction and non-verbal behavior [11]. One of the main advantages of MR technology in this context is its ability to create a low-cost environment that allows for the manipulation of various social factors, such as gaze and facial expressions [6, 9]. This creates a controlled research environment in which repeatable social scenarios can be simulated, yielding reliable and consistent test results.

Another challenge that MR technology can solve is the lack of replication, meaning that it is difficult to exactly replicate the same social situation across multiple test instances with human subjects [10]. With MR technology, a social situations can be simulated multiple times while allowing an immersive and interactive experience, thus providing repeatable and reliable test results [9].

As a result, MR technology has gained popularity as a tool for researchers studying social interaction. Next, this section discusses the specifics of non-verbal behavior and MR technology, along with the development of MR technology in the context of social interaction behavior research.

2.1 Non-verbal behavior: An Overview

Non-verbal behavior refers to all forms of communication that do not use words to convey meaning [4]. It includes cues such as body language, facial expressions, eye contact, gestures, and posture [20]. These cues can complement language to convey a wide range of information, such as a person's emotions, intentions, and personality traits. They significantly impact social interactions, which is why non-verbal behavior is a crucial part of communication. [14]

Non-verbal communication is a complex phenomenon, influenced by various factors such as culture and context. For instance, the meaning of a gesture may differ across cultures, and the same gesture can have different meanings depending on the situation. According to Matsumoto [21], people use different levels of interpersonal space in the same situation depending on the culture that they belong in. For example, Arab people often sit closer to each other than Americans.

This paper focuses on two aspects of non-verbal behavior: gaze and ex-

pression. Gaze refers to eye contact and the direction of where the eyes are looking, and their role in communication [5]. Expression refers to the facial expressions and cues that people use to convey emotion [21].

Studying non-verbal behavior using traditional research methods can be challenging, since the variability of human subjects can result in inaccurate data. For instance, some people might smile more when they interact with a particular person [18]. These inconsistencies between test subjects' behaviors might lead to inaccurate results. This is where technologies such as MR can provide valuable insights. MR allows researchers to regulate these parameters, and thus observe non-verbal behavior in a controlled environment, providing an accurate and reliable means of studying non-verbal behavior [9].

2.2 Mixed Reality: Definition and Advancements

Mixed reality is a technology that blends physical and digital environments together in real time. It is often used as an umbrella term that includes multiple immersive technologies, such as Augmented Reality (AR) and Virtual Reality (VR). [26] In Azuma et al.'s (1997) survey, AR is defined as a technology that allows the user to see the real world with virtual content overlaid onto it. Additionally, VR is defined as a technology that completely immerses the user in a virtual environment. In Figure 1, the relationships of MR, AR and VR are illustrated in a Reality-Virtuality continuum, as proposed by Milgram and Kishino [23].

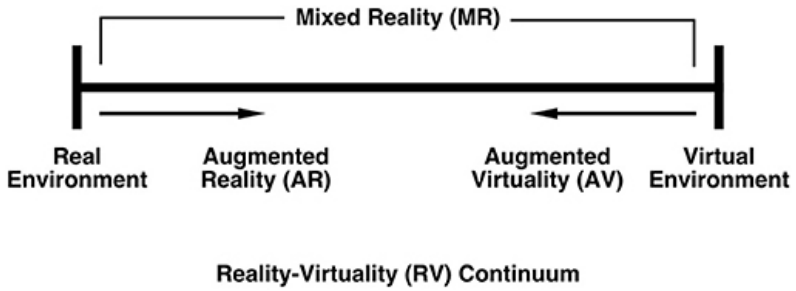


Figure 1. Reality-Virtuality continuum [23]

The advancements in MR technology have been rapid in recent years. One big advancement in connecting the digital world and the real world is the development of spatial computing. Spatial computing enables a computer to perceive and interact with the real world and 3D elements by using technologies such as sensors [7].

Another advancement in MR technology is the development of tracking and sensing technologies. These include motion interactive wearable devices, such as ultrathin skin-like sensors, and proximity and pressure sensing on-skin devices [19]. The sensors allow for tracking of the movements of users in real time to create realistic and interactive MR experiences, while keeping the experience as naturalistic as possible by limiting the amount of bulky equipment the user must wear.

2.3 Evolution of MR Technology in Social Interaction Research

The evolution of MR technology in social interaction research has been fast in the past decades. In 2002, Blascovich et al. [10] discussed the use of virtual environments in social psychology studies. They concluded that virtual environments could be used to solve the typical problems in traditional research methods.

One area of research that has particularly benefited from MR technology is the study of social anxiety. Researchers have used MR to simulate social environments where participants can gradually become more comfortable in social situations. For example, Anderson et al. [1] studied the effects of virtual reality exposure therapy for social anxiety disorder, and found that the therapy produces long-lasting benefits for the condition.

More recently, researchers have begun to explore the use of MR technology for studying group interaction. Shared virtual environments can be created where multiple participants can interact with each other in real time. Moustafa et al. [24] studied small group interaction in social virtual reality. Participants were required to regularly engage with each other in social VR applications. They were connected to the virtual environment via smartphones and head-mounted displays. The researchers found that users engaged in VR can experience stimuli similar to real life, and that social interaction in VR was directly comparable to that in real life, and that it evokes genuine affective responses.

3 Simulating non-verbal behavior with MR

As established thus far, the use of MR technology offers a novel approach for studying social interaction. Researchers have leveraged MR technology to create immersive, interactive environments where the non-verbal behavior of the virtual agents can be standardized [11]. The non-verbal

behavior includes the social cues gaze and expression, which can be simulated with various methods.

3.1 Gaze and Eye Contact in Virtual Environments

Gaze is an important aspect of non-verbal communication, as it plays a crucial role in conveying information about a person's attention, interest, and emotions [14]. Gaze is used in both directions in social interaction: indicating one's attention and sensing the attention of others [2]. In traditional research methods, measuring gaze can be difficult, as it requires precise tracking of eye movements. With MR technology, the gaze behavior of the virtual agents can be regulated, and variables such as matching the eye heights of the participants can be controlled [6]. This enables researchers to study gaze behavior in a controlled setting to conduct research with regulated parameters and should thus provide more accurate results. Additionally, simulating gaze of the virtual agents in virtual environments is important to achieve a more immersive virtual experience. This allows the participants to feel as if they are interacting with a real human, and thus creating a better environment for social interaction research.

For example, in a recent study by Narang et al. [25], a novel interactive approach called PedVR was proposed for simulating gaze and gestural behaviors of virtual agents in virtual environments. PedVR combines 2D navigation with full-body motion synthesis and gaze computation to provide users with the ability to interact with a virtual crowd. The virtual agents were able to compute collision-free trajectories to avoid the user and other virtual agents. In addition, the believability of the virtual experience was greatly increased due to the virtual agents' capability to exhibit gaze and gestural behaviors. The study showed a fourfold increase in user preference for the approach with the introduction of gaze, which demonstrates the importance of simulating gaze of the virtual agents in virtual environments.

However, there are also challenges that need to be addressed. For example, participants might not respond to virtual gaze and eye contact as they would in real life scenarios, which could limit the validity of the research. Moreover, if participants are aware that the eye contact they are experiencing is virtual, this could reduce their sense of presence and engagement in the environment, which would further undermine the validity of the research.

3.2 Expression in MR Environments

Expression holds significant importance in social interactions, as it allows individuals to convey their feelings and intentions to others. Facial expressions, body posture, and gestures are among the primary means through which people express emotion non-verbally [3]. Thus, simulating realistic emotional expressions in virtual agents is essential when creating MR environments for social interaction research.

Researchers have employed various techniques to simulate emotional expressions. One approach is to use the Facial Action Coding System (FACS) when creating facial expressions for virtual agents. FACS categorizes facial movements based on the action of individual facial muscles to 58 action units. The action units can be mapped onto a virtual agent's face, and through animation, they can be employed to characterize any facial expression. [12]

In a recent study by Tisserand et al. [27], an interactive tool was developed to simulate virtual humans' emotional facial expressions in real-time, considering both musculoskeletal and Autonomic Nervous System (ANS) features to increase expressive realism. The tool allows users to control ANS parameters and thirty action units for expression control, making it possible to create a wide range of facial expressions. This approach can help enhance the believability and realism of virtual agents' emotional expressions, leading to more realistic social interactions in MR environments.

A challenge in simulating expression in MR environments is the accurate recognition of emotions in virtual agents. In real-world social interactions, people rely on a combination of verbal and non-verbal cues to recognize and interpret others' emotions [4]. However, this can be more difficult in MR environments, where virtual agents may not exhibit the full range of emotional expressions or may display emotions that are less recognizable to users.

3.3 Limitations and Challenges of MR in Social Interaction Research

Despite the numerous benefits of using MR technology in social interaction research, there are several limitations and challenges to be addressed. These include:

Lack of sense of presence: One of the main challenges faced when us-

ing MR technology in social interaction research is the lack of a sense of presence experienced by the participants. Sense of presence refers to the extent to which a person feels that they are actually present in a social situation in the virtual environment. It can be difficult to create virtual agents that closely resemble real humans, which can negatively affect the virtual experience. [9] Since social interaction research is about gaining insight into human behaviour [4], it is important to create believable virtual environments and virtual agents when using MR technology for the research.

Uncanny valley phenomenon: Another challenge faced in social interaction research with MR is the uncanny valley phenomenon. The phenomenon occurs when virtual agents appear almost, but not quite human. This can lead to a sense of unease or discomfort for participants interacting with the virtual agents. [22] As a result, since the goal is to study human behaviour, the uncanny valley phenomenon can negatively impact the ecological validity of the study and thus make the findings less valid.

4 Discussion and Future Directions

In this paper, we have highlighted the potential of MR technology for simulating social interactions and non-verbal behavior. In the future, there are several possible research avenues and technological advancements that can further enhance the effectiveness of MR in the context of social interaction research.

Integration of AI and machine learning: By incorporating state-of-the-art AI techniques, virtual agents can iteratively learning human behaviour, resulting in them being more human-like. This will improve the believability of the virtual agents, and thus will help to create better research environments.

Improved sensory feedback: To enhance the immersion in MR environments, future developments could focus on providing users with more realistic sensory feedback, such as haptic cues. This will allow the participants to feel more present in the virtual environment, since the sensory feedback will feel more like that of real life. Additionally, the participants' reactions to the sensory feedback could be monitored to further gain insight into the interactions.

Exploration of cross-cultural differences: As MR technology becomes prevalent in social interaction research, researchers can utilize it to study

cross-cultural differences in social interactions. With the ability to simulate social scenarios with cultural variations, researchers can gain valuable insights into how culture affects our ways of understanding and interpreting non-verbal cues.

Ethical considerations: As virtual environments created with MR technology become more believable and realistic, it is essential to address ethical concerns related to privacy, consent, and potential psychological effects on the participants. Guidelines should be established to ensure that the use of MR technology in social interaction research is responsible and ethical.

5 Conclusion

This paper has provided a comprehensive review of the use of MR technology in social interaction research and simulating non-verbal behavior. We have discussed the advantages of MR technology in creating controlled research environments, where repeatable social scenarios can be simulated. Additionally, we have discussed the potential of MR for simulating various aspects of non-verbal behavior, such as gaze and expression. Furthermore, we have examined the limitations and challenges faced when using MR technology in social interaction research, including the lack of a sense of presence and the uncanny valley phenomenon.

As the field of MR technology continues to advance rapidly, it holds great promise as a tool for improving our understanding of social interaction and non-verbal communication. The future of MR technology in social interaction research will likely involve exploring ways to overcome current limitations and enhance the realism of virtual agents and environments. By addressing these challenges and leveraging the unique capabilities of MR technology, researchers will be better equipped to gain valuable insights into human communication and interaction.

References

[1] Page L Anderson, Shannan M Edwards, and Jessica R Goodnight. Virtual reality and exposure group therapy for social anxiety disorder: Results from a 4–6 year follow-up. *Cognitive Therapy and Research*, 41:230–236, 2017.

[2] Sean Andrist, Michael Gleicher, and Bilge Mutlu. Looking coordinated: Bidirectional gaze mechanisms for collaborative interaction with virtual characters. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 2571–2582, 2017.

[3] Michael Argyle. Non-verbal communication in human social interaction. *Non-verbal communication*, 2, 1972.

[4] Michael Argyle. *Social interaction*. Routledge, 2017.

[5] Michael Argyle and Janet Dean. Eye-contact, distance and affiliation. *Sociometry*, pages 289–304, 1965.

[6] Jeremy N Bailenson, Jim Blascovich, Andrew C Beall, and Jack M Loomis. Equilibrium theory revisited: Mutual gaze and personal space in virtual environments. *Presence: Teleoperators & Virtual Environments*, 10(6):583–598, 2001.

[7] S Balakrishnan, M Syed Shahul Hameed, K Venkatesan, and G Aswin. Interaction of spatial computing in augmented reality. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 1900–1904. IEEE, 2021.

[8] Roy F. Baumeister, Kathleen D. Vohs, and David C. Funder. Psychology as the science of self-reports and finger movements: Whatever happened to actual behavior? *Perspectives on Psychological Science*, 2(4):396–403, 2007. PMID: 26151975.

[9] Elisabetta Bevacqua, Romain Richard, and Pierre De Loor. Believability and co-presence in human-virtual character interaction. *IEEE computer graphics and applications*, 37(4):17–29, 2017.

[10] Jim Blascovich, Jack Loomis, Andrew C Beall, Kimberly R Swinth, Crystal L Hoyt, and Jeremy N Bailenson. Immersive virtual environment technology as a methodological tool for social psychology. *Psychological inquiry*, 13(2):103–124, 2002.

[11] Dario Bombari, Marianne Schmid Mast, Elena Canadas, and Manuel Bachmann. Studying social interactions through immersive virtual environment technology: virtues, pitfalls, and future challenges. *Frontiers in psychology*, 6:869, 2015.

[12] Marc Fabri, David Moore, and Dave Hobbs. Mediating the expression of emotion in educational collaborative virtual environments: an experimental study. *Virtual reality*, 7:66–81, 2004.

[13] R Michael Furr and David C Funder. Behavioral observation. *Handbook of research methods in personality psychology*, pages 273–274, 2007.

[14] Randall P Harrison. Nonverbal communication. *Human Communication As a Field of Study: Selected Contemporary Views*, 113:16, 1989.

[15] Hong-zhi Hu, Xiao-bo Feng, Zeng-wu Shao, Mao Xie, Song Xu, Xing-huo Wu, and Zhe-wei Ye. Application and prospect of mixed reality technology in medical field. *Current medical science*, 39:1–6, 2019.

[16] Ann Huang, Pascal Knierim, Francesco Chiossi, Lewis L Chuang, and Robin Welsch. Proxemics for human-agent interaction in augmented reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2022.

[17] Charles E Hughes, Christopher B Stapleton, Darin E Hughes, and Eileen M Smith. Mixed reality in education, entertainment, and training. *IEEE computer graphics and applications*, 25(6):24–30, 2005.

[18] David A Kenny. The design and analysis of social-interaction research. *Annual Review of Psychology*, 47(1):59–86, 1996.

[19] Hojoong Kim, Young-Tae Kwon, Hyo-Ryoung Lim, Jong-Hoon Kim, Yun-Soung Kim, and Woon-Hong Yeo. Recent advances in wearable sensors and integrated functional devices for virtual and augmented reality applications. *Advanced Functional Materials*, 31(39):2005692, 2021.

[20] Mark L Knapp, Judith A Hall, and Terrence G Horgan. *Nonverbal communication in human interaction*. Cengage Learning, 2013.

[21] David Matsumoto. Culture and nonverbal behavior. *The SAGE handbook of nonverbal communication*, pages 219–235, 2006.

[22] Ryan P McMahan, Chengyuan Lai, and Swaroop K Pal. Interaction fidelity: the uncanny valley of virtual reality interactions. In *Virtual, Augmented and Mixed Reality: 8th International Conference, VAMR 2016, Held as Part of HCI International 2016, Toronto, Canada, July 17-22, 2016. Proceedings 8*, pages 59–70. Springer, 2016.

[23] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–292. Spie, 1995.

[24] Fares Moustafa and Anthony Steed. A longitudinal study of small group interaction in social virtual reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pages 1–10, 2018.

[25] Sahil Narang, Andrew Best, Tanmay Randhavane, Ari Shapiro, and Dinesh Manocha. Pedvr: Simulating gaze-based interactions between a real user and virtual crowds. In *Proceedings of the 22nd ACM conference on virtual reality software and technology*, pages 91–100, 2016.

[26] Maximilian Speicher, Brian D Hall, and Michael Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15, 2019.

[27] Yvain Tisserand, Ruth Aylett, Marcello Mortillaro, and David Rudrauf. Real-time simulation of virtual humans' emotional facial expressions, harnessing autonomic physiological and musculoskeletal control. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*, pages 1–8, 2020.

Using formal verification with instant messaging protocols

Uuna Saarela
uuna.saarela@aalto.fi

Tutor: Lachlan Gunn

Abstract

Instant messaging is a popular way to keep in touch with family, friends, and colleagues. In some cases, instant messaging apps are also used to communicate information to large audiences. This means they can be sites of public life in addition to private life. They are typically also used to communicate sensitive or personal information, making them lucrative targets for attackers.

Formal verification can provide security assurances regarding instant messaging protocols. Interestingly, instant messaging protocols also provide useful material for formal verification tools to be developed against. This paper provides an overview of the use of formal verification and related tools to support protocol design for instant messaging.

KEYWORDS: instant messaging, formal verification, MTProto 2.0, Signal, WhatsApp, Facebook Messenger

1 Introduction

Instant messaging is a type of communication technology which facilitates synchronous, low-latency transmission of various media such as text or images over a network, typically the Internet. Instant messaging is in-

tegrated into many social media and e-commerce platforms, as well as being available in dedicated applications on both desktop and mobile devices. Various instant messaging applications, or chat applications, are estimated to be used by over three billion people worldwide, with instant messaging applications being among the most used smartphone applications. These applications are used in in personal, public, and professional contexts alike.

Because instant messaging is such a ubiquitous part of modern life, there is rising concern about its security as a communication channel. Instant messaging applications are often used to communicate sensitive and personal information, making them lucrative targets for attackers or state surveillance. End-to-end encryption has emerged as a central standard for secure instant messaging. End-to-end encryption refers to the content of an instant message being encrypted via the entire transit from one client device to another. In particular, messages cannot be decrypted by a local or cloud server storing or routing messages belonging to the service provider. The Electronic Frontier Foundation (EFF), a non-profit organisation that promotes digital privacy, featured end-to-end encryption as an important measure in their original Secure Messaging Scorecard.[1]

Pressure from users to has lead to the rapid deployment of new end-to-end encryption features, leading to a mixed landscape of actual security for the end user. End-to-end encryption has been adopted by most popular instant messaging applications, with the net result being that most instant messaging applications are now end-to-end encrypted. However, "[m]any products mislead users by advertising with grandiose claims of “military grade encryption” or by promising impossible features such as self-destructing messages" and "many purportedly “secure” tools do not even attempt end-to-end encryption".[22] Even when cryptographic protocols are implemented in good faith, they can be difficult to construct correctly. "[E]even well-studied protocols, such as Transport Layer Security (TLS), have been shown to contain serious protocol flaws found years after their deployment."[15] How then can we gain confidence regarding encrypted instant messaging as an industry standard?

Formal verification may be a useful tool for this purpose. It is a framework which provides the possibility to obtain proofs of correctness of protocols, algorithms, and software. Cryptographic protocols may be tested against a formal specification of intended security properties or functionality. Formal models can help to identify design flaws in protocols or pro-

vide a concrete proof of security for a complex stack.

Formal methods have been deployed for the purposes of security assurances by many popular instant messaging providers, including Signal[9] and Telegram[17]. This paper provides an overview of the use of formal verification to provide confidence on the security of instant messaging. The paper is organised as follows; in Section 2 we will provide background information on the instant messaging protocol stack, popular cryptographic protocols, and formal verification tools. In Section 3, we will explore the research on formally verifying security properties provided by the cryptographic layer of instant messaging protocols. In Section 4, we will analyse the usefulness of this approach in terms of security for the end user. In section 5, we provide some concluding remarks.

2 Protocols and Tools

2.1 The Instant Messaging Protocol Stack

The instant messaging protocol stack is built on top of standard Internet messaging protocols. Features such as asynchronous messaging are implemented by the message layer. This specialised type of communication is then secured by cryptographic protocols at the message security layer. Instant messaging clients utilise a combination of protocols in the stack. WhatsApp, for example, uses a proprietary variant of the Signal Protocol with XMPP and Noise over HTTP.[3]

Figure 1. The instant messaging protocol stack.

The Transport Layer

Instant messengers typically utilise TCP/IP at the transport layer. An alternative transport protocol is SMS , which transmits messages over a cellular network. The limitations of the SMS framework include being limited to 140 characters, needing both parties to be online simultaneously to establish a secure session, and costs for end users.[14] Although unencrypted communication via SMS is still supported by some instant messaging applications, such as Signal, the majority of messengers work over TCP/IP.

The Cryptographic Layer

Often, instant messaging clients use cryptographic protocols like TLS or Noise to secure communication over the network.[3][13] However, these protocols alone do not provide security features for asynchronous messaging protocols. For example, TLS requires an interactive handshake to initiate communication. In an instant messaging context, we cannot count on both parties being online at the same time, but should still be able to deliver encrypted messages.

The Application Layer

At the application layer, instant messaging clients may use a variety of protocols to transmit messages. These include HTTP(S) and WebSocket[13]. This layer need not be used; some protocols at the message layer can work directly over SMS or TCP.

The Message Layer

At the message layer, instant messaging protocols specify how messages are packaged and communicated. One such messaging protocol is XMPP (eXtensible Message and Presence Protocol), an open-source XML-based messaging protocol introduced in 1999 and made an IETF standard in 2004.[12] XMPP is used by WhatsApp, Kik Messenger, Zoom, and a number of smaller open-source clients such as Conversations. [25] Another open messaging standard is Matrix.org or [matrix] which uses JSON rather than XML.[16] The Matrix.org protocol is not a messaging protocol per se. Rather, it provides a database of JSON objects to facilitate interoperation between different instant messaging clients.

The Message Security Layer

Cryptographic protocols at the message security layer manage keys and encryption of messages. One of the most popular protocols for this purpose

is the Signal Protocol, which refers both to a specific suite of protocols developed by Open Whisper Systems and a more abstract description of a cryptographic message protocol with certain components.

The Signal Protocol was originally conceived as an implementation of an XMPP extension called the OTR (Off-the-record) protocol over SMS.[12] The OTR protocol implemented perfect forward secrecy using ephemeral key exchange, which can be described as a public key ratchet. Another inspiration for the Signal Protocol was the symmetric key ratchet in SCIMP (Silent Circle Instant Message Protocol).[24] These two "ratchets" from the OTR and SCIMP protocols were combined in Signal to form the Double Ratchet protocol. More generally, the Diffie-Hellman key agreement protocol, the Double Ratchet for key management, and a protocol for message encryption and decryption can be together be referred to as the Signal Protocol. Notable implementations of the Signal Protocol include Olm by Matrix.org[23], an XMPP extension called OMEMO[19], Signal's own protocol suite, as well as proprietary variants developed for WhatsApp[3] and Viber[2].

The Signal Protocol has been widely adopted and can be considered an unofficial industry standard among clients that offer encrypted communication. A notable exception to this is Telegram, which uses MTProto 2.0, a custom cryptographic protocol. Telegram is a popular instant messenger which advertises speed, seamless synchronisation across devices, and security. MTProto is Telegram's custom cryptographic protocol which is designed to facilitate high-speed delivery and reliability on weak connections.[21] MTProto 1.0 was widely criticised and shown to have numerous security issues, such as lack of forward secrecy and vulnerabilities to replay attacks.[17] MTProto 2.0 represents the current redesigned protocol with several security patches. There are two distinct protocols for standard client-server / server-client encryption and end-to-end encryption between users. The developers claim the standard protocol is IND-CCA and INT-CTXT secure and that secrets chats ensure perfect forward secrecy.[17] End-to-end encryption is a feature which can be enabled for chats between at most two parties.[21]

2.2 Formal Verification Tools

Formal verification refers to the process of modelling protocols and verifying that they meet given security properties. Formal verification can be performed in either a symbolic or computational model. The symbolic

model is based on perfect black-box cryptograhic primitives, such as hash functions that never collide. In the computational model, cryptographic primitives are functions over bitstrings and the security of a given function is given as a probability.[15] In general, symbolic models are useful for finding attacks on logical flaws in the protocol, while computational models provide a concrete cryptographic proof of security.

Although formal verification can be performed manually, there is a diverse set of formal verification tools available. These tools are useful because they standardise the modelling approach and automate the verification process. In the literature on instant messaging protocols, the primary tools used are ProVerif, CryptoVerif, and Tamarin. ProVerif[7] and CryptoVerif[6] are open-source tools developed by Bruno Blanchet for the purposes of formally verifying cryptographic protocols. ProVerif is a symbolic verifier and CryptoVerif is a computational verifier. Tamarin[20] is another tool for verification in the symbolic model.

3 Formally verifying security properties

In this section, we will examine how formal verification tools have been deployed to prove desirable security properties such as integrity, confidentiality, authenticity, perfect forward secrecy, and reliability. We will include a review of the literature formally modelling and verifying the Signal Protocol, MTProto 2.0, and group messaging protocols.

3.1 The Signal Protocol

Cohn et al.[9] model the Signal Protocol in the computational model manually. They find that the Signal Protocol satisfies several standard security properties, such as secrecy and authentication of message keys.

Kobeissi et al. [15] model a variant of the Signal Protocol using ProVerif. They also include an analysis with CryptoVerif of a now-defunct implementation of the Signal Protocol called Cryptocat. This investigation uncovered "several weaknesses, including previously unreported replay and key compromise impersonation attacks"[15] which the authors fix and reverify.

Alwen et al. [5] present a modularization of the Signal Protocol. They posit that a generalized Signal Protocol has three components: a continuous key agreement (CKA), forward-secure authenticated encryption with

associated data (FS-AEAD), and two-input hash function that is a pseudorandom function (PRF-PRNG). They show that these components together provide post-compromise security, which means that once a key is compromised, secure encrypted communication can resume soon after.

The Signal client relies on the Sesame protocol for session management. The protocol was modelled and formally verified in [11] using Tamarin. The authors found that although Signal guarantees post-compromise security as the session level, design choices in Sesame result in a violation of PCS at the conversation level. They propose that the verification tools be developed to facilitate better modelling of such protocols. As models move from abstract to concrete, security assurances also become more practical.

3.2 MTProto 2.0

Miculan and Vitacolonna [17] model Telegram's suite of cryptographic protocols in ProVerif. They show that security properties such as authentication, integrity, confidentiality and perfect forward secrecy are guaranteed by MTProto 2.0. They model the protocol in the classic Dolev-Yao setting, assuming the possibility of malicious servers in particular. The formalization accounts for user behavior, indicating that if users do not check the fingerprints of their shared keys, a MitM attack is possible.

The authors conclude that MTProto 2.0 does not have any logical flaw but that vulnerabilities could arise from implementation flaws, side-channels exfiltration, or incorrect user behavior. In their symbolic model the encryption primitive of MTProto 2.0 is assumed to be a perfect authenticated encryption scheme providing IND-CCA and INT-CTXT security. They suggest that computational models like CryptoVerif or EasyCrypt be deployed to prove that this underlying encryption scheme is sound.

Albrecht et al. [4] examine the cryptographic primitives in MTProto 2.0 in the computational model. They focus on the security features offered by the standard cloud chats instead of the end-to-end encrypted "secret chats" The security model developed in the paper considers various secure channel styles, including the situation where the channel operates over an unreliable transport protocol. The formal model of MTProto developed in the paper considers the cryptographic primitives used in MTProto and studies whether they satisfy the necessary security notions to achieve the desired security of the protocol. The authors prove that their variant of MTProto 2.0 achieves channel confidentiality and integrity in their model under certain assumptions on the components used in its construction.

Vulnerabilities were found and Telegram has implemented the proposed alterations, providing some assurance about the security of the currently deployed MTProto 2.0.

3.3 Group Messaging

Secure group messaging represents a compromise between security and efficiency. Implementing the Signal Protocol between each two members of a group is possible, but computationally expensive. Approaches vary from simply accepting the cryptographic overhead or creating a dedicated group messaging protocol. OMEMO, for example, simply accepts the cost of securing group communication with one-to-one protocols.[19] Matrix.org has implemented a separate library called Megolm to facilitate group messaging.[23] Telegram avoids the problem entirely by only offering end-to-end encrypted communication for at most two parties.

In [18], the authors investigate the landscape of secure group messaging and find several security issues. As the group messaging protocol is not specified by the Signal Protocol itself, how group messaging is conducted is left to the implementation. Three implementations of the Signal Protocol for group messaging are analysed. They conclude that standard security properties as well as strong security properties such as perfect forward secrecy do not hold for group messaging.

Recent research has emerged to solve this issue. [10] presents a new design called Asynchronous Ratcheting Trees (ART) that combines synchronous group messaging techniques with strong security guarantees from asynchronous messaging. ART achieves post-compromise security for group messaging and offers the same security level for groups as for pairwise communications. The design has gained industry interest and has been adopted as a starting point by the IETF Message Layer Security (MLS) working group.

4 Discussion

Unger et al. [22] and Cohn-Gordon and Cremers [8] identify the following key challenges in addition to the correctness of cryptographic protocols in the field of secure messaging.

Trust Establishment

Trust establishment refers to "the process of users verifying that they are actually communicating with the parties they intend."[22] Both the Signal Protocol and MTProto 2.0 are designed to be used in a non-federated context, where a server stores users' prekey bundles under some universal identifier. A malicious server could easily initiate communication with the wrong party or intercept communication with malicious keys. Users of the Signal Protocol clients are advised to compare their public keys in person to achieve authenticated encryption. Other protocols designed for use in a federated context, such as Matrix.org, also assume public key comparison to happen via a separate secure channel. However, users rarely perform these checks. Public key distribution represents a major challenge in the field of secure messaging, and it is the object of recent research by Google and others.[8]

Transport Privacy

Transport privacy refers to "how messages are exchanged, with the goal of hiding message metadata such as the sender, receiver, and conversation to which the message belongs."[22] As the Signal Protocol and MTProto 2.0 are both designed to work with a client / server architecture, they do not meaningfully address transport privacy. Federated message protocols, such as Tor, may provide a model for how to address transport privacy. In recent years, a number of experimental open-source projects providing network-level anonymity have been created. These include Bitmessage, Briar, Ricochet, and Tox.[12] Protecting communication metadata may become an industry standard in the future.

Other Real-World Challenges

According to Cohn-Gordon and Cremers [8], instant messaging service providers often need to support more features than are usually specified in the formal verified protocol models. For example, users often want to have access to backups of their message history. They may also value reliability over security and want the protocol to automatically retry failed messages. Integration with external services enabling GIF search or URL previews also introduce complexity to security concerns. These features are important to users and introduce many variables into the attack surface of protocols, but are rarely accounted for in the existing research.

5 Conclusion

The process of formal verification contributes to the empirical robustness of popular protocols. Several papers have discovered bugs, attacks, or shortcomings in these protocols using formal verification.[17][11] Formal verification tools help to standardize and automate the process of formal verification.

The landscape of instant messaging protocols is diverse and shifting. According to Kobeissi [15], "modern web applications often embed custom cryptographic protocols that evolve with each release." Formal verification in the symbolic model can thus provide an agile framework for cryptography researchers working in this context. Using tools such as ProVerif, they can modularize and model a suite of protocols and automate the verification process. When one or more of the protocols are updated, the existing model can be updated to obtain a new verification of the whole suite.

Formal verification may also benefit from the existence of instant messaging protocols. Over the course of researching for this paper, it became clear that several of the researchers formally verifying instant messaging protocols are also involved in the development of the formal verification tool being used. The landscape of instant messaging protocols presents a diversity of implementations of various open standards and protocols. As such, it presents a fruitful object of study to develop formal verification tools against.

Many proprietary instant messaging service providers claim to implement the Signal Protocol, but this claim cannot be verified. Other popular messengers, such as Line and WeChat, do not implement end-to-end encryption at all. Even when software and protocols are open, the end user may be stuck with an opaque binary from a closed source vendor.[8] The user may also not care or know how to verify the authenticity of the person they are communicating with. Future research and development that addresses these problems, as well as protecting communication metadata, will contribute to private and free communication for all.

References

[1] Secure messaging scorecard.

[2] Viber Encryption Overview. Technical report.

[3] WhatsApp Encryption Overview. Technical report, 01 2023.

[4] Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, and Igors Stepanovs. Four Attacks and a Proof for Telegram. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 87–106, 2022.

[5] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The Double Ratchet: Security notions, proofs, and modularization for the Signal Protocol. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11476 LNCS:129 – 158, 2019. Cited by: 39.

[6] Bruno Blanchet. CryptoVerif: Cryptographic protocol verifier in the computational model. https://bblanche.gitlabpages.inria.fr/CryptoVerif/. [Accessed 12-Apr-2023].

[7] Bruno Blanchet. ProVerif: Cryptographic protocol verifier in the formal model. https://bblanche.gitlabpages.inria.fr/proverif/. [Accessed 11-Apr-2023].

[8] Katriel Cohn-Gordon and Cas Cremers. Mind the Gap: Where Provable Security and Real-World Messaging Don't Quite Meet. Cryptology ePrint Archive, Paper 2017/982, 2017. https://eprint.iacr.org/2017/982.

[9] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A Formal Security Analysis of the Signal Messaging Protocol. *Journal of Cryptology*, 33(4):1914 – 1983, 2020. Cited by: 26.

[10] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. Cryptology ePrint Archive, Paper 2017/666, 2017. https://eprint.iacr.org/2017/666.

[11] Cas Cremers, Charlie Jacomme, and Aurora Naska. Formal Analysis of Session-Handling in Secure Messaging: Lifting Security from Sessions to Conversations. Cryptology ePrint Archive, Paper 2022/1710, 2022.

[12] Ksenia Ermoshina, Francesca Musiani, and Harry Halpin. End-to-End Encrypted Messaging Protocols: An Overview. In Franco Bagnoli, Anna Satsiou, Ioannis Stavrakakis, Paolo Nesi, Giovanna Pacini, Yanina Welp, Thanassis Tiropanis, and Dominic DiFranzo, editors, *Internet Science*, pages 244–254, Cham, 2016. Springer International Publishing.

[13] The Signal Foundation. Technical information. https://signal.org/docs/. [Accessed 12-Apr-2023].

[14] Nadim Kobeissi. *Formal verification for real-world cryptographic protocols and implementations*. Theses, Université Paris sciences et lettres, December 2018.

[15] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach. In *2017 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 435–450, 2017.

[16] Matrix.org. Matrix specification. https://spec.matrix.org/latest/. [Accessed 12-Apr-2023].

[17] Marino Miculan and Nicola Vitacolonna. Automated verification of Telegram's MTProto 2.0 in the symbolic model. *Computers and Security*, 126, 2023. Cited by: 0.

[18] Paul Rösler, Christian Mainka, and Jörg Schwenk. More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema. In *2018 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 415–429, 2018.

[19] Andreas Straub, Daniel Gultsch, Tim Henkes, Klaus Herberth, Paul Schaub, and Marvin Wißfeld. XEP-0384: OMEMO Encryption, Jan 2022.

[20] The Tamarin Team. Tamarin-Prover Manual. Technical report, 02 2023.

[21] Telegram. Mtproto mobile protocol. https://core.telegram.org/mtproto. [Accessed 12-Apr-2023].

[22] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure Messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249, 2015.

[23] Richard van der Hoff. docs/olm.md · master · matrix-org / Olm · GitLab. https://gitlab.matrix.org/matrix-org/olm/blob/master/docs/olm.md, 2020. [Accessed 11-Apr-2023].

[24] Sebastian R. Verschoor and Tanja Lange. (In-)Secure messaging with the Silent Circle instant messaging protocol. Cryptology ePrint Archive, Paper 2016/703, 2016. https://eprint.iacr.org/2016/703.

[25] xmpp.org. Instant messaging. https://xmpp.org/uses/instant-messaging/. [Accessed 12-Apr-2023].

Comparative Analysis of Static Analysis Kubernetes Security Tools

Ville Vastamäki
ville.vastamaki@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

KEYWORDS: Cloud Computing, Kubernetes, Security tools, Static analysis

1 Introduction

Cloud computing has become a popular choice for deploying software, and Kubernetes is a popular option for deploying software in a cloud environment. Kubernetes offers many benefits for deploying software, such as automatic scaling, roll-backing, load-balancing and updating of applications [19]. These benefits can reduce costs of running software compared to physical and virtual machines. Kubernetes can also improve service reliability, since it is easy to have many instances of the software running on different machines and physical locations either with public cloud providers or in private data centers.

As Kubernetes has become more popular in running software, the security aspects are also evaluated by the developers that use Kubernetes to run the software. One of the biggest concerns of users is the security of Kubernetes [1]. A vulnerability in the platform or software that runs on the platform can compromise the entire cluster, which can run many dif-

ferent applications. For example, hackers were able to compromise Tesla's AWS credentials to mine cryptocurrency [2]. Kubernetes offers many different features that can help make clusters more secure. These features might not be used correctly, and vulnerabilities in manifests could be exploited by attackers [17]. Because of this, there are also several security tools for improving the security of Kubernetes clusters that can help developers notice possible vulnerabilities.

This paper compares available static Kubernetes security tools, their benefits and their weaknesses. The tools analyzed include Kubernetes' own features that improve security, and third-party tools that analyze Kubernetes objects and components and highlight weaknesses in them.

This paper is divided into sections. Section 2 presents the built-in Kubernetes security features. Section 3 presents the static security tools analyzed. Section 4 discusses the strengths and weaknesses of the tools analyzed, and suggests a ways to run these tools in a developers workflow. Finally, Section 5 concludes this paper.

2 Kubernetes built-in security features

Kubernetes has several features built-in that can improve a Kubernetes cluster's security, such as Role-Based Access Control (RBAC), memory and CPU limits, pod security standards, network policies and namespace separation [21]. This section list some of the features and how they can be used to improve the security of the platform. The features and their security benefits are presented in Table 1.

Kubernetes Feature	Security Benefit
CPU and Memory limits	Reduces effectivity of DDoS attacks
Role-Based Access Control	Increases difficulty of priviledge escalation
Pod Security Standards	Limits access to Kubernetes processes
Network Policies	Reducing lateral movement of attacker

Table 1. Table showing Kubernetes built-in security features and the security benefit of using them

2.1 CPU and Memory limits

Kubernetes allows setting cpu and memory limits, that restrain the amount of memory and CPU that a pod can use. If a limit is surpassed, the plat-

form can wait before continuing execution or the pod can be evicted [14]. This can mitigate a denial of service attack, as the pod will be evicted before it uses all of the computing power available in the node [21]. Because of this, CPU and memory limits should be used in all production environments. These limits can also help spot bugs in the application, as if a pod regularly needs more memory than expected, it will keep crashing.

2.2 Role-Based Access Control

Role-Based Access Control (RBAC) can be used to limit access to computer or network resources [15]. Configuring RBAC correctly can mitigate privilege escalation, because all users cannot modify all of the resources. In general configuring users with minimal access is the best practice, because it limits the attack surface as the attacker cannot modify resources as freely to compromise the entire cluster.

2.3 Pod Security Standards

Pod security standards can be used to limit access to Kubernetes processes. Pod security standards has three different levels of policies. The privileged policy is unrestricted, the baseline policy aims to limit known privilege escalations and the restricted policy enforces current best practices for pod hardening [13]. Configuring these policies allows the developers of the cluster to either know the weaknesses of the software they develop or make their cluster more secure. The way these policies are applied in the cluster are controlled with pod security admission. Pod security admission can either trigger an audit annotation, warn the user of the policy violation or completely reject the pod [12]. Enforcing the policy applied is a good practice to limit privilege escalations in the cluster, as software requiring privileged rights is not run in the first place. This reduces the attack surface of the cluster, and is in general an easy way to improve the security of the cluster.

2.4 Network Policies

Network policies can be used to to control traffic flow at OSI layers 3 and 4 [11]. With network policies traffic to a pod can be either allowed or blocked from different sources, such as other pods, namespaces and IP addresses. Network policies are controlled seperately for ingress and egress traffic. Network policies protect attacks between pod IP addresses, reducing the

lateral movement for an attacker [18]. This is beneficial, as an attacker might not be able to move inside the cluster to gain more control of its resources.

3 Static Analysis Security Tools for Kubernetes

Even though Kubernetes has security features built in, they do not guarantee that the platform and software running on it are secure. The built in features and software deployments are often misconfigured by developers [20]. Kubernetes also releases minor versions three times a year, and each minor version has one year of patch support [7]. The minor releases add, change or remove some of Kubernetes' API's, making changes inevitable. Because of this, there exists many tools for verifying the deployment manifests and Kubernetes components. These tools can help developers configure their Kubernetes clusters and deployments for newer versions, and keep up with new best practices. The tools analyzed in this section, the Kubernetes Components and Objects they can check, and what Kubernetes built-in feature areas they affect are described in Table 2.

Security Tool	Kubernetes Components and Objects	Kubernetes built-in feature areas
Kubesec	Pods, Deployments, StatefulSets and DaemonSets	CPU and Memory limits, Network policies
Kubelinter	Pods, Deployments, StatefulSets, DaemonSets and Helm Charts	CPU and Memory limits, Network policies, RBAC
Kube-bench	Master, Controlplane, Node, etcd and Policies	Pod Security Standards, RBAC

Table 2. Table showing the security tools analyzed, how they can be run and what issues they can spot

3.1 Kubesec

Kubesec is an open source tool for scanning Kubernetes deployment files. It can be used from the command line, as a Docker container, as a HTTP server or as a service hosted by Kubesec [10]. The ability to run Kubesec from the command line makes it possible to integrate into a CI/CD pipeline

as well. Kubesec uses a ruleset to analyze the deployment manifests [9]. These rules contain checks for memory and cpu limits, priviledges of the container that is run, bad network configurations and apparmor configurations. The rules contain both positive and negative checks, either adding or reducing points for the manifest analyzed. The result contains the total score and scoring details, highlighting issues to fix in the deployment manifest. The checks match the CPU and Memory limits, Network policies of the Kubernetes built-in features described in section 2. However, as the rules are based purely on string matching sections of the manifest file, it is difficult to analyze how reliably Kubesec can identify secure deployment manifests. For some checks in Kubesec, there is no written documentation available, and it is up to the user to figure out what they check. Kubesec also only analyzes Pods, Deployments, StatefulSets and DaemonSets, so RBAC and Pod Security Standards are not taken into account.

3.2 KubeLinter

KubeLinter is another open source tool for analyzing the deployment files [5]. It can be used locally from the commandline, allowing it to be integrated into a CI/CD pipeline. KubeLinter supports Kubernetes YAML files and Helm charts. This differentiates it from Kubesec, as Kubesec only supports Kubernetes YAML files natively. It also has more checks than Kubesec, with some of them turned off by default [6]. These checks include CPU and Memory requests and limits, if the specified group has access to secrets and rights to create pods, if the deployment is run as non-root, if the ports specified are invalid or priviledged and if the pod is not selected by any network policy. Of the built-in Kubernetes features discussed in section 2, these checks match CPU and Memory limits, Network polices and also somewhat RBAC.

The amount of checks increases the coverity of KubeLinter compared to Kubesec. All of the checks are also properly documented, compared to Kubesec that does not have explanations for all of the checks it runs. With the addition of Helm charts supported, KubeLinter seems to be a better tool than Kubesec for static analysis of Kubernetes YAML files.

3.3 Kube-bench

Kube-bench is an open source tool that uses CIS Benchmarks as the base for checking if Kubernetes is deployed securely [3]. Kube-bench can be run as a container and within a cluster. Unlike Kubesec, Kube-bench checks master, controlplane, node, etcd and policies. This makes Kube-bench a tool for the overall security of the cluster, which does not take deployments into account. Kube-bench is a suitable tool for checking that the cluster-level is configured properly, as it can find missconfigurations in Pod Security Standards and RBAC. Another tool should be used for checking deployments.

CIS Benchmarks are a good reference for hardening Kubernetes Clusters [16]. There are also host-specific CIS Benchmarks for Amazon EKS, Microsoft AKS and Google GKE [4]. These can help in setting up a secure cluster for a new public cloud. The CIS Benchmarks also have some issues, most notably the newest Kubernetes version that has a CIS benchmark is 1.24.0. Kubernetes 1.24 End of Life day is little more than 3 months away [8], and the three newer Kubernetes minor versions do not have a released CIS Benchmark. This affects Kube-bench as a tool, because its support for Kubernetes versions is tied with released CIS benchmarks. When a newer version of Kubernetes is released, updating to it is recommended [21].

4 Discussion

Kubernetes is a complex platform to run software on, but it offers obvious benefits for doing so. The security features of Kubernetes and third-party open source tools discussed in this paper are only a surface level look into the world of Kubernetes and cloud security.

To improve the security of Kubernetes clusters and deployments is not an easy task, and as with security in general it will never be complete. To improve security, the developer should know at least the basics of Kubernetes security and its features. The features and third-party tools offer a starting point for improving security, alongside manual reviews of configuration files.

Of the tools analyzed, Kube-bench offers a clear benefit for creating and securing a cluster running on different public clouds, as the CIS benchmarks have specific checks for all of the major public cloud available.

However, the lack of support for newer versions of Kubernetes is a major flaw in Kube-bench, and there might be a better alternative that is able to keep up with the release schedule of Kubernetes.

KubeLinter has great documentation of the checks it has, and how to configure it further to fit the needs of the developer. KubeLinter can improve the security of Kubernetes YAML files and Helm charts, and it seems to be updated frequently. Overall, KubeLinter seems to be the best tool analyzed in this paper, but it is built for a different purpose than Kube-bench. Kubesec does not seem to have any upside for it compared to KubeLinter.

The amount of available tools for Kubernetes security is huge, so finding a tool might require time. Tools should be used to aid in keeping up with the changes and to have at least some kind of confirmation that basic security rules are in place. Tools should be used in a way that fits the developer, either inside the cluster, after changing the configuration files from the command line, or ran as a part of a CI/CD pipeline. Adding a tool to a CI/CD pipeline should not take too much time, and updating that tool when a new release is available could be a good way to have an easy sanity check for the developer.

5 Conclusion

As Kubernetes has gained a lot of popularity as a platform to run software, the security aspects naturally interest and concern users. Kubernetes is a complex system with many services, deployments and cluster-wide settings such as namespaces and policies to configure. Configuring a cluster to run software requires many different files. These files are bound to have issues, just like application code itself.

The complexity of configuring Kubernetes clusters is not eased by frequent releases and new, changed or deprecated API's between Kubernetes versions. Each new release of Kubernetes can break the deployment of an application, so updating Kubernetes versions also take time.

The area of Kubernetes security is complex, as there is so many things that affect the security of the software running on Kubernetes. Image scanning, static scanning of configuration files, dynamic scanning and penetration testing all have a place in improving the security of software running on Kubernetes. These areas are not that well researched, and would benefit from further research. As of right now, the area of Ku-

bernetes security mainly comes from an open-source or corporate background, as Kubernetes is adopted in different organizations.

References

[1] Cncf 2022 annual survey. https://www.cncf.io/reports/cncf-annual-survey-2022/ [Online]. Accessed: 17.4.2023.

[2] Hackers exploit Tesla's aws servers to mine cryptocurrency. https://securitybrief.com.au/story/hackers-exploit-teslas-aws-servers-mine-cryptocurrency [Online]. Accessed: 2.2.2023.

[3] Kube-bench. https://github.com/aquasecurity/kube-bench [Online]. Accessed: 2.2.2023.

[4] Kube-bench platforms. https://github.com/aquasecurity/kube-bench/blob/main/docs/platforms.mdcis-kubernetes-benchmark-support [Online]. Accessed: 2.2.2023.

[5] Kubelinter. https://docs.kubelinter.io// [Online]. Accessed: 2.2.2023.

[6] Kubelinter checks. https://docs.kubelinter.io//generated/checks [Online]. Accessed: 2.2.2023.

[7] Kubernetes release cycle. https://kubernetes.io/releases/release/ [Online]. Accessed: 2.2.2023.

[8] Kubernetes release history. https://kubernetes.io/releases [Online]. Accessed: 2.2.2023.

[9] Kubesec ruleset. https://github.com/controlplaneio/kubesec/blob/b39fb3e915d162745b045b5ace7615b130a434 [Online]. Accessed: 2.2.2023.

[10] kubesec.io. https://kubesec.io/ [Online]. Accessed: 2.2.2023.

[11] Network policies. https://kubernetes.io/docs/concepts/services-networking/network-policies/ [Online]. Accessed: 2.2.2023.

[12] Pod security admission. https://kubernetes.io/docs/concepts/security/pod-security-admission/ [Online]. Accessed: 2.2.2023.

[13] Pod security standards. https://kubernetes.io/docs/concepts/security/pod-security-standards/ [Online]. Accessed: 2.2.2023.

[14] Resource management for pods and containers. https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ [Online]. Accessed: 2.2.2023.

[15] Role based access control good practices. https://kubernetes.io/docs/concepts/security/rbac-good-practices/ [Online]. Accessed: 2.2.2023.

[16] *Kubernetes Security and Observability*. O'Reilly Media, Inc., 2021.

[17] Dibyendu Brinto Bose, Akond Rahman, and Shazibul Islam Shamim. 'under-reported' security defects in kubernetes manifests. In *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, pages 9–12, 2021.

[18] Francesco Minna, Agathe Blaise, Filippo Rebecchi, Balakrishnan Chandrasekaran, and Fabio Massacci. Understanding the security implications of kubernetes networking. *IEEE Security Privacy*, 19(5):46–56, 2021.

[19] Subrota Mondal, Rui Pan, H M Dipu Kabir, Tan Tian, and Hong-Ning Dai. Kubernetes in it administration and serverless computing: An empirical study and research challenges. *The Journal of Supercomputing*, 78, 02 2022.

[20] Akond Rahman, Shazibul Islam Shamim, Dibyendu Brinto Bose, and Rahul Pandita. Security misconfigurations in open source kubernetes manifests: An empirical study. *ACM Trans. Softw. Eng. Methodol.*, jan 2023. Just Accepted.

[21] Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, and Akond Rahman. Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices. pages 58–64, 09 2020.

Managing Secrets in Cloud Applications

Vipul Kumar
vipul.kumar@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

Cloud computing has become a popular platform for deploying modern applications. However, managing secrets, such as passwords and API keys, in a secure and efficient manner remains a challenging task for cloud application developers and administrators. This paper provides a comparative analysis of the secret management solutions offered by four major cloud providers, including AWS Secrets Manager, Azure Key Vault, Google Cloud Secret Manager, and Alibaba Cloud Secret Manager. The solutions are evaluated based on key features such as security, integration, rotation, auditing, secret versioning, and customization. This analysis shows that each solution has its strengths and weaknesses, and selecting the right one depends on specific application requirements and workflows.

KEYWORDS: secret management, cloud providers, secrets, access control

1 Introduction

As cloud applications continue to gain popularity and become more distributed, ensuring the protection of both the cloud infrastructure and the data it stores from malicious attacks is increasingly imperative [4]. The use of multi-cloud systems has become increasingly popular, not only for

their benefits in terms of decentralization and scalability but also for their potential to better secure data and services [10]. Storing and rotating secrets in a plain-text documents or over emails and messaging apps is common yet one of the most vulnerable ways to manage secrets. Despite the files or repositories being private, even one compromised user or misconfiguration can leak out sensitive information.

Efficiently protecting sensitive data from unauthorized access is crucial for these applications, and requires an appropriate security framework. The way to achieve this is to encrypt the data and access credentials. While encryption provides protection for resources, secret management would provide access control for the protected resources [6].

The cloud infrastructure entails the establishment of diverse secrets, including keys, tokens, passwords, and certificates, among others. These secrets can be vulnerable to different types of attacks, depending on their storage mechanisms. For example, credentials may be leaked in version control repository systems such as Git, making them susceptible to unauthorized access. In 2021, the source code of Nissan was leaked online due to a git server being exposed online with a default credentials combination of $admin/admin$ [2]. In addition to this, secrets can be vulnerable to physical or memory-based attacks, such as buffer overflow and improper bounds check. hard-coded credentials, in particular, are among the most dangerous software weaknesses listed in the CWE Top 25 list [4].

The conventional approaches for managing secrets, such as sharing customer keys with service providers, fail to comply with regulatory requirements and pose scalability and cost challenges. Managing secrets securely can be a complex and costly process, particularly when using customer-owned infrastructure.

This paper provides a review of four major cloud-based secret management services: AWS Secrets Manager, Azure Key Vault, Google Cloud Secret Manager, and Alibaba Cloud Secret Manager. The study compares and contrasts these services across six important dimensions: security, external integration methods, rotation, auditing, secret versioning, and customization/extensibility. The analysis finds that while each service has its own unique strengths and limitations, they all provide a range of features and capabilities that can help developers and administrators to manage secrets effectively in a multi-cloud environment. The paper concludes by outlining some limitations and future research directions, as well as providing practical recommendations for cloud application de-

velopers and administrators.

2 Secret Management

This section will emphasize on the different phases of secret in Sec. 2.1 and features of cloud secret management providers in Sec. 2.2 and Sec. 2.3.

2.1 Phases of life-cycle of a secret

The journey of a secret can be divided into four distinct phases, as illustrated in Figure 1.

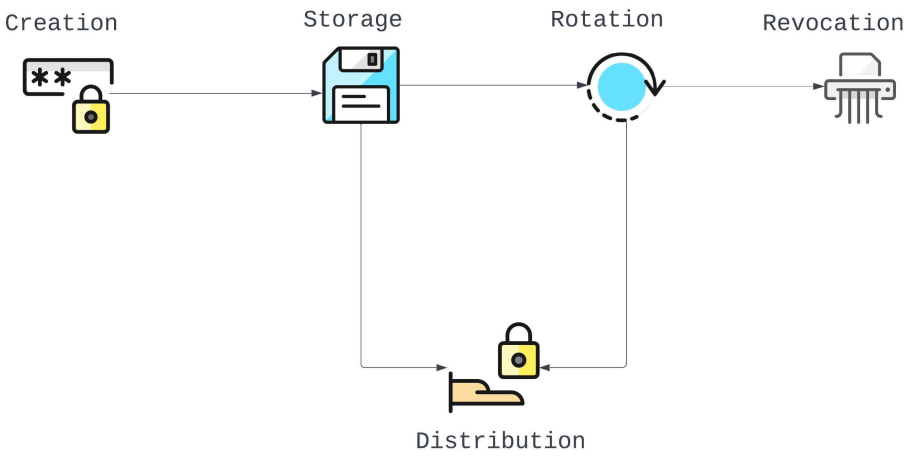


Figure 1. Phases of life-cycle of a secret

Creation refers to the phase when new secrets are generated. For example, the creation of a new password or certificate.

Storage phase comprises of storage mechanism of the secrets. Secrets can be stored in various formats. For example, they can be encrypted and then stored as a file, or they can be stored by some external secret management service.

Distribution refers to the secure and efficient distribution of secrets across different components of the cloud infrastructure. By centralizing secret management, these services can offer greater control and visibility over secrets, including access control, versioning, and auditing.

Rotation phase refers to changing or updating the secrets either on demand or on a fixed schedule. Rotating secrets improves security and decreases the risk from older secrets being exposed.

Revocation refers to destroying a secret, or removing the access permissions from it. The secret is in this phase when it is no longer required

or has been updated by a new version.

2.2 Cloud Secret Management Models

Cloud solutions can store and manage secrets including credentials and API keys. These secrets can be fetched by the application through an API call on demand. They also provide the option to manage access to the secrets through permission policies. These solutions also manage secret rotation without causing any interruption in active applications. Users or admins can also monitor the usage of the secrets and get notified about any unfavourable event as soon as it occurs [1]. This has been described in Figure 2, where the admin configures the resource and secret manager in steps 1 and 2, and then the application interacts with the secret manager to retrieve the credentials in steps 3 and 4, to access the resource in step 5.

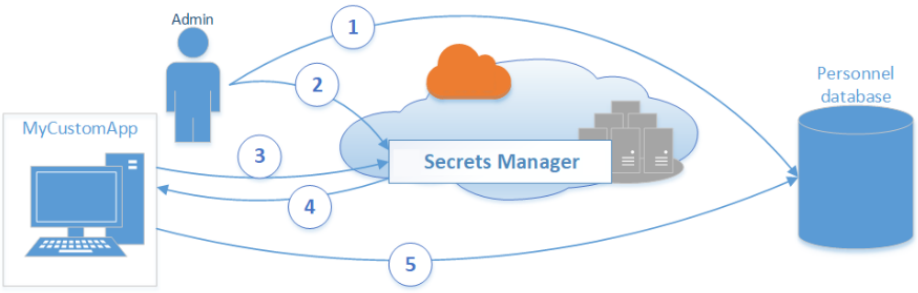


Figure 2. Storing and using credentials for a database in Secrets Manager [1]

2.3 Cloud Secret Management Solutions

This paper aims to provide an analysis of the most commonly used cloud secret management solutions. The focus of the discussion will be on the integration and usability of each solution. The following solutions will be examined:

- Google Cloud Platform (GCP) Secret Manager
- Amazon Web Services (AWS) Secret Manager
- Azure Key Vault
- Alibaba Cloud Secret manager

Each of these solutions provides unique features and benefits, and it is essential to understand their strengths and weaknesses when deciding which solution to use. The analysis will cover several factors, including access control mechanisms, SDK availability, security features, and cost-effectiveness. By examining each solution in detail, this paper aims to provide valuable insights to organizations seeking to secure their sensitive data stored in the cloud.

3 Architecture

The architecture of Secret Managers is designed to provide a secure and scalable way to store and manage secrets in the cloud. It usually allows to store and manage sensitive information like API keys, passwords, and certificates.

API: An API is the front-end interface that allows users to interact with the secret management service. It provides a set of RESTful endpoints for creating, reading, updating, and deleting secrets. This API is often accessed through a command-line tool, SDK, or web-based interface.

Secret Storage: The secret storage component is where the secrets are stored. It is responsible for encrypting and securely storing secrets at rest. The secret storage system can be based on cloud storage including Amazon S3 or Google Cloud Storage, or a dedicated key vault such as Azure Key Vault.

Encryption: Encryption is used to secure the secrets that are stored in the secret storage. The encryption keys can be managed either by the secret management service or by an external key management system like AWS KMS or Google Cloud KMS.

Access Control: Access control is used to manage who can access the secrets stored in the secret management service. Fine-grained access control is typically provided through role-based access control (RBAC), which is managed through an identity and access management (IAM) system. Access control ensures that only authorized users or applications can access the secrets.

Auditing: Auditing is an important component of a secret management service. It provides a way to track who has accessed the secrets and when. Audit logs can be stored in a log management system like Amazon CloudWatch or Google Cloud Logging.

	Cloud Secret Management Services			
Component	AWS Secret Manager	GCP Secret Manager	Azure Key Vault	Alibaba Cloud Secret Manager
API	Yes	Yes	Yes	Yes
Secret Storage	Google Cloud Storage	Amazon S3	Dedicated key vaults	Dedicated key vaults
Encryption	Google Cloud KMS	AWS KMS	built-in encryption	built-in KMS
Access Control	IAM roles and policies	IAM roles and policies	Resource Access Management	Resource Access Management
Auditing	Google Cloud Logging	Amazon CloudWatch	built-in auditing	built-in auditing
Integration	GCP	AWS	Azure	Alibaba Cloud

Table 1. Comparison of different architecture components

4 Analysis

In this section, we will examine various factors that may influence the decision to choose a particular cloud secret management solution in a distributed multi-cloud environment. The selection of a cloud secret manager is a crucial consideration for organizations that rely on cloud-based services to store, access, and manage sensitive information. Several factors may influence this decision, including the availability of Software Development Kits (SDKs), ease of integration, access control mechanisms, extensibility, security features, and versioning. Therefore, it is essential to explore these different aspects in detail to make an informed decision when selecting a cloud secret management solution.

4.1 Access Control Mechanism

Variations in access control mechanisms among different cloud services can significantly impact the selection of a cloud secret management solution. For instance, when deciding which cloud secret manager to use, it is essential to consider the level of ease of integration with a given service. If a service has already integrated with AWS Secret Manager,

it would be easier to integrate it with GCP Secret Manager than with Azure Key Vault since both GCP and AWS Secret Managers are based on Identity and Access Management (IAM) policies [1][5]. Azure Key Vault Resource Access Management (RAM) provides role-based access control to Azure Key Vault resources, while AWS Secrets Manager IAM policies provide a more general-purpose mechanism for controlling access to AWS resources, including Secrets Manager. Alibaba Cloud also uses Resource Access Management (RAM) to control access to secrets [3].

4.2 External Integration Methods

The selection of secret management solutions is influenced significantly by factors such as the availability of Software Development Kits (SDKs) for popular programming environments, as well as the level of community and support. Typically, cloud providers offer access to API tools for managing secrets and integrating them with user applications, although certain situations may require additional support.

AWS Secrets Manager, Azure Key Vault, Google Cloud Secret Manager, and Alibaba Cloud Secret Manager all offer integration with many services from their own provider, as well as APIs and SDKs for integrating with other cloud providers. Seamless integration can improve the efficiency and security of secret management in a distributed multi-cloud environment. Therefore, evaluating the level and ease of integration is an important consideration when selecting a secret management provider.

4.3 Rotation

AWS Secrets Manager automatically rotates secrets based on specific time intervals or custom lambda functions. Azure Key Vault also provides automated key rotation, which can create a new key version at a specified frequency [9]. Google Cloud Secret Manager supports both automatic and manual rotation, as well as secret versioning. Alibaba Cloud Secret Manager provides automatic rotation for secrets of specific types. as well as custom scheduled rotation, which can be automated using Alibaba Cloud Function Compute. [3].

All four services provide functionality for secret rotation and secret versioning. AWS and Google Cloud offer automatic rotation, while Azure and Alibaba Cloud require manual rotation, which can be automated using cloud functions.

4.4 Auditing

AWS Secrets Manager offers audit capabilities through API requests and changes logging and supports granular access control using IAM policies. Similarly, Azure Key Vault provides audit logs for secret-related operations and offers custom policies for access control [8]. Azure Key Vault integrates with Azure Active Directory to provide role-based access control and multi-factor authentication. Google Cloud Secret Manager provides audit logs for secret-related operations and supports custom IAM roles and permissions. The Secret Manager supports VPC Service Controls for network-level access control. Alibaba Cloud Secret Manager provides auditing through its cloud audit system and custom roles and permissions for access control. Secret Manager integrates with Alibaba Cloud Resource Access Management for fine-grained permissions control.

Auditing and access control are important aspects when selecting a secret management provider. It allows tracking and controlling access to secrets to maintain sensitive data security and comply with regulatory requirements.

4.5 Secret Versioning

AWS Secrets Manager automatically versions secrets and enables storing and retrieving multiple versions through the AWS Command Line Interface (CLI) or Software Development Kit (SDK). Azure Key Vault maintains a history of changes for auditing and allows users to create and access secret versions via the Azure portal, CLI, or SDKs [7].

Google Cloud Secret Manager supports up to 10,000 versions per secret, and users can list, get, or delete specific versions. Alibaba Cloud Secret Manager allows for the creation and management of multiple secret versions, with the option to specify version numbers during creation or update.

Versioning secrets facilitates the tracking and management of changes for auditing, and enables the retrieval of previous versions in case of accidental modification or compromise.

4.6 Customization and extensibility

AWS Secrets Manager provides customization and extensibility through custom Lambda functions and a flexible tagging system. Azure Key Vault

also offers similar customization and extensibility options through custom policies and Azure Functions.

Google Cloud Secret Manager offers extensibility via Google Cloud Functions and custom IAM roles and permissions. Alibaba Cloud Secret Manager provides similar customization through Function Compute and a flexible tagging system.

Overall, the level and flexibility of customization and extensibility is a crucial considerations when evaluating secret management providers. It determines the ability to adapt secret management to specific application requirements and workflows.

5 Conclusion

In conclusion, secret management is a crucial aspect of cloud security, and selecting a secret management provider requires careful consideration of various factors. This review of four popular cloud secret management solutions - AWS Secrets Manager, Azure Key Vault, Google Cloud Secret Manager, and Alibaba Cloud Secret Manager - revealed that they all offer similar core functionalities such as secure storage, access control, rotation, auditing, and versioning.

The selection of a suitable secret management solution depends on various factors, including the cloud service utilized by the company or other project components. However, certain infrastructure decisions related to secrets management may also influence the choice of secret management service. These decisions may include the type of application, the nature of the secrets to be managed, and the level of access control required. As such, it is crucial to carefully consider these factors when selecting a secret management solution to ensure it aligns with the organization's needs and objectives.

However, there are significant differences in their features and capabilities, such as the level of integration with different cloud services, ease of customization, and automation of rotation. These differences should be carefully evaluated based on the specific needs and requirements of the cloud application being developed or managed.

Limitations and future directions for research include the need for more comprehensive evaluations of other cloud secret management solutions and the investigation of new technologies such as homomorphic encryption for securing secrets.

Practical implications and recommendations for cloud application developers and administrators include the importance of selecting a secret management provider that integrates well with the cloud services used, supports the required customization and automation, and provides robust auditing and access control capabilities. It is also important to regularly review and update secret management policies and practices to ensure ongoing security and compliance.

References

[1] Amazon. Aws secrets manager. https://aws.amazon.com/secrets-manager/, 2023. Accessed: 2023-02-02.

[2] Catalin Cimpanu. Nissan source code leaked online after git repo misconfiguration. https://www.zdnet.com/article/nissan-source-code-leaked-online-after-git-repo-misconfiguration/, 2021. Accessed: 2023-02-02.

[3] Alibaba Cloud. Secrets manager. https://www.alibabacloud.com/help/en/key-management-service/latest/secrets-manager, 2023. Accessed: 2023-03-06.

[4] CWE. Cwe view: Weaknesses in the 2022 cwe top 25 most dangerous software weaknesses. https://cwe.mitre.org/data/definitions/1387.html/, 2022. Accessed: 2023-02-02.

[5] Google. Secrets manager overview. https://cloud.google.com/architecture/security-foundations/keys-secret-managementsecret-manager, 2023. Accessed: 2023-03-06.

[6] Xiaolong Huang. A survey of key management service in cloud. https://ieeexplore.ieee.org/document/8663805, 2018.

[7] Microsoft. Azure key vault keys, secrets and certificates overview. https://learn.microsoft.com/en-us/azure/key-vault/general/about-keys-secrets-certificates, 2023. Accessed: 2023-04-11.

[8] Microsoft. Azure key vault logging. https://learn.microsoft.com/en-us/azure/key-vault/general/logging, 2023. Accessed: 2023-04-11.

[9] Microsoft. Configure cryptographic key auto-rotation in azure key vault. ://learn.microsoft.com/en-us/azure/key-vault/keys/how-to-configure-key-rotation, 2023.

[10] Sumedh N. Pundkar and Narendra Shekokar. Cloud computing security in multi-clouds using shamir's secret sharing scheme. pages 392–395. Institute of Electrical and Electronics Engineers Inc., 11 2016.

Usability of MFA solutions

Walerius Kyllönen
walerius.kyllonen@aalto.fi

Tutor: Mario Di Francesco

Abstract

Textual passwords have been a major weakness in many systems. Multi-factor authentication one possible solution to this problem. However, the adoption of MFA can come with significant emotional and time costs. Reducing the cost of MFA is critical for increasing adoption and attitude towards MFA. These costs can be reduced via proper risk communication and better usability. Additionally, risk-based authentication can allow users to spend less time authenticating, and thus lessen the cost of authentication.

KEYWORDS: Multi-Factor Authentication, Risk-Based Authentication, Usability,

1 Introduction

Authentication is everywhere nowadays. Many services depend on strong authentication. For example, when accessing online banking services, the bank must ensure that only the owner of the account has access to it. Traditionally authentication on the Internet has been achieved using textual passwords. The main issue with textual passwords is that user-selected passwords are easily guessable, but random password are not easily mem-

orisable. Thus, passwords have been a major weakness in authentication systems since 1979 [17]. Therefore, stronger authentication has become necessary. Multi-factor authentication (MFA) is a popular solution to this problem [3].

The term MFA is used to describe an authentication process where the user authenticates using at least two different factors. These factors can be divided in to the following categories:

- Knowledge (*something you know* such as password)
- Possession (*something you have* such as device)
- Inherent (*something you are* such as biometrics)

Using MFA makes authentication more secure when compared to single-factor authentication (SFA), since a compromised secret does not necessarily compromise the whole account. However, many perceive MFA negatively, and thus are not willing to adopt MFA [7]. Users frustrated with MFA may find ways of working around MFA thus reducing security.

In addition to human factors, there are many technical issues related to MFA implementations. Few notable weaknesses include insecure delivery of one-time passwords (OTP), insecure account recovery methods, and synchronization attacks, which exploit the intertwinement of mobile and desktop compute [13].

This paper reviews the challenges related to the security and usability of MFA. Specifically: Section 2 introduces popular MFA methods. Section 3 describes challenges with usability and security of MFA. Section 4 discusses potential to enchance the security of MFA through better usability. Finally, section 5 concludes the paper.

2 MFA Methods

There exists multiple different methods for multi-factor authentication. Most of the commonly used methods have two factors for authentication, knowledge and possession. This section presents a few popular types of MFA.

2.1 Single-Factor

Not all solutions utilize different factors. They enhance security usually by asking the user to provide multiple secrets. Thus the compromise of one of the secrets would not lead directly to the compromise of the account. A representative example of this would be a scheme, where the user provides username and password, and then provides a one-time password (OTP) sent to them via email. In such case, the user is authenticated twice based on knowledge.

There exists many issues with this type of MFA. Some examples of these include predictable secrets, and reuse of secrets on multiple accounts. These are not widely used as MFA.

2.2 Knowledge and possession

Knowledge and possession is possibly the most common combination of factors for MFA. Often achieved through time-based one-time passwords (TOTP). A common workflow might proceed as follows: First the user authenticates using a password, then the system asks for a code from TOTP-producing device, which can be a mobile phone or a dedicated TOTP-device.

Another variant is based on (non time-based) one-time passwords (OTP). These are often delivered via SMS messages, which makes them less secure in comparison to TOTP-based schemes.

A third common implementation involves physical tokens (such as Yubikey) instead of OTPs. Yubikey implements the FIDO Universal 2nd Factor -protocol (U2F). U2F devices are cryptographic modules, which are used to store a master key and calculate the response to the received challenge [1].

Some mobile phone applications (e.g. Microsoft Authenticator) support authentication via push notifications. The user authenticates with a username and password, after which their mobile device will receive a push notification where they can accept or reject the authentication attempt. The application may presents some information about the authentication attempt, such as location, time, or IP-address.

2.3 Knowledge, possession, and inherent

Combining all three different types of factors is not common. This mostly happens in situations, where security is of utmost importance, such as online banking. However, using more than two factors is not required even at the highest security level [10]. Typical authentication workflow may proceed as following: first the user provides their credentials (knowledge), then verifies their authentication request on their mobile phone (possession) using a fingerprint reader (inherent).

Since smartphones are often used as a second factor and they often are equipped with a fingerprint scanner and multitude of other sensors, biometrics could be integrated into the authentication process as a third factor. This may already happen in practice: if a user has to unlock their phone with a fingerprint to use it as a second factor, they practically have a third factor. However, some authentication methods, such as push notifications, may circumvent this by allowing the user to authenticate from their lockscreen.

3 Challenges

MFA has been rising in popularity, but there still exists many difficulties, especially in the adoption and use of MFA. This section provides an overview into these challenges faced by MFA.

Many of the issues with MFA can be traced to usability and user perception, and thus can not be solved by only technical means. Generally, MFA applications are not considered usable [7]. One possible explanation for the low usability may be the myth that security and usability are mutually exclusive [16]. A contributing factor may be that the usability of MFA solutions has not been an important factor for researchers, as less than 10% of research focusing on MFA has included any usability studies [8].

Another complicating factor is that users do not understand why MFA is necessary and thus react negatively when forced to use MFA [7]. One important concern raised by users is disaster recovery. How complicated is the process of recovery? Is recovery even possible? These are important questions, and the answers might be unclear. The recovery mechanisms may even be unsafe, such as email link, which may further raise concerns.

Users are more likely to seek workarounds for MFA if it seems unnecessary [6]. They may, for example, avoid MFA by using a mobile application

that does not require MFA, even when the desktop application requires it. Additionally, if an application requires MFA every time it is used, users may try to avoid using the application. Less frequent use may cause problems, for example if employees read their work email less often.

Another challenge for widespread adoption and user acceptance may be established routines [9]. As passwords have been used as the main authentication method in practically all IT systems, users will likely be very familiar with passwords and thus may prefer using them instead of more secure options, such as MFA. Even though MFA results in enhanced security, interruptions caused by MFA, especially during time sensitive tasks, are taxing for users.

Users' risk perception is often flawed. Many are confident that they can protect themselves against phishing attacks, even though phishing has become more frequent [6]. There is a disconnect between how users perceive security of an authentication system and how secure the system is in reality [14]. For example, many users consider fingerprints to be highly secure, when in reality the fingerprint readers used in smartphones are susceptible to mimicking attacks. Authentication processes with more steps are considered more secure than those with less steps, even though there might not be any difference in security.

In certain situations, combining MFA and password managers can increase risks. Some password managers, such as Google Password Manager, do not require any authentication to autofill passwords. Combining this type of password manager with an MFA application that does not require authentication, such as Google Authenticator, on the same device negates some of the benefits of MFA. Since the secret is saved on the same device that is used as a second factor, the user is authenticated twice based on posession of the same device, thus weakening the security.

However, not every challenge poses a direct security risk. Excessive security policies may end up wearing out members of an organization [12]. This will inevitably lead to productivity losses. In addition, the excessive security policies might even end up driving people away from the organization.

4 Possible solutions

This section presents some potential solutions for the challenges described in Section 3. The main focus will be on enhancing security through better

usability, but some new technologies will also be presented as potential solutions.

Proper risk communication is an effective strategy for increasing MFA adoption and improving users' security behavior [11]. The common factor with many of the challenges is a mismatch between what level of security the users think is necessary and what the authentication methods provide. Risk communication may allow users to realise the risks of not using MFA, and thus increasing the likelyhood of them adopting MFA. Additionally, if users perceive MFA as a valuable tool that helps keep them safe, they are less likely to experience negative emotions, such as frustration, when using MFA [5].

Dividing user accounts into privileged and non-privileged accounts, and mandating MFA only on the privileged account could make MFA feel more justified [6]. The division would allow the users to use MFA only when enhanced security is necessary, which would reduce the emotional cost of mandatory MFA. Alternatively, the use of MFA could be made somewhat less frequent by applying risk-based authentication (RBA).

RBA can improve the usability without compromising security [18]. In RBA, a risk level will be calculated for each authentication attempt, and the user will only need to authenticate in accordance to this risk level. For example, when accessing company email on their on premises workstation, providing single factor (e.g. password) would suffice, but when accessing the same email on their personal laptop from airport, providing more factors would be necessary (e.g. password + TOTP). Since the additional factors are required less often, RBA reduces both the time and emotional cost of authentication without significantly compromising security. NIST recommends the use of RBA for services, where return visits are common [10].

Addressing user concerns is a valuable tool both in enhancing the security and usability of the system. Especially disaster recovery is important for users. However, the recovery process must be secure, since otherwise the improved security of MFA can be circumvented via the recovery process.

Since having a secure backup of possession factor is difficult, users should be allowed and encouraged to have multiple authenticators of the same factor. For example, a TOTP-device for daily use and a U2F-token as a backup.

Another effective option for improving the security of MFA would be to

improve the usability of MFA. At the moment many potential users of many different MFA solutions perceive MFA as too difficult and inconvenient to use. The situation can probably be improved through better risk communication, but the potential in improving usability should not be overlooked. One of the most important steps in aiding adoption of MFA would be better instructions during user enrolment. Unclear and outdated instructions during setup cause issues even for experienced users [5]. Older adults are especially vulnerable, since they have less experience in use of information technology and thus have even more trouble with unclear instructions [4].

Additionally, benefits of educating users about their MFA solution, especially concerning the potential weaknesses of the solution, should not be overlooked. User education is especially important in cases, where a single misclick from user may cause a security breach, as is the case with so called "MFA fatigue"-attacks [2].

New authentication methods have been proposed to improve both usability and security. One interesting option is Human-computable OTP [15]. The method would allow users to compute their OTP on their own, based on a visual pattern on a grid, thus eliminating the need for additional devices. However, even though an experienced user can authenticate themselves in 15 seconds, the process is more complicated than password based authentication, and thus faces many of the same challenges as MFA. Additionally, learning to use the protocol and memorize the secret image takes 30-45 minutes of repeated practice, which is significantly more than just inventing a new password or using password manager.

5 Conclusion

This paper has reviewed challenges related to the security and usability of multi-factor authentication and presented potential solutions to the challenges presented in this paper, with a focus on usability and user education.

Many of these challenges rise from poor usability. Thus, to enhance security, usability must be taken into consideration.

However, possibly more effective option would be addressing users' flawed risk perception. Thus, the potential of proper risk communication should not be underestimated. For example, proper risk communication can help users understand why MFA is necessary, which in turn reduces the emo-

tional cost of using MFA.

An interesting topic for further research would be the effects of multi-factor authentication policies in organizations. Since many organisations enforce mandatory MFA, finding out what steps have been taken to reduce the cost of MFA. Another interesting topic might be the viability of deploying human-computable OTP proposed by Matelski [15] on an organisational scale.

References

[1] Key generation. https://developers.yubico.com/U2F/Protocol_details/Key_generation.html. Accessed: 28-03-2023.

[2] MFA Fatigue Attack. https://www.beyondtrust.com/resources/glossary/mfa-fatigue-attack. Accessed: 23-01-2023.

[3] Chris Dale. SANS 2021 Password Management and Two-Factor Authentication Methods Survey. Technical report, SANS Institute, May 2021. https://resources.yubico.com/53ZDUYE6/at/f8jx2k2ks84ksc77gbk3q9tf/SANS_2021_Password_Management_and_Two-Factor_Authentication_Methods_Survey.pdf.

[4] Sanchari Das, Andrew Kim, Ben Jelen, Lesa Huber, and L. Jean Camp. Non-inclusive online security: Older adults' experience with two-factor authentication. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, volume 2020-January, page 6472 – 6481, 2021.

[5] Sanchari Das, Andrew Kim, Shrirang Mare, Joshua Streiff, and L. Jean Camp. Security mandates are pervasive: An inter-school study on analyzing user authentication behavior. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 306–313, 2019.

[6] Sanchari Das, Shrirang Mare, and L. Jean Camp. Smart storytelling: Video and text risk communication to increase mfa acceptability. In *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pages 153–160, 2020.

[7] Sanchari Das, Bingxing Wang, and L Jean Camp. Mfa is a waste of time! understanding negative connotation towards mfa applications via user generated content. *arXiv preprint arXiv:1908.05902*, 2019.

[8] Sanchari Das, Bingxing Wang, Zachary Tingle, and L Jean Camp. Evaluating user perception of multi-factor authentication: A systematic review. *arXiv preprint arXiv:1908.05901*, 2019.

[9] Florian M Farke, Lennart Lorenz, Theodor Schnitzler, Philipp Markert, and Markus Dürmuth. " you still use the password after all"—exploring fido2 security keys in a small company. In *Proceedings of the Sixteenth USENIX Conference on Usable Privacy and Security*, pages 19–35, 2020.

[10] Paul Grassi, James Fenton, Elaine Newton, Ray Perlner, Andrew Regenscheid, William Burr, Justin Richer, Naomi Lefkovitz, Jamie Danker, Yee-Yin Choong, Kristen Greene, and Mary Theofanos. Digital identity guidelines: Authentication and lifecycle management [includes updates as of 03-02- 2020], 2020. https://doi.org/10.6028/NIST.SP.800-63b.

[11] Marian Harbach, Markus Hettig, Susanne Weber, and Matthew Smith. Using personal examples to improve risk communication for security & privacy decisions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 2647–2656, New York, NY, USA, 2014. Association for Computing Machinery.

[12] Philip G. Inglesant and M. Angela Sasse. The true cost of unusable password policies: Password use in the wild. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 383–392, New York, NY, USA, 2010. Association for Computing Machinery.

[13] Radhesh Krishnan Konoth, Victor van der Veen, and Herbert Bos. How anywhere computing just killed your phone-based two-factor authentication. In *Financial Cryptography and Data Security*, volume 9603 LNCS of *Lecture Notes in Computer Science*, pages 405–421, Berlin, Heidelberg, 2017. Springer/Verlag.

[14] Karola Marky, Kirill Ragozin, George Chernyshov, Andrii Matviienko, Martin Schmitz, Max Mühlhäuser, Chloe Eghtebas, and Kai Kunze. "nah, it's just annoying!" a deep dive into user perceptions of two-factor authentication. *ACM Transactions on Computer-Human Interaction*, 29, 02 2022.

[15] Sławomir Matelski. Secure human identification protocol with human-computable passwords. In *Information Security Practice and Experience*, Lecture Notes in Computer Science, pages 452–467. Springer International Publishing, Cham, 2022.

[16] M. Angela Sasse, Matthew Smith, Cormac Herley, Heather Lipford, and Kami Vaniea. Debunking security-usability tradeoff myths. *IEEE Security & Privacy*, 14(5):33–39, 2016.

[17] Viktor Taneski, Marjan Heričko, and Boštjan Brumen. Password security — no change in 35 years? In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1360–1365, 2014.

[18] Stephan Wiefling, Markus Dürmuth, and Luigi Lo Iacono. More than just good passwords? a study on usability and security perceptions of risk-based authentication. In *Annual Computer Security Applications Conference*, ACSAC '20, page 203–218, New York, NY, USA, 2020. Association for Computing Machinery.

Gaze Interactions in Virtual Characters and Their Impact on Player Experience

Wendy Yunuen Arevalo Espinal
wendyyunuen.arevaloespinal@aalto.fi

Tutor: Robin Welsch

Abstract

This paper provides an overview of the current state of gaze interactions in virtual characters in Virtual Reality (VR) video games. Gaze behavior is fundamental for human communication, and its integration in virtual characters can create a sense of interaction and presence between the user and the character. Despite this potential, the use of gaze interactions in VR games is still limited. This paper examines different techniques proposed by researchers and game developers for modeling gaze behavior and explores how these interactions affect player experience. Additionally, it highlights opportunities and challenges in this field, and discusses immersive VR games using gaze in virtual characters in experimental and commercial environments.

KEYWORDS: gaze, social interaction, virtual characters, non-player characters, video games, virtual reality

1 Introduction

Gaze behavior is a fundamental aspect of human communication and interaction, which allows them to interpret emotions and understand intentions [1]. Although this behavior is not natural for virtual characters, it can be modeled to create a sense of interaction and presence between the

user and the character. Despite this potential, the integration of gaze behavior in Virtual Reality (VR) games is still limited, resulting in a missed opportunity to bring a higher level of realism and depth to player interactions with Non-Player Characters (NPCs).

Researchers and game developers have proposed different techniques for modeling gaze behavior [2, 3, 4], and have explored new game mechanics and interaction methods based on gaze [5, 6]. Although many non-immersive games have effectively incorporated these gaze interactions, they are yet to be widely adopted in commercial VR games.

This literature review provides an overview of the current state of gaze interactions implemented in virtual characters in Virtual Reality (VR) video games. The review also explores how these interactions affect player experience and highlight opportunities and challenges in this field.

The paper is organized as follows. Section 2 presents the difference between Avatars and NPCs as Virtual Characters, and how they interact with the player. Section 3 describes current gaze interactions implemented in virtual characters. Section 4 outlines the impact of virtual characters gaze on player engagement and immersion in games. Section 5 shows immersive VR games using gaze in virtual characters in both experimental and commercial environments. Finally, Section 6 gives some insights on the opportunities for game development.

2 Virtual characters in video games

Virtual characters are digital entities that share features with real living beings, such as humans or animals [7]. However, they are not necessarily represented as their living counterparts.

These characters are often found in various forms of entertainment, such as animated movies or video games. In the context of video games, these characters serve as both player *avatars* and interactive entities within the game world (NPCs).

Avatars: According to Nowak [8], an avatar is a virtual representation of the person using it. In the context of video games, avatars are often customizable and can reflect the personality, preferences, or style of the user. An avatar can also allow players to express themselves through body language and other visual cues [9].

This paper does not explore gaze interactions between avatars because the player does not usually interact with them but through

them.

Non-Player Characters: A Non-Player Character (NPC) is a computer-controlled character in a video game. NPCs serve as interactive entities within the game world, they often help the player to reach certain goals, fight for them, and, sometimes, they are the main focus of the story line [10].

In the following parts of this paper, the terms *NPC* and *virtual character* will be used interchangeably.

2.1 Interactions between Virtual Characters and Players

In recent years, Virtual Reality (VR) games have become increasingly popular due to the introduction of more accessible VR headsets in the end-user market [11]. These games offer immersive experiences that allow players to fully engage with virtual characters in a simulated environment. Such interactions play a crucial role in enhancing the overall experience of the game [12].

Examples of interactions between players and NPCs in games include both verbal and non-verbal communication. Verbal interactions include conversations using speech or text prompts, while non-verbal interactions include body movements, gestures and facial expressions [13].

In the same way that eye expressions within these interactions support human to human communication [13], gaze behavior is relevant for social interaction between virtual characters and players to increase the feeling of immersion and presence [12].

3 Gaze interactions implemented in VR games

Virtual reality games offer great potential for creating more immersive and engaging experiences by incorporating realistic gaze interactions between players and NPCs. These interactions can include various behaviors, such as eye contact, gaze as cues, and gaze aversion. Although such interactions are not yet fully implemented in VR games, the future looks promising for their inclusion.

3.1 Eye contact

In video games, eye contact refers to the virtual eyes of a NPC meeting and following the gaze of the player. This behavior creates a sense of

connection and engagement between the user and the virtual character, as it mimics the way humans make eye contact during social interactions [1].

Rooney et al. [14] conducted an experiment where users interacted with two virtual characters, one that made eye contact with them and one that did not engage with them. Results showed that users felt a stronger connection with the NPC that maintained eye contact and could identify its mental state more accurately. Similarly, other studies [15, 16] have shown that eye contact between the user and NPCs improves communication and creates a more natural social experience.

It has been shown that eye contact interactions between players and NPCs can significantly increase trust in virtual characters, and NPCs that cannot maintain eye contact with the player may be perceived as less trustworthy or engaged [17]. However, prolonged eye contact can create discomfort in social interactions. Similarly, in video games, excessive staring by virtual characters may be viewed as invasive or disrespectful, leading to a negative player experience [12].

3.2 Cues

Gaze can provide important non-verbal cues to others, such as an inviting look or rolling one's eyes, and plays a significant role in directing spatial attention during social interactions [1, 13].

In NPCs, gaze can be animated to achieve a more engaging and interactive experience. For example, Dong et al. [3] conducted different experiments where virtual characters showed stress and fear through eye animation. Users identified the scared character more accurately when the NPC displayed gaze, blinks, and pupil animations, while changes in pupil size alone were less effective.

In another study, Martinez et al. [18] used virtual characters with animated head movements and gaze to point at targets on a screen. The results indicated that gaze cues helped users to identify the target pointed by the NPC faster. Similarly, Kulms and Kopp [19] asked users to perform tasks while NPCs helped them focus on the correct task using social cues, such as facial expressions and gaze. All users indicated that gaze was the only cue they monitored.

These findings suggest that gaze cues can enhance the user experience and improve task performance in virtual environments.

3.3 Saccades

Saccades refer to the rapid eye movements that shift the gaze towards a new target. They are important for face-to-face communication as they enable individuals to focus their attention on different parts of the face, such as the eyes, mouth, or gestures, to extract important social cues, emotions, and intentions [1].

In the context of NPC interactions, researchers found that the direction and frequency of saccades can indicate the level of trust and engagement between the virtual character and the player. Players trusted more NPCs with neutral to happy face expressions, while there was no significant difference in trustworthiness when the NPCs had a neutral to grumpy expression [17].

Other studies have shown the potential of saccades in creating expressive virtual characters. Lance and Marsella [20] developed an algorithm that combined head orientation and vestibulo-ocular reflexes to replicate natural-looking saccades, enabling them to simulate gaze shifts of an actor displaying various emotions. Similarly, Queiroz et al. [21] aimed to replicate human behaviors, such as concentration and discomfort, by creating a virtual character that included saccades with different frequencies. The study revealed that users consistently identified animations with distinct eye behaviors as the most expressive, highlighting the importance of eye movements in conveying general expressiveness.

3.4 Gaze Aversion

Gaze aversion, the act of avoiding eye contact, can convey multiple meanings in different contexts. It can indicate the processing of information, considering a response, or even signs of aggression, disinterest, or dishonesty.

Studies have shown that in virtual conversations with NPCs, intermittent gaze aversion can make users feel heard, engaged, and reflective, leading to more positive and longer interactions [6, 22]. Gaze aversion can also regulate the flow of conversation by signaling the end of a sentence or a pause, which encourages users to wait for the NPC to finish speaking [6]. Additionally, gaze aversion in NPCs can provide cues to players to avoid interaction, which can be useful in distinguishing between characters that participate in the game and those that do not [23, 12]. Finally, gaze aversion combined with the appearance of the virtual character can

also transmit feelings of submission, which has been used in some game designs to appeal to a male audience [24].

Understanding the diverse interpretations of gaze aversion can influence its effective utilization in virtual conversations with NPCs and its potential effect on the gaming experience.

4 Impact of virtual characters gaze on player experience

This section discusses a few ideas about the impact of virtual character gaze on player experience in terms of engagement and immersion.

4.1 Engagement

One of the main benefits of virtual character gaze is that it can enhance player engagement. By using gaze cues [3, 18, 19], NPCs can create a sense of connection and interaction between the player and the character. This can facilitate social presence and increase emotional attachment to the character, making the player feel more immersed in the game world.

4.2 Immersion

Virtual character gaze can improve player immersion by providing a more natural and realistic interaction [17, 20, 21]. Gaze behavior can help players understand the intentions and emotions of virtual characters, which in turn enhances the sense of presence. Realistic eye movements and gaze behaviors can also reduce the effect where virtual characters appear too artificial or unsettling, thus improving the overall immersion of the game.

5 VR video games using gaze interactions

Gaze interactions can be useful in many types of games. However, genres such as action, adventure, role-playing, and first-person shooter games prioritize character development and storytelling, making gaze interactions more relevant. These genres aim to create immersive worlds, and gaze interactions can make virtual characters appear more alive, allowing players to form deeper connections with them. Games that emphasize player choice and interactions with NPCs are more likely to include gaze interactions to facilitate natural interactions.

Currently, there are few VR games available in the market that focus on gaze interactions, or if they do, players may not have noticed such ani-

mations. Furthermore, independent game developers who play VR games have developed *mods* (according to Scacchi, 'mods are extensions to existing game software systems' [25]) to change the way eyes were originally animated, giving them richer color and definition, thus impacting on the perception of gaze.

5.1 Experimental games

This paper examined several experimental games that explore the use of gaze interactions in virtual environments.

One of the games studied was created by Carroll [26], who developed a virtual world where gaze behavior was used to determine whether players and NPCs would engage in a social interaction. While the project provides detailed information to replicate such a world, it could be enhanced by adding a story line.

Dobre et al. [10] collaborated with two game companies to develop an immersive game environment that collected data for a machine learning model. This model studied social interactions between players and non-player characters exhibiting different emotions. The findings of this model can be applied to create characters that realistically exhibit emotions, improving the overall gaming experience.

Coffee without Words [14], another game studied in this paper, is a VR experience that challenges players to interact with a NPC in a cafe using only their gaze. Players are given a short story and must identify the emotions conveyed by the gaze of the virtual character at the end of the interaction. This innovative use of gaze interactions enhances the immersive experience and creates a more engaging game play.

Although not a game, it is essential to mention the experiment by Fox and Bailenson [24] to consider the ethical use of gaze animation in games. They developed an immersive environment where users interacted with female virtual characters that had gaze animations and stereotypical appearances representing *vamps* and *virgins*. Among their findings, users showed greater acceptance of rape myths when the characters looked more provocatively and maintained eye contact.

Overall, these implementations demonstrate the potential of gaze interactions to enhance social interactions and emotional experiences in games, while also highlighting the need for ethical considerations in their use.

5.2 Commercial games

After an informal consultation to players of VR games, it was discovered that only few commercial games utilize gaze animation effectively.

Figure 1. Improved eyes Skyrim enables users to modify the aspect of the eyes of NPCs in the game. The figure shows the difference in the animation of Dawnguards when using the mod.

In *The Elder Scrolls V: Skyrim VR*, the use of mods such as Pandorable's NPC or Improved Eyes Skyrim (see Fig. 1) is a popular practice among players to improve the appearance and eye animations of virtual characters, making social interactions more realistic and engaging. The eye contact feature is particularly interesting, as it makes players feel heard and acknowledged by the NPCs. Similarly, in *Lone Echo*, the implementation of gaze behavior in the eyes of the captain, Olivia Rhodes, creates a natural and realistic look (see Fig. 2), allowing players to interpret emotions and intentions more accurately. This feature augments the immersion and engagement of the story and the game world according to the users.

Figure 2. Eyes of Captain Olivia Rhodes. Users mention that her eyes help to create a stronger connection with her.

Hellblade: Senua's Sacrifice VR, is a game that covers psychosis, the gaze interactions with Senua feel realistic because the creators of the game worked with neuroscientists and people who also experienced it. This attention to detail amplifies the authenticity and empathy of the narrative (see Fig. 3). In contrast, in *Trover Saves The Universe*, the non-realistic animation of eyes in NPCs, such as blinks, eye contact, saccades, and the absence of eyes, increases the humor and absurdity of the game (see Fig. 4), showing that animation does not need to be accurate, but well thought and implemented.

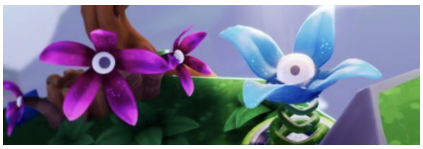
Overall, the use of gaze interactions in these games shows how this fea-

(a) Senua eye contact.

(b) Senua gaze aversion.

Figure 3. Animations of the eyes of Senua. Users believe that they often express sadness and fear.

(a) Trevor (left) and Glorkon (right).



(b) Eyed-flowers to jump.

Figure 4. Animations found in Trevor Saves The Universe. In (b) Trevor can jump over the flowers and they blink to show they are hurt by said action.

ture can enhance immersion, engagement, and storytelling in virtual environments. The attention to detail in the animation of eyes, and the use of eye contact, saccades, and other gaze behaviors, can make virtual characters more realistic and relatable to players. The examples provided in these games demonstrate the potential of gaze interactions in creating more compelling and meaningful virtual experiences.

6 Discussion for future game implementations

As previously discussed, incorporating gaze animation in NPCs is crucial to creating immersive and captivating experiences in VR games. By implementing realistic gaze interactions between players and virtual characters, game developers can enhance game play and make it more convincing and engaging. However, it is essential to balance eye contact and gaze aversion to ensure NPC interactions are both realistic and engaging, while avoiding any discomfort or unease for the player. By modeling accurate gaze of NPC characters, game developers can create a sense of engagement and presence, allowing players to perceive the characters as real and responsive.

For example, gaze behavior can be used to convey non-verbal cues, such as interest, attention, boredom, distraction, or disinterest, making it an essential aspect of the game's narrative and story line. NPCs with distinctive gaze behaviors may communicate certain emotions or traits that align with their character arc or role in the story of the game, providing

clues or information to the player and leading to new game play opportunities and discoveries.

Additionally, it is crucial to consider the ethical implications of creating social presence in virtual characters, particularly as NPCs become more human-like in behavior and expression. Players may develop emotional attachments to NPCs, leading to ethical concerns regarding controversial themes, such as violence, sexuality, and social interactions.

Finally, a well-executed gaze animation system can improve the sense of immersion, making the game world feel more realistic and interactive. Unfortunately, this important aspect is often overlooked in most VR games available in the market. In fact, many NPCs in these games wear sunglasses, masks or eye patches, which greatly reduces the potential for meaningful interaction between virtual characters and players. Therefore, it is vital for developers to consider gaze animation as a fundamental aspect of creating immersive and engaging VR games, with careful consideration and implementation.

7 Conclusion

Gaze interactions are a crucial component of creating immersive and engaging VR experiences. However, there is a lack of implementation in many games currently available in the market. Realistic gaze behavior in virtual characters can significantly enhance the game play, narrative, and story line, providing players with a sense of engagement and presence. Non-verbal cues, such as interest, attention, boredom, or distraction, can convey emotions and traits that align with the character arc or role in the game, leading to new game play opportunities and discoveries.

By incorporating a well-executed gaze animation system, developers can improve the sense of immersion, making the game world feel more realistic and interactive. However, ethical implications must be considered when creating social presence in virtual characters to avoid discomfort or unease for the player.

As the VR gaming industry continues to evolve, it is crucial for developers to prioritize gaze interactions as a fundamental aspect of creating immersive and engaging experiences. By doing so, players can enjoy a more captivating and enjoyable experience in the virtual world.

References

[1] K. Ruhland, S. Andrist, J. Badler, C. Peters, N. Badler, M. Gleicher, B. Mutlu, R. Mcdonnell, K. Ruhland, S. Andrist, J. B. Badler, C. E. Peters, N. I. Badler, M. Gleicher, B. Mutlu, and R. Mcdonnell, "Look me in the eyes: A survey of eye and gaze animation for virtual agents and artificial systems," pp. 69–91, 2014.

[2] T.-H. D. Nguyen, E. Carstensdottir, N. Ngo, M. S. El-Nasr, M. Gray, D. Isaacowitz, and D. Desteno, *Modeling Warmth and Competence in Virtual Characters*. 2015.

[3] Y. Dong, S. Jörg, and E. Jain, "Is the avatar scared? pupil as a perceptual cue," *Computer Animation and Virtual Worlds*, vol. 33, 3 2022.

[4] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency, "Openface 2.0: Facial behavior analysis toolkit," pp. 59–66, IEEE, 5 2018.

[5] E. Velloso and M. Carter, "The emergence of eyeplay," pp. 171–185, ACM, 10 2016.

[6] S. Andrist, B. Mutlu, and M. Gleicher, *Conversational Gaze Aversion for Virtual Agents*. 2013.

[7] J. Huang and Y. Jung, "Perceived authenticity of virtual characters makes the difference," *Frontiers in Virtual Reality*, vol. 3, 11 2022.

[8] K. L. Nowak and J. Fox, "Avatars and computer-mediated communication: A review of the definitions, uses, and effects of digital representations on communication," *Review of Communication Research*, vol. 6, pp. 30–53, 2018.

[9] F. Biocca, "The Cyborg's Dilemma: Progressive Embodiment in Virtual Environments [1]," *Journal of Computer-Mediated Communication*, vol. 3, 09 1997. JCMC324.

[10] G. C. Dobre, M. Gillies, D. C. Ranyard, R. Harding, and X. Pan, "More than buttons on controllers," pp. 1–8, ACM, 9 2022.

[11] J. C. F. Ho and R. Ng, "Perspective-taking of non-player characters in prosocial virtual reality games: Effects on closeness, empathy, and game immersion," *Behaviour & Information Technology*, vol. 41, pp. 1185–1198, 4 2022.

[12] M. Vidal, R. Bismuth, A. Bulling, and H. Gellersen, "The royal corgi," vol. 2015-April, pp. 115–124, ACM, 4 2015.

[13] R. S. Hessels, "How does gaze to faces support face-to-face interaction? a review and perspective," *Psychonomic Bulletin & Review*, vol. 27, pp. 856–881, 10 2020.

[14] B. Rooney, C. Burke, K. Balint, T. O'Leary, T. Parsons, C. T. Lee, and C. Mantei, "Virtual reality, presence and social cognition: The effect of eye-gaze and narrativity on character engagement," pp. 1–6, IEEE, 10 2017.

[15] M. L. Yuan, G. G. Chua, F. Farbiz, and S. Rahardja, *Eye contact with a virtual character using a vision-based head tracker*. 2011.

[16] N. Bee, J. Wagner, E. André, F. Charles, D. Pizzi, and M. Cavazza, "Interacting with a gaze-aware virtual character," vol. 28, pp. 71–77, ACM, 2 2010.

[17] A. Normoyle, J. B. Badler, T. Fan, N. I. Badler, V. J. Cassol, and S. R. Musse, "Evaluating perceived trust from procedurally animated gaze," pp. 141–148, ACM, 11 2013.

[18] S. Martinez, R. J. S. Sloan, A. Szymkowiak, and K. Scott-Brown, *Using Virtual Agents to Cue Observer Attention Assessment of the impact of agent animation*. 2010.

[19] P. Kulms and S. Kopp, *Using Virtual Agents to Guide Attention in Multi-task Scenarios*. 2013.

[20] B. J. Lance and S. C. Marsella, "The expressive gaze model: Using gaze to express emotion," *IEEE Computer Graphics and Applications*, vol. 30, pp. 62–73, 7 2010.

[21] R. B. Queiroz, L. M. Barros, and S. R. Musse, "Providing expressive gaze to virtual animated characters in interactive applications," *Computers in Entertainment*, vol. 6, pp. 1–23, 10 2008.

[22] S.-H. Kang, A. Feng, A. Leuski, D. Casas, and A. Shapiro, "Smart Mobile Virtual Humans: "Chat with Me!"," in *Proceedings of the 15th International Conference on Intelligent Virtual Agents (IVA)*, (Delft, Netherlands), pp. 475–478, Springer, Aug. 2015.

[23] M. Lankes, "Social gaze in minimalist games," pp. 450–460, ACM, 11 2020.

[24] J. Fox and J. N. Bailenson, "Virtual virgins and vamps: The effects of exposure to female characters' sexualized appearance and gaze in an immersive virtual environment," *Sex Roles*, vol. 61, pp. 147–157, 8 2009.

[25] W. Scacchi, *Modding as an Open Source Approach to Extending Computer Game Systems*. IGI Global, 2011.

[26] D. Carroll, "Attention and communication in virtual worlds: Interacting with non-player characters in virtual reality," Master's thesis, 2022.

Review on the security of OpenID Connect

Xu Feng
xu.feng@aalto.fi

Tutor: Aleksi Peltonen

Abstract

OpenID Connect is a widely adopted single-sign-on protocol. Researchers have conducted many analyses on the security of the protocol. This paper reviews related literature and summarizes critical security vulnerabilities, mitigation approaches and diagnostic tools for OpenID Connect. The findings indicate that most vulnerabilities arise from the unlinkability between multiple protocol phases and depend on several classical attacks. Both roles involved in the protocol should take measures to mitigate security vulnerabilities. This review provides a high-level perspective of the security of OpenID Connect and will help developers build secure applications.

KEYWORDS: OpenID Connect, security vulnerabilities, authentication, authorization, session integrity

1 Introduction

In recent years, the use of modern web applications has become increasingly popular. However, it is a challenging task for users to manage a large number of application accounts. Therefore, researchers have proposed Single Sign-On (SSO) protocol, where users delegate authentication

on the Relying Parties (RP) to other Identity Providers (IdP) [1]. OpenID Connect, developed by Google, is one of the most widely-adopted SSO protocols. Its use cases include web, cloud, mobile devices and IoT [2, 3]. OpenID Connect is based on the OAuth 2.0 protocol [4, 5], which provides a secure way for users to grant access to their resources to third-party applications without sharing their credentials directly.

Although OpenID Connect helps users manage their application accounts, it suffers from potential security and privacy vulnerabilities. Based on the reviewed analyses, many relying parties suffer from at least one security vulnerability in their OpenID Connect implementations. Therefore, it is crucial to have an overview of the different types of security vulnerabilities of OpenID Connect, the recommended mitigation approaches and diagnostic tools.

This paper reviews the latest security analyses of OpenID Connect, including security vulnerabilities, mitigation approaches and diagnostic tools.

The rest of the paper is organized as follows. Section 2 explains OpenID Connect fundamentals. Section 3 summarizes the security vulnerabilities, related mitigation approaches and diagnostic tools for OpenID Connect. Section 4 provides findings related to the reviewed security analyses of OpenID Connect. Finally, Section 5 concludes this paper.

2 OpenID Connect

2.1 Overview

OpenID Connect enables users to log into the relying party by authenticating themselves at the identity provider [5]. The overall processes can be divided into three phases [6]. At the discovery phase, the RP connects to the correct IdP and retrieves necessary information for authentication. At the registration phase, the RP registers at the IdP and obtains its client identifier. At the login phase, the user authenticates at the IdP. The IdP then returns the ID token and access token to the RP. The **ID token** is the proof of the user's authentication at the IdP, with which the RP can confirm the identity of the user [5]. The **access token** proves that the user has authorized the RP to access the protected resources at the IdP [5].

2.2 Protocol Flow

The protocol flow of OpenID Connect is the interaction between the end user (browser), relying party and identity provider. The protocol specifies three modes: authorization code mode, hybrid mode and implicit mode. Most RPs adopted the authorization code mode. Next, we briefly summarizes the basics of these three modes.

Authorization code mode. In this mode, after the user's authentication, the RP will receive an authorization code from the IdP [5]. Then, the RP redeems the *id_token* and *access_token* by presenting the received code at the IdP. The following is a detailed step-by-step protocol flow of OpenID Connect in authorization code mode [6]. The flow is also shown in Figure 1.

1. The user enters the RP login page to initiate an SSO request by submitting the email.
2. The RP requests the email domain server and determines the correct IdP for later connection based on the WebFinger mechanism [7].
3. The RP receives the discovery endpoint of the IdP.
4. The RP retrieves all necessary OIDC configurations from the IdP.
5. The IdP responds with configurations including the *issuer* (the issuer identifier of the IdP), *jwksURI* (JSON Web Key Set URI [8]), *regEP* (registration endpoint), *authEP* (authentication endpoint), *tokenEP* (token endpoint) and *userinfoEP* (user information endpoint).
6. The RP retrieves the public key from the IdP by accessing *jwksURI* in case the RP does not have one.
7. The IdP responds with its public key, which is used later by the RP to validate signatures from the IdP.
8. The RP provides its redirect URIs and registers itself at the IdP through *regEP*.
9. The IdP responds with the *client_id* of the RP and possibly a *client_secret*.
10. The RP sends back to the browser the *authEP*, a randomly-generated *state* value, redirect URI of the RP, *client_id* / *client_secret* and possibly a *nonce*.
11. The browser is redirected to the *authEP* with all the parameters in step 10.
12. The browser displays the login page of the IdP.

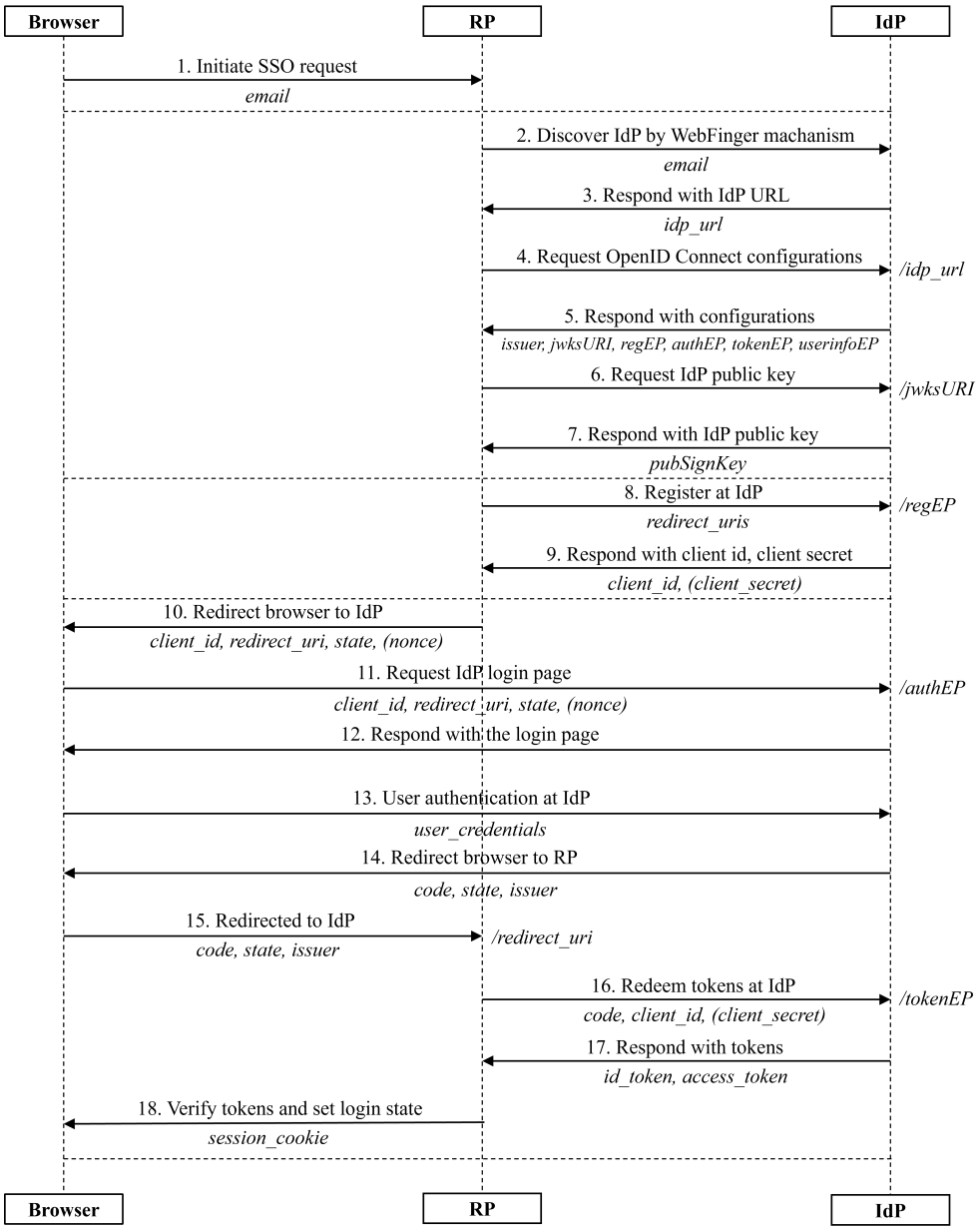


Figure 1. Protocol flow of OpenID connect (authorization code mode) [5, 6]

13. The user submits valid credentials to the IdP for authentication.
14. The IdP sends back the generated authorization code, the issuer identifier and the state value, which the RP sends in step 11.
15. The RP received the authorization code, the issuer identifier and the state value. If the state value and issuer identifier match the previous steps, the received code is trustworthy.
16. The RP redeems tokens at the IdP by presenting the authorization code and the *client_id*/*client_secret* to the *tokenEP*.
17. The IdP issues the *id_token* and *access_token*.
18. After the verification of tokens by using the public key, the RP of-

ficially confirms the user's identity. The RP can later show the *access_token* at *userinfoEP* to obtain the user's protected resources at the IdP.

Hybrid mode. The IdP issues the authorization code and either the *id_token* and *access_token*. Then, the RP extracts them from the URL fragment and uses the code to request the other token [5].

Implicit mode. The IdP directly issues the *id_token* and *access_token*. These values are appended as the URL fragment, and the RP uses JavaScript to extract them and then send them to the RP [5].

3 Security Analysis

Fett et al. [6] first formally analyze OpenID Connect and proves the security properties of OpenID Connect, including authentication, authorization and session integrity. However, OpenID Connect still suffers from a large number of security vulnerabilities when it is deployed to real-world production. This section illustrates some critical vulnerabilities in the analysis, the related mitigation approaches and diagnostic tools. Fortunately, all of those vulnerabilities are reported and fixed.

3.1 Threat Models

Security analyses of OpenID Connect implementations are based on threat models, where attackers have different capabilities to perform malicious actions. The web attacker model [9, 10] is mostly used in the analyses. In this model, a web attacker is capable of sending arbitrary HTTP requests to any public web applications and receiving responses. Furthermore, the attack can post malicious links or contents on the public web application, which trick victims into opening a malicious Uniform Resource Identifier (URI) [10]. In some analyses, a passive network attacker is also considered, who is able to intercept unencrypted network data [9].

Dolev-Yao model [11] is a widely adopted formal model, which is used to prove the properties of protocols. Based on the Dolev-Yao model, Fett et al.'s [12] propose FKS web model, which defines a general communication model for web systems, to formally analyze the security of OpenID Connect [6].

3.2 Malicious Endpoints Attacks

The malicious endpoints attacks take advantage of second-order vulnerabilities in OpenID Connect [10], where the attacker places the attack vectors in the discovery phase and executes the attack in later phases. Based on the goals of the attacks, Mainka et al. classifies them into four types [10]:

(1) IdP Mix-Up Attack. Fett et al.'s [6, 13] analysis first reports this attack. The user is tricked to initial the SSO request to the attacker's malicious IdP (possibly by Client Side Request Forgery, CSRF). Then, the attacker makes the *regEP* and *tokenEP* (token endpoint) point to the malicious IdP but keeps the *authEP* pointing to the honest IdP (step 5). As a result, the user authenticates at the honest IdP, but the RP redeems tokens at the malicious IdP, leading to authorization code leakage.

(2) Server Side Request Forgery (SSRF). In this case, the attacker can intentionally set some endpoints (step 5) to point to other service endpoints of the client. Later, the client may call those endpoints, leading to unexpected consequences.

(3) Code Injection Attack. Like the IdP mix-up attack, when the user redeems tokens at the malicious IdP, the attacker injects malicious code into the messages, which may be stored in the application and cause persistent XSS (Cross Site Scripting) attacks.

(4) Denial-of-Service (DoS) Attack. Similar to SSRF, some endpoints are modified to point to large data files, which leads to the client wasting network and memory resources.

An effective way to mitigate Idp mix-up attack is to add the IdP issuer identifier to the response in step 15, which allows the RP to check whether the code received in step 15 is from the expected IdP [6]. This binds the discovery phase and authentication phase together [10]. Mainka et al. proposed two other mitigation approaches, including the whitelist verification of IdPs in the discovery phase, and the binding of the *authEP* and *tokenEP* [10].

To mitigate the SSRF and Code injection attacks, the client should carefully check and escape the received endpoints and messages. To protect against the DoS attack, Mainka et al. suggests that the client check the *Content-Length* HTTP header before accessing any endpoints [10].

3.3 Session Swapping Attack

The session swapping attack [9, 6] is a type of CSRF attack that happens because the RP developers forget to generate a state value in step 10 or check the returned state value in step 15. To start the attack, the attacker logs into the RP following the normal protocol flow until step 14. The attacker intercepts the redirect link containing its own authorization code and constructs a CSRF attack using that link. As a result, after a click the link (step 15), the user logs into the RP using the attacker's identity, which may cause private data leakage. According to Li et al.'s assessment [9], 24 of the 33 surveyed RP clients are vulnerable to the session swapping attack.

Li et al. [9] suggests that the RP should generate a state value in step 10 and check whether the returned state value in step 15 matches with the one in step 10. Since many RP developers in the real world often ignore the state parameter, they propose a new way to mitigate such CSRF attacks. The RP needs to ensure the Referer Header of the response contains the domain of either the IdP or RP before redeeming tokens [14, 15]. However, even if the state value is appropriately used, it can be stolen by extracting the HTTP Referer Header in a CSRF or XSS attack [6, 13], which still leads to the session swapping attack. Fett et al. [6] suggested suppressing the Referer Header by applying referer policies and letting the state value expire right after usage.

3.4 Naïve RP Session Integrity Attack

The naïve RP session integrity attack [6] happen because the specification does not instruct developers to avoid using naïve user intention tracking, where the *tokenEP* is directly appended to the redirect URI in step 10. In that case, the attacker may make the *tokenEP* point to another IdP and replace the authorization code with the attacker's code issued from that IdP. As a result, the user logs into the attacker's account at the RP. The RP should use explicit user intention tracking to avoid this attack, where the *tokenEP* is stored within the session data on RP servers.

3.5 307 Redirect Attack

HTTP status codes 303 and 307 can both be used for temporary redirection after authentication. However, 303 always redirects using a GET

request, while 307 preserves the original HTTP request method. In the latter case, the browser will resend the HTTP POST request containing credentials from the previous request to the RP, which is vulnerable to impersonation. Fett et al. [6, 13] advised developers to use a 303 status code for redirection since the specification does not mention it.

3.6 Automatic Authorization Granting Attack

Google's "automatic authorization granting" feature will automatically respond with tokens to the RP if the user has already been authenticated to Google and granted permission for the RP [9]. The malicious script may send an authorization request to Google by an XSS attack. As a result, the *access_token* is extracted and sent to the attacker's server. Li et al.'s [9] assessment shows that all the RPs investigated are vulnerable to such attack.

3.7 Wrong Implementation

Some RP developers misunderstand the OpenID Connect specification [9]. For example, 6 of the 33 RPs are suspicious of authenticating users only by Google ID instead of the redeemed *id_token* [9]. Thus, the attacker may impersonate the user if the user's Google ID is leaked. 15 out of the 19 RPs authenticate users by *access_token* [9], which may lead to other RPs impersonating users by submitting the same *access_token*. Some RP clients even connect to the IdP without SSL protection. Li et al. [9] suggest that all RP developers understand and strictly follow the specification before implementation.

3.8 Security Diagnostic Tools

In the real world, it is impossible to expect each developer to fully understand and strictly follow the specification without making any mistakes. Thus, many security diagnostic tools are designed to detect vulnerabilities automatically, warn users and fix potential issues. This section explains some newly-designed security diagnostic tools.

PrOfESSOS. PrOfESSOS is a simple tool for automatic analysis of OpenID Connect implementations. It can detect, and evaluate both single-phase and cross-phase attacks [1].

OAuthGuard. OAuthGuard is a Javascript Chrome browser extension, capable of automatically monitoring HTTP requests, identifying potential

vulnerabilities and performing appropriate mitigation approaches [16]. OAuthGuard supports detecting five classes of OpenID Connect security vulnerabilities, including CSRF attacks, impersonation attacks, authorization flow misuse, unsafe token transfers and privacy leaks. According to the experiment [16], OAuthGuard successfully revealed that half of the 137 RPs had at least one security vulnerability.

4 Discussion

This section provides some thought-provoking findings based on the review of security analyses, which might enlighten OpenID Connect developers.

(1) Most vulnerabilities arise from the unlinkability between phases. Most attacks take advantage of the unlinkability between two or more phases. For example, in malicious endpoints attacks [10], the RP does not know whether the endpoints received in the discovery phase are reliable. In session swapping attacks [9, 6] and naïve RP session integrity attacks [6], the RP fails to redeem tokens at the IdP where the user authentication happens. All the mitigation approaches add additional parameters, such as the issuer identifier and the state value, to link different phases together. Mainka et al.'s [1] analyses shows similar results. The lack of binding between multiple phases causes cross-phase attacks, which are difficult to detect. On the contrary, single-phase attacks, such as the wrong recipient attack, replay attack and signature bypass attack [1], are well-researched. If RP developers follow the specification, these attacks can be avoided.

(2) Most attacks rely on other classical attacks. Nearly all the attacks reviewed in this paper depend on a CSRF or XSS attack, which leads to the user sending unauthorized requests or leaking essential values. Developers should keep paying attention to these well-studied classical attacks and adopting appropriate mitigation approaches.

(3) Both RPs and IdPs should take responsibility. Most attacks are caused by RP developers failing to follow the security guidelines in the specification [5]. Developers either misunderstand how to implement the OpenID Connect service or sacrifice security for easy implementation. However, IdPs should also be responsible for these attacks. For example, according to an analysis [9], Google did not return the state value in step 14, and the sample code did not instruct developers to verify the

state value. In addition, the "automatic authorization granting" feature also brings potential vulnerabilities. All in all, both RPs and IdPs need to collaborate to build more web applications.

5 Conclusions

This paper reviews common attacks against OpenID Connect, methods, and tools to mitigate and detect those attacks. The review finds that the loose bindings between multiple protocol phases may be the primary reason for most security vulnerabilities. Some classical attacks are the foundation of most OpenID Connect attacks. In the future, it is highly recommended that RP developers should understand and follow the specification before implementation. After that, developers can utilize appropriate security diagnostic tools to detect potential vulnerabilities in their implementation. In addition to RPs, IdPs should improve their OpenID Connect service documentation and actively fix reported security vulnerabilities.

References

[1] Christian Mainka, Vladislav Mladenov, Jörg Schwenk, and Tobias Wich. SoK: single sign-on security—an evaluation of OpenID Connect. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 251–266. IEEE, 2017.

[2] Glauber Batista, Charles Miers, Guilherme Koslovski, Maurício Pillon, Nelson Mimura Gonzalez, and Marcos Simplicio. Using externals IdPs on OpenStack: A security analysis of OpenID Connect, Facebook Connect, and OpenStack authentication. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 920–927. IEEE, 2018.

[3] Nitin Naik and Paul Jenkins. Securing digital identities in the cloud by selecting an apposite federated identity management from SAML, OAuth and OpenID Connect. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 163–174. IEEE, 2017.

[4] Michael Jones and Dick Hardt. The OAuth 2.0 authorization framework: Bearer token usage. Technical report, 2012.

[5] Natsuhiko Sakimura, John Bradley, Mike Jones, Breno De Medeiros, and Chuck Mortimore. OpenID Connect core 1.0. *The OpenID Foundation*, page S3, 2014.

[6] Daniel Fett, Ralf Küsters, and Guido Schmitz. The web SSO standard OpenID Connect: In-depth formal security analysis and security guide-

lines. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 189–202. IEEE, 2017.

[7] Paul Jones and Gonzalo Salgueiro. WebFinger. Technical report, 2013.

[8] Michael Jones, John Bradley, and Nat Sakimura. JSON web token (JWT). Technical report, 2015.

[9] Wanpeng Li and Chris J Mitchell. Analysing the security of Google's implementation of OpenID Connect. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13*, pages 357–376. Springer, 2016.

[10] Vladislav Mladenov, Christian Mainka, and Jörg Schwenk. On the security of modern single sign-on protocols: Second-order vulnerabilities in OpenID Connect. *arXiv preprint arXiv:1508.04324*, 2015.

[11] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[12] Daniel Fett, Ralf Küsters, and Guido Schmitz. An expressive model for the web infrastructure: Definition and application to the browser id SSO system. In *2014 IEEE Symposium on Security and Privacy*, pages 673–688. IEEE, 2014.

[13] Daniel Fett, Ralf Küsters, and Guido Schmitz. A comprehensive formal security analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1204–1215, 2016.

[14] Wanpeng Li, Chris Mitchell, and Thomas Chen. Mitigating CSRF attacks on OAuth 2.0 systems. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–5. IEEE, 2018.

[15] Wanpeng Li, Chris Mitchell, and Thomas Chen. Mitigating CSRF attacks on OAuth 2.0 and OpenID Connect. *arXiv preprint arXiv:1801.07983*, 2018.

[16] Wanpeng Li, Chris Mitchell, and Thomas Chen. Oauthguard: Protecting user security and privacy with OAuth 2.0 and OpenID Connect. In *Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop*, pages 35–44, 2019.

Adversarial Attacks on Machine Learning Based Malware Detection Systems

Sanzhar Yeleuov
sanzhar.yeleuov@aalto.fi

Tutor: Blerta Lindqvist

Abstract

KEYWORDS: adversarial attacks, malware detection

1 Introduction

In recent years, interest in machine learning is rapidly growing both in industry and academia. Machine learning is extensively used in prediction and complex decision-making processes due to the ability to analyze and learn from huge data sets. Among applications that benefit from using machine learning algorithms, there are critical applications such as self-driving cars and malware detection.

However, security concerns have been raised due to the potential for machine learning algorithms to be exploited in adversarial settings, leading to misclassification [4, 6, 11, 14, 15]. This type of behavior is known as *Adversarial Attack*.

To address these issues some studies proposed defense strategies, some examples of which are Adversarial Training and Gradient Hiding. [5]. Although, some protection mechanisms were proved to be insufficient by techniques presented by other researchers [3].

This paper reviews black-box testing data attacks against malware de-

tection models, their limitations and success rates of attacks against benchmark models. Section 2 covers background knowledge about adversarial attacks and techniques used to detect malware. A detailed review of the algorithms proposed by state-of-the-art attacks will be introduced in section 3. Section 4 will cover experimental results against VirusTotal [1]. Section 5 concludes with a summary and further research proposals.

2 Background

2.1 Malware Detection Methods

Malware detection methods can be categorized into three types: signature based, specification based and heuristic based. [20]

Signature-based malware detection is a traditional approach that involves the identification of known malware through the use of predefined patterns, or signatures, that are unique to each malware variant. This technique relies on the comparison of the digital fingerprint of a suspicious file or code segment with a database of signatures that are generated by analyzing the characteristics of known malware samples. When a match is found between the suspicious code and a signature in the database, the file is classified as malicious and appropriate action is taken. Although signature-based detection is effective against known malware, it may fail to detect new and previously unseen malware variants that do not match any existing signatures.

Specification-based malware detection is a technique that involves the creation of a set of rules, or specifications, that define the expected behavior of legitimate software applications or operating system functions. [20] These specifications are used to verify the behavior of a program or system component. If the observed behavior deviates from the expected specifications, the program or component is flagged as potentially malicious. Advantages of this method are ability to detect zero-day attacks and low rate of false positives. On the other side, main disadvantage of this approach is that it requires a detailed understanding of the expected behavior of the software or system components being monitored, which can be challenging in complex and constantly evolving software environments.

Heuristic-based malware detection is another technique that observes

behavior of the program in order to identify malware. However, in contrast with specification-based approach, it does not require specification of tested software. Instead, heuristic-based malware detection utilizes machine learning to make a decision. [20] By analyzing specific attributes or patterns within a program, these techniques aim to uncover previously unknown or undetected malware by drawing on a broader understanding of malicious behavior. However, it may also produce false positives and may be less effective against sophisticated malware that is specifically designed to evade heuristic-based detection methods.

2.2 Machine Learning in Malware Detection

Application of machine learning varies in malware detection process. Firstly, machine learning model can be used as a classifier. According to Kaspersky, data used for classification can be divided into two categories: pre-execution phase data and post-execution phase data [12]. Pre-execution phase data refers to information that can be retrieved without file execution, while post-execution phase data refers to artifacts of the program, e.g. logs of events triggered during execution [12]. Classifiers utilizing pre-execution phase data are safer and require less computation power, since program is not executed to be tested. However, malicious part of some malware is either encrypted or absent, since it is downloaded from remote server during execution and, additionally, it is not always possible to identify malware solely based on pre-execution phase data. Consequently, most vendors use classifiers utilizing post-execution phase data or multiple classifiers based on both types of data [12, 18]. Secondly, machine learning can be used to generate signatures for signature-based malware detection [19].

2.3 Adversarial Attacks

Adversarial attacks are a class of attacks that aim to exploit vulnerabilities in machine learning models by intentionally modifying input data to cause misclassification or erroneous predictions. Based on which data is manipulated, those attacks can be categorized into two types: training and testing data attacks.

Training data attacks, also known as poisoning attacks [16], refer to attempts to manipulate or compromise the data used to train machine learning models. These attacks may involve introducing biases or distor-

tions into the training data to produce models that exhibit undesirable behavior or are susceptible to targeted attacks.

Testing data attacks aim to manipulate input data that elicits incorrect outputs or outputs revealing sensitive information from the target model, without making any direct modifications to the underlying architecture of the model itself [16]. Based on amount of knowledge attacker possess about target model, testing data attacks can be divided into two categories: white-box and black-box.

White-box attacks are characterized by adversary conversant of internal structure and parameters of the target model.

Black-box attacks are characterized by lack of knowledge regarding target model. As a result, attackers are typically restricted on collecting input-output pairs to infer decision-making process of the model.

3 State-of-the-Art Attacks

Many research is executed covering various adversarial attacks. However, most of the classical research works in the field of adversarial attacks, both white-box [3, 4, 15] and black-box [6, 14], are done in the field of image recognition [3, 6]. The primary limitation transferring adversarial attacks designed for image recognition models to the domain of malware detection is the restricted freedom in modifying malware samples. While several operations performed on image samples may produce negligible alterations, the same operations, when applied to executable files, may disrupt the format of the executable file or impact the behavior of the program, rendering it non-malicious. Due to this reason this section will cover only works specifically focused on adversarial attacks on malware detection machine learning models.

3.1 Padding Attack

The white-box adversarial attack proposed by Kolosnjaji et al. [13] seeks to generate adversarial malware samples that evade detection by deep learning models trained for malware detection. The approach involves analyzing the target model's decision boundary and the algorithm utilizing a gradient-based optimization technique that iteratively modifies the malware samples to generate adversarial perturbations, which are undetectable by the model. The mutation process in proposed work employed

the property, where appending bytes to the PE executable has no effect to the behavior of the program. Experimental results demonstrated the ability of proposed algorithm to generate effective adversarial malware samples evading detection by MalConv, the deep neural network malware detection model proposed by Raff et al. [17].

3.2 DOS header manipulation

Another successful white-box adversarial attack targeted against MalConv [17] was proposed by Demetrio et al. [9]. Additionally, authors of this work also utilized the algorithm proposed by Kolosnjaji et al. [13] to generate perturbations. However, instead of padding samples, generated byte sequence was injected into DOS header, utilizing the fact that the majority of bytes contained within the DOS header of a PE file have no impact on execution of the program.

3.3 RAMEN

Article called "Adversarial EXEmples" by Demetrio et al. [8] propose adversarial attack framework called RAMEN. RAMEN allows to reduce adversarial examples generation to the form of problem optimization. In case of white-box scenario, adversary can approach reduced problem using gradient-based attacks [8]. Same method was employed by previously mentioned works [13, 9]. On the other hand, author suggests that black-box attacks may be approached by exploiting gradient-free optimizers, e.g. genetic algorithms and zeroth-order optimizers [6]. Additionally, another contribution of this paper is description of functionality-preserving modifications from previous works, as well as introduction of three novel attack strategies. Some of the modification strategies will be referred in subsequent sections, i.e. header fields, extend and shift.

Header Fields locates perturbations in fields inside COFF and optional file headers, which can be altered in an independent and arbitrary manner.

Extend expands the DOS header's area and injects there adversarial examples the same way as in previously mentioned DOS header manipulation [9].

Shift forges space for the perturbations inside the samples by shifting first section and modifying offset in section table.

3.4 GAMMA

Black-box adversarial attack framework called Genetic Adversarial Machine learning Malware Attack (GAMMA) was presented by Demetrio et al. [10]. Framework employs functionality-preserving modifications, e.g. padding and DOS header manipulation, as a base block. This work is the implementation of RAMEN using genetic optimizer. Genetic optimizer starts by generating initial population. Initial population of the GAMMA is generated from set of benign applications in attempt to optimize query efficiency. Population consists of vectors of functionality-preserving modifications applied to generate mutation of the malware. Furthermore, new samples are produced by mixing and mutating population samples. All malware mutations, including both population and new samples, are tested against target classifier. Finally, new population is selected by picking N samples with lowest classification outputs. Process is repeated until query limit is reached and sample with lowest classifier outputs is returned.

3.5 MalGAN

Hu and Tan [11] proposed approach utilizing a generative adversarial network (GAN) to craft adversarial perturbations called MalGAN. MalGAN consists of generator and substitute detector. Generator is a multi-layer feed-forward neural network. Input vector fed to the generator is consisted of feature vector extracted from malware and noise vector. Firstly, generator, which is limited to adding insignificant features to the malware to maintain malicious features intact, produces perturbations. Secondly, generated adversarial example is tested against black-box classifier. Lastly, input-output pair is sent to second multi-layer feed-forward neural network, substitute detector. Primary objective of the substitute detector is to fit target model and supply generator with gradient information.

4 Experiments

Majority if not all vendor malware detection solutions utilize machine learning to some extent. Goal of conducted experiments was evaluation of adversarial attacks against proprietary antivirus products. For ex-

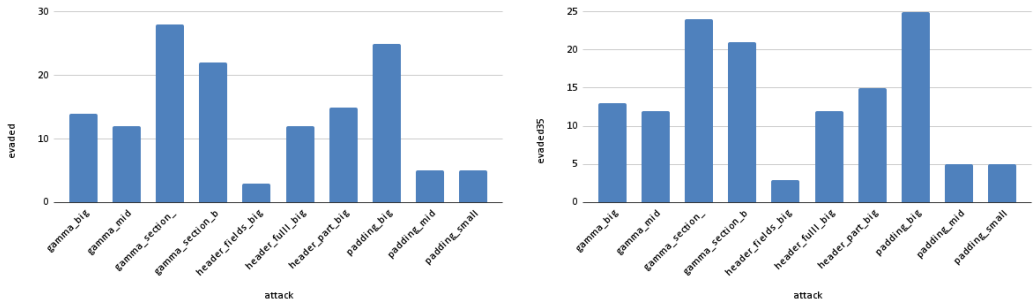


Figure 1. Demonstrates total amount of malware detectors that marked initial sample as malicious, but was not able to identify at least one mutation as malicious within all populations of samples and within first 35 mutations

periments secml-malware python library [7] was utilized. Library contains implementation of multiple attacks mentioned in section 3. Secml-malware also contains MalConv pre-trained on Ember dataset [2].

4.1 Limitations

VirusTotal [1] limits daily API usage on free accounts. Additionally, conceivably accounts activities are monitored to prevent malicious actors abusing service to generate undetectable mutation of the malware. Combination of these constrains limited ability to conduct profound and statistically sound experiments.

4.2 Setup

MAX malware detection solution was selected, as it was one of the products utilizing machine learning and providing maliciousness score as an output. Decision boundary appears to be either 1 or 10, since minimal found value is 11. Attacks participated in the testing trails are: gamma padding [10], gamma section [10], padding, partial DOS header modification [9], full DOS header modification [8], header fields, shift and extend [8]. Attacks were tested with random malware sample from VirusTotal [1]. The samples were classified into three categories based on their size, namely, large (approximately 1MB), medium (approximately 500KB), and small (approximately 10KB).

4.3 Results

Despite the fact that testing sample is too small to draw any conclusions there are still some indicators of specific trends for further research. Firstly, despite the fact that 100% of black-box padding attack mutations

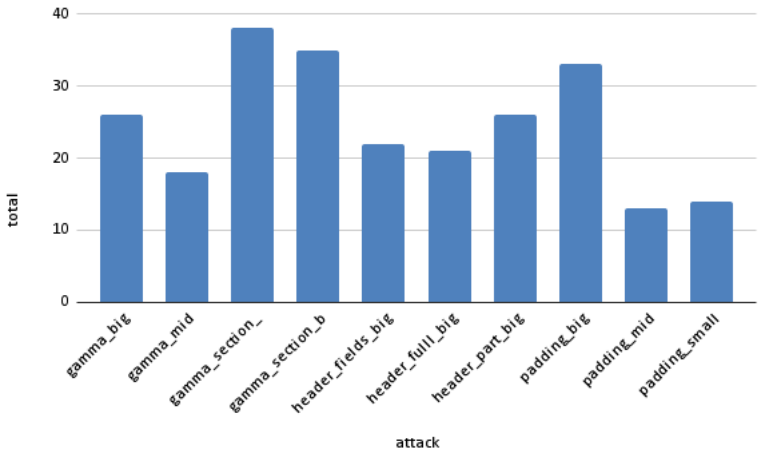


Figure 2. Illustrates total amount of malware detectors that was not able to identify at least one mutation as malicious including original sample

were detected in the original work [7] against MalConv trained on [2], it was the only algorithm that was misclassified by MAX. Moreover result was achieved after first mutation. Short trails utilizing rest of adversarial attacks allowed to achieve misclassification as well. Another interesting insight was that gamma section attack is significantly outperformed other attack strategies. Without considering outlier sample, every trial except gamma section with 500KB sample, showed score within range between 100 and 80. On the other hand gamma section scores were within range between 60 and 69. Additionally, gamma section showed better evasion rates comparing to other types of attacks, which can be seen in figures 1 and 2.

5 Conclusion and future work

This paper covered recent black-box adversarial attacks in the domain of malware detection. Some of the covered attacks tested against vendors solutions. As a future work it would be interesting direction to test performance of state-of-the art attack in conditions with higher query limit and to include attacks that affect behavior of the program, e.g. by appending irrelevant API system calls [11].

References

[1] Virustotal. `https://www.virustotal.com/`.

[2] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

[3] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 3–14, 2017.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[5] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[6] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[7] Luca Demetrio and Battista Biggio. Secml-malware: Pentesting windows malware classifiers with adversarial exemples in python. *arXiv preprint arXiv:2104.12848*, 2021.

[8] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Alessandro Armando, and Fabio Roli. Adversarial exemples: Functionality-preserving optimization of adversarial windows malware. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[9] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. Explaining vulnerabilities of deep learning to adversarial malware binaries. *arXiv preprint arXiv:1901.03583*, 2019.

[10] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Transactions on Information Forensics and Security*, 16:3469–3478, 2021.

[11] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. In *Data Mining and Big Data: 7th International Conference, DMBD 2022, Beijing, China, November 21–24, 2022, Proceedings, Part II*, pages 409–423. Springer, 2023.

[12] Kaspersky. Machine learning for malware detection. `https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf`, 2021.

[13] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Adversarial malware binaries: Evading deep learning for malware detection in executables. In *2018 26th European signal processing conference (EUSIPCO)*, pages 533–537. IEEE, 2018.

[14] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[15] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[16] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5):909, 2019.

[17] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. Malware detection by eating a whole exe. *arXiv preprint arXiv:1710.09435*, 2017.

[18] SentinelOne. Machine learning with a little magic on top! `https://www.sentinelone.com/blog/machine-learning-little-magic-top/`, 2018.

[19] Daniel Stepanic and Andrew Davish. Linux malware protection in elastic security. `https://www.elastic.co/blog/linux-malware-protection-in-elastic-security`, 2022.

[20] Rabia Tahir. A study on malware and malware detection techniques. *International Journal of Education and Management Engineering*, 8(2):20, 2018.

Approahes to Accelerate Mesh Deformation in Practical Situations

Fan Yuanhao
yuanhao.fan@aalto.fi

Tutor: Mauranen Henry

Abstract

This paper introduces an overview of recent articles that are dedicated to accelerating the mesh deformation process. Mesh deformation plays an important role in computer graphics. Therefore several approaches have been devised to accelerate the mesh deformation process. The paper mainly discusses two classes, mesh simplification concerning methods, and transform matrix modification. The paper also considers some real-life, especially medical-concerned application topics of mesh deformation, and explains how these approaches improve the results under these situations.

KEYWORDS: Mesh Deformation, Laplace coordinates, medical application, vertex contracting, Hierarchical Progressive Meshes.

1 Introduction

Deformation is always a subject undergoing intense study in computer graphics (CG). The construction of 3D models emerges after the appearance of the technology of 3D-CG, and develops with 3D-CG or 3D-CGI (Computer Graphic Imagery), thus promoting the so-called Computer-Aided Modeling(CAM) and Computer-Added Design(CAD). In practice, 3D Models are always represented as meshes, i.e. approximated with trian-

gular facets on the surface, called surface meshes, and tetrahedral mesh in the interior, called volumetric meshes. One important topic of these meshes is mesh deformation, which concerns the study of how to represent the deformation of the soft body with the mesh.

In this study, we start with the knowledge of Laplacian Coordinates(LCs), one of the fundamental mathematical tools for deformation simulation. The study then presents some modifications of the standards key-points based deformation simulation technology. There exist mainly two different approaches. One focuses on mesh simplification, and one focuses on the improvement of deformation representation. The study then discusses the result of these approaches and gives some application scenarios.

This paper is organized as follows: Section 2 describes the backgrounds, Section 3 provides the selected Method, Section 4 demonstrates the Results of the survey, Section 5 presents Discussion of the overall survey outcome, and ends with Section 6 Conclusion.

2 Background

This section will introduce the fundamental process of Deformation with Laplacian Coordinates(LCs). The idea is developed by Sorkine, Olga as in [8, 7]. Uwe Hahne and Andrew Nealen further developed this method as in [3, 5], and the synthesized method was well-represented by Tao Ju in the course CSE 554 of Washington University St.Louis (wusl) as presented below.

In Practice, mesh deformation defines key points, and implements the process of deformation by fitting other points according to these key points. In most practical situations, several points in the original mesh called key points will be assigned to a new position. This happens, for example, when the user drags some points in the mesh with the mouse. To simulate the soft body deformation, the other points also need to change their position to make a smooth transition. One simple but fundamental scheme is just to first calculate the transform matrix and then applied it to all the points. The paper will introduce this method further in Background Section. However, it is always too slow to simply adapt this direct-matrix-applying method.

Suppose the source mesh and the target mesh have been aligned, the problem could be set up with input: n source points $p_1, p_2, ..., p_n$ and the target location of m handles/key points $q_1, ..., q_m$ $m \leq n$, and output: de-

formed locations of all the source points $p'_1, ..., p'_n$. The desired p'_i could be seen as the one that minimizes two terms: E_f fitting term measuring how close the deformed source is to the target and E_d distortion term measuring how the source shape is changed. While the former could be simply written as $E_f = \sum_{i=1}^{m} ||p'_i - q_i||^2$, the latter need more consideration.

To measure the degree of distortion, Tao introduces a "Laplacian" to describe the bumpiness at each vertex. This Laplacian Coordinates is a linear operator that maps vectors[15, 8] to vectors and is defined below:

$$L[\mathbf{p}_i] = \mathbf{p}_i - \frac{1}{|N_i|} \sum_{j \in N_i} \mathbf{p}_j \tag{1}$$

where N_i are indices of neighboring vertices of $\mathbf{p}_i$. Let $\boldsymbol{\delta}_i = L[\mathbf{p}_i]$, the distortion term could be written as $E_d = \sum_{i=1}^{n} ||L[\mathbf{p}'_i] - \boldsymbol{\delta}_i||^2$ Thus we can rearrange the two items into a form of $\min \sum_{i=1}^{k} ||\mathbf{a}_i^T \mathbf{x} - b_i||^2 = ||\mathbf{A}\mathbf{x} - \mathbf{B}||$. At this stage, we can easily solve it with partial derivatives, formulated as $\mathbf{x} = (\mathbf{A^T A})^{-1} \mathbf{A^T B}$

It is noticeable that the LC operator is invariant under translation, but not invariant under rotation and scaling. So we need to add another transforming term T_i to the original Laplacian vectors before comparing them to the deformed Laplacians like $E_d = \sum_{i=1}^{n} ||L[\mathbf{p}'_i] - \mathbf{T}_i \boldsymbol{\delta}_i||^2$. It is also noticeable this Laplacian coordinator operator is a differential element of the famous Laplacian Operator $\Delta = \nabla^2$

3 methods

This survey mainly contains articles from IEEE. The paper referenced by this survey is firstly on the problem of mesh deformation and secondly satisfied the 15-year limit, i.e. from 2010 till now. With a belief that every scientific research matters, this survey paper does not restrict the impact factor of the articles, but only concerns the relativity of the keywords e.g. mesh deformation and vertex contracting. The paper is selected either from the IEEE official website with Xplore or from the reference made by the former, preferrable with a higher citation rate and excludes the ones with deep mathematics such as Finite Element Methods(FEM), or with Artificial Intelligence(AI), Deep Learning (DL) added to narrow the scope. A detailed survey and summary were made from these articles, thus selecting innovative accelerate approaches.

Based on the steps in which the methods are applied, the methods are roughly classified into 2 categories, mesh simplifications, and deformation

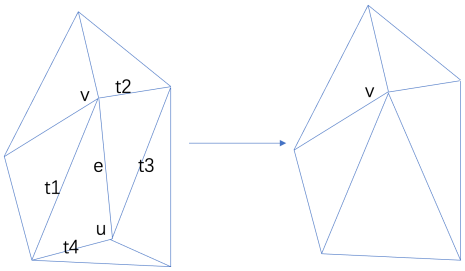


Figure 1. Figure 3.3.1

representation. The process of mesh deformation could be generalized in 3 steps First, construct the mesh, second, solve the mathematical representation of the deformation based on some key points, last, apply this representation to all the points in the mesh and therefore get the deformed mesh. Thus a simple problem-solving structure is constructed, though there is still space to speed. To accelerate, two main positions exist, after the first step, simplify the mesh, and during the second step, modify the deformation representation according to the simplified mesh.

4 Results

In this section, the paper analyses the approaches used in the 10 articles, filtered from approximately 20 articles selected via the method described above. The approaches are categorized and organized as mentioned, firstly the mesh simplification and then the deformation representation. Some impressive approaches are highlighted in the corresponding sections and others might be mentioned briefly. The last part of this section will introduce the application of several approaches under their original practical scenery, and evaluate their performance.

4.1 Mesh Simplification

Essentially mesh simplification methods are edge collapse, which eliminates certain edges in the original mesh and therefore simplifies the mesh. This method was utilized in [1, 4] and successfully reduced the computation cost. Figure 3.1.1 shows a simple model of edge collapse. Here the edge e and the vertex u are collapsed, and therefore edges $t3$ and $t4$ are also collapsed, resulting in a more simplified mesh. The problems continued as to how to select the less essential edge of the mesh, since un-

der certain conditions the elimination of several critical edges would lead to the collapse of the whole mesh. Here Shungang Hua in the paper [4] gives an important measurement as shown below:

$$\lambda_u = \sum_{t \in (T(u) - T(uv)} \max_{f \in T(uv)} \{1 - t_n \cdot f_u\}$$

where $T(u), T(uv)$ is the set of triangles containing vertex u, and both$u$ and v respectively, while t_n, f_n are the normals of triangle t, f, respectively. Since $1 - t_n \cdot f_u$ denotes the degree of coplanar of two triangles, the more coplanar the two facets are, the smaller λ_n is, and the more trivial replacing these plans is. [1] uses simply Hausdorff distance of the edge to measure the importance of this edge with the formula shown below:

$$\Delta(\boldsymbol{v}) = \sum_{\boldsymbol{p} \in \text{planes}(v} (\boldsymbol{p}^T \boldsymbol{v})^2$$

where $\boldsymbol{p} = [a, b, c, d]^T$ represents the plane defined by the equation $ax+by+cz+d = 0$ where $a^2 + b^2 + c^2 = 1$.[2] Moreover, [1] found the simplification process could be abstracted as a sequence shown below. Thus [1] applies a CPM(Critical path method) algorithm with three restrictions to assure there will be no elimination of critical edges.

$$M = M_0 \overset{ecol(v_1)}{\longrightarrow} M_1 \overset{ecol(v_2)}{\longrightarrow} ... \overset{ecol(v_1)}{\longrightarrow} M_n = M_{simplified}$$

The restrictions of [1] are namely (a) at most two vertices can be collapsed at once, (b) For all edge $e = (v_1, v_2)$ that will be collapsed, and any vertex w that is connected to both v_1 and v_2, triple (v_1, v_2, w) must define a valid triangle, and (c) For all edge $e_1 = (v_1, v_2)$ that will be collapsed, and any edge $e_2 = (w_1, w_2)$ forms a quadrilateral (v_1, v_2, w_1, w_2) e_1 and e_2 cannot be collapsed at once.

Jingui Pan in [6] provides another hierarchical approach for mesh simplification. Firstly, the paper constructs a dual map of the original map(the dual map is basically changing the surface of the original map into vertices and edges into lines) to store the connectivity information. Secondly, Jingui Pan in [6] defines three measurements to describe the consequences of the collapse of one edge, E_{fit}, planarity (similar to coplanarity in [4]), E_{dir}, orientation variety, and E_{shape}, overall shape regularity. The total error is defined as $E = E_{fit} + \alpha_1 E_{dir} + \alpha_2 E_{shape}$ with two user modifiable parameter α_1, α_2. It is also noticeable that [6] also add constructed a tree of the collapsed vertices in the simplification process. The tree is useful for the construction of multi-Level Detailed Model, which is a

data structure storing different clustering-level meshes that is used for the further acceleration of deformation implementation. Thus the process is called Hierarchical Progressive Meshes (HPM) for the hierarchical, i.e. tree-like structure.

Multiple simplification schemes can be used as shown in [14]. [14] adapts an iterative binary space partition to obtain surface clustering. The criterion to select the meshes forming a cluster mainly contains the normal variation (identical to coplanar in [4]) and geometry variation. The latter is computed with three steps: firstly, compute covariance matrix $\mathbf{C} = \sum_{\mathbf{P}_i \in \Omega} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T$($\Omega$ the partitioned region, $\bar{\mathbf{p}}$ the centroid of Ω) secondly, do the eigenvalue of $\mathbf{C}$ with the greatest three eigenvalues $\lambda_0, \lambda_1, \lambda_2$ and lastly compute geometry variation $\sigma_g = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$.

4.2 Deformation Representation

As Tao Ju illustrated in the Background Section, since the Laplacian coordinate operator is not invariant under rotation and scalar, another operator is needed to assist deformation. This assistant operator would be plausible if it is linear, therefore it can be written as a matrix. There are different approaches to constructing the desired matrix [4, 11]used affine matrix, with an explicit method and an implicit method respectively. This paper will introduce the main idea of both of them.

[4] introduced an explicit affine matrix construction, based on the unknown rotation and translation matrix, while the whole mathematical derivation could be slightly laborious. Suppose the unknown rotation matrix R_i is a 3×3 matrix for the ith key point, with the form of

$$\mathbf{R}_i = \begin{pmatrix} r_{11}^i & r_{12}^i & r_{13}^i \\ r_{21}^i & r_{22}^i & r_{23}^i \\ r_{31}^i & r_{32}^i & r_{33}^i \end{pmatrix} = (\mathbf{c}_1^i, \mathbf{c}_2^i, \mathbf{c}_3^i) \tag{2}$$

. and the unknown translation vector for the ith key point is 3×1 vector defined as $\mathbf{T}_i = (t_1^i, t_2^i, t_3^i)$. Then we can construct two terms i.e. regularization term E_{reg} realizing smooth translation and rotation term E_{rot} realizing rotation, where

$$\begin{aligned} E_{reg} &= \sum_{i=1}^{m} \sum_{k \in N(i)} ||\mathbf{R}_i(\mathbf{P}_k - \mathbf{P}_i) + \mathbf{P}_i + \mathbf{T}_i - (\mathbf{P}_k + \mathbf{T}_k)||_2^2 \\ E_{rot} &= \sum_{i=1}^{m} ((\mathbf{c}_1^i \cdot \mathbf{c}_2^i)^2 + (\mathbf{c}_1^i \cdot \mathbf{c}_3^i)^2 + (\mathbf{c}_2^i \cdot \mathbf{c}_3^i)^2 \\ &\quad + (\mathbf{c}_1^i \cdot \mathbf{c}_1^i - 1)^2 + (\mathbf{c}_2^i \cdot \mathbf{c}_2^i - 1)^2 + (\mathbf{c}_3^i \cdot \mathbf{c}_3^i - 1)^2) \end{aligned} \tag{3}$$

where $N(i)$ denote the neighbour of i. m is the total number of key points, $\mathbf{P}_i = (p_1^i, p_2^i, p_3^i)^T$ is the position vector of the ith key point The goal is to minimize the weighted sum of these two terms, i.e.

$$\min_{R_1,T_1,...,R_m,T_m} \alpha E_{reg} + \beta E_{rot} \tag{4}$$

where α, β are user defined parameters. Let $f(\mathbf{x})$ satisfy $f(\mathbf{x})f(\mathbf{x})^T = \alpha E_{reg} + \beta E_{rot}$, where $\mathbf{x} = (r_{11}^i, r_{12}^i, r_{13}^i, r_{21}^i, r_{22}^i, r_{23}^i, r_{31}^i, r_{32}^i, r_{33}^i, ..., t_1^i, t_2^i, t_3^i)^T$ $12m$ items, $f(\mathbf{x})$ must be $3 \times \sum_{i=1}^{m} |N(i)| + 6m$ entries vector. Recombining all quadratic terms, $f(\mathbf{x})$ could be shown as below

$$\begin{aligned} f(\mathbf{x}) =& (\phi_1^{ij}(\mathbf{x}), \phi_2^{ij}(\mathbf{x}), \phi_3^{ij}(\mathbf{x}), ..., \\ & \psi_1^i(\mathbf{x}), \psi_2^i(\mathbf{x}), \psi_3^i(\mathbf{x}), \psi_4^i(\mathbf{x}), \psi_5^i(\mathbf{x}), \psi_6^i(\mathbf{x}), ...)^T \end{aligned} \tag{5}$$

where $i \in [m], j \in N(i)$, and

$$\phi_k^{ij}(\mathbf{x}) = \mathbf{R}_i(\mathbf{P}_j - \mathbf{P}_i) + \mathbf{p}_k^i + \mathbf{t}_k^i - (\mathbf{p}_k^j + \mathbf{t}_k^j) \tag{6}$$

$$\begin{aligned} \psi_1^i(\mathbf{x}) = \mathbf{c}_1^i \cdot \mathbf{c}_2^i, & \psi_1^i(\mathbf{x}) = \mathbf{c}_1^i \cdot \mathbf{c}_3^i, \\ \psi_3^i(\mathbf{x}) = \mathbf{c}_2^i \cdot \mathbf{c}_3^i, & \psi_1^i(\mathbf{x}) = \mathbf{c}_1^i \cdot \mathbf{c}_1^i - 1 \\ \psi_1^i(\mathbf{x}) = \mathbf{c}_2^i \cdot \mathbf{c}_2^i - 1, & \psi_1^i(\mathbf{x}) = \mathbf{c}_3^i \cdot \mathbf{c}_3^i - 1 \end{aligned} \tag{7}$$

synthesis of all the equations above the optimization goal is modified to $\min_x F(\mathbf{x}) = f(\mathbf{x})^T f(\mathbf{x})$, which is a non-linear least-squares problem. It could be solved by the Gaussian-Newton algorithm, i.e. approximate with the linear term of Taylor expansion and obtain a solution via iteration.

The approaches above introduce a feasible solution, while it is obvious that it cost a lot in mathematical derivation and computation. [11] provides another method that calculates the affine matrix implicitly, with the main idea to adapt the minimum properties in its definition. As we can see in the formula introduced in the background $\tilde{E}(V') = \sum_{i=1}^{n} ||L(\mathbf{v}_i') - \mathbf{T}_i \delta \mathbf{A}_i||^2 + \sum_{i \in H}^{n} ||\mathbf{v}_i' - u_i||^2$(where H is the handle region and u_i are key points while v_i all the points in ROI, L Laplace coordinate position and $\delta \mathbf{A}_i = L(u_i)$, v' denotes deformed vertices). The goal is to obtain T. Considering the T should be the smallest matrix to transform all the points and the fact that the least square can deduce a linear function, T should satisfy

$$\mathbf{T}_i = \arg\min_{\mathbf{T}_i} (||\mathbf{v}_i' - \mathbf{T}_i \mathbf{v}_i||^2 + \sum_{j=1}^{k} ||\mathbf{v}_{i_j}' - \mathbf{T}_i \mathbf{v}_{i_j}'||^2) \tag{8}$$

where $\mathbf{v}_{i_j}'$ denotes the jth neighbour of v_i. With the introduction of the Laplacian Coordinates(LC) operator, the translation component could be

omitted since the LC operator is translation invariant. For the fact that anisotropic scaling is rare and complex, the affine matrix could be written as

$$\mathbf{T}_i = \begin{bmatrix} s & -h_3 & h_2 & 0 \\ h_3 & s & -h_1 & 0 \\ -h_2 & h_1 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

. with s scaling coefficient and h_i rotation coefficients. Solving optimal $\mathbf{T}_i$ is equivalent to solve optimal vector $(s_i, h_{i1}, h_{i2}, h_{i3})$. Get the partial derivative of the vector, therefore, reached the solution

$$(s_i, h_{i1}, h_{i2}, h_{i3})^T = (\mathbf{A}_i^T \mathbf{A}_i)^{-1} \mathbf{A}_i^T \mathbf{b} \tag{10}$$

where

$$\mathbf{A}_i = \begin{pmatrix} v_{jx} & 0 & v_{jz} & -v_{jy} \\ v_{jy} & -v_{jz} & 0 & v_{jx} \\ v_{jz} & v_{jy} & -v_{jx} & 0 \\ \dots & \dots & \dots & \dots \end{pmatrix}, j \in \{i\} \cap N_i \tag{11}$$

$$\mathbf{b} = \begin{pmatrix} v'_{jx} \\ v'_{jy} \\ v'_{jz} \\ \dots\dots \end{pmatrix} j \in \{i\} \cap N_i \tag{12}$$

Similar formula as [4] are derived in [14], written

$$\min \sum_{j \in N_s(i)} w_{ij} ||\mathbf{R}_i \mathbf{p}_j + \mathbf{t}_i - \mathbf{p}_j||^2 \tag{13}$$

but here the user could use $N_s(i)$ to parameterize s-ring neighbor of ith vertex and w_{ij} the weight of the neighbor to control the degree of the rigidness of the mesh deformation. In HPM [6], the multi-Level Detail model also adapts a weight in the mesh transformation. HPM needs to traverse from the root to construct the multi-Level Detail model, and there are weights related to each cluster to control the resolution. Only if encountered with those clusters with larger weights than a predefined threshold will the algorithm go to the submesh of these nodes, otherwise the deformation will apply to only this simplified mesh. These two approaches further accelerate the deformation progress.

Another simple but effective deformation method is the normal-alignment method used in [12, 13]. In circumstances where the normal of a triangle facet is known as $\mathbf{n}_i$ and the intersection of the norm and the facet is

known as $\mathbf{p}_i$, one can move the extra $\hat{\mathbf{v}}$ on the boundary of facet to minimize $\frac{\mathbf{n}_i \cdot (\mathbf{p}_i - \hat{\mathbf{v}})}{||\mathbf{n}_i|| \cdot ||(\mathbf{p}_i - \hat{\mathbf{v}})||}$, the resulting $\hat{\mathbf{v}}$ would be perpendicular to the facet and therefore the triangle facet would divide into two and the shape can be deformed

4.3 Applications

Dongdong Zhou in [15] designed the experiment mainly on palm deformation. The project meshes the whole human body and focuses on the challenge of controlling the huge number of nodes/points to implement mesh deformation. The program used the standard Laplacian Deformation discussed previously and applied dedicated data structures to store the topological information. The test scenario was designed as examining the rotation of the palm. A detailed comparison between different numbers of control points, the deformed effect, and the iteration time was recorded. In this scheme, with a topological structure stored for each tetrahedral, 8 control point is enough to produce plausible effects.

[1, 6] adapted only the mesh simplification modification, and their result is quite impressive. Sha Chenming et. al in [1] successfully applied their edge collapse method to simulate a deformed robot cat and a walking dinosaur, while Jingui Pan et. al in [6], as shown above, utilize the HPM with dual map. They also designed a comparison between their modified method with original vertices and some other deformation methods. While the challenge concerns the deformation of Horse, Armadillo, and Dragon meshes, the original vertices methods took thousands of milliseconds, and their methods 177.90 ms for the most complicated Dragon meshes, achieving interactivity and real-time performance especially for large-scale models.

[14, 4] used both mesh simplification and modified deformation representation to accelerate. Yong Zhao in [14], as mentioned in section 4.2, parameterized the stiffness/rigidness of the body by controlling the size of the s-ring neighborhood in the formula (13). Meshed of a Bar and a House were designed for deformation tests and the results showed the larger the size, the more stiff the mesh. Shungang Hua et al. in [4] test their affine matrix methods along with edge collapse methods via camel and cat meshes, containing 48485 and 7007 vertices respectively. However, the simplification method reduces the size to 200 key points under both circumstances and succeeded in simulation leg-moving, head-turning and model-stretching. This indicated the two methods can extract and express

the feature of the original shape effectively.

5 Discussion

The experiments are confined to parts of human bodies, yet the results are plausible enough. It is easy to see that the application scenarios contain mainly body parts, such as teeth, faces, and palms, and under these circumstances, the modified deformation scheme is sufficient for real-time situations. While in the 3D reconstruction field, the mesh deformation is also competent in detailed 3D modeling.

The technique required for the overall movement of the human body, such as jumping, running, playing basketball, or martial arts, is another topic of deformation. However, these topics containing more than minor deformation require not only pure mathematical calculation but also approaches including skeleton construction, and sometimes aided from deep learning with neural networks as shown in projects like [10, 9] and After an in-depth study of these articles, the future study may concern more about the overall human body movement

6 Conclusion

This paper gives a literature review of several projects concerning methods that support accelerated mesh deformation. This paper roughly classified the approaches into two categories i.e. mesh simplifications and deformation representation. The former contains mainly the edge collapse method with a brief introduction to hierarchical progressive meshed(HPM) and bipartition clustering. The latter contains mainly explicit and implicit affine matrix calculation and a brief mention of normal vector alignment. The application results of all of these approaches are satisfiable enough and greatly save computation time resulting in the support of real-time deformation. However, the scenarios are limited in minor deformation mainly on body parts. For more complicated deformation such as body movement, another scheme containing skeleton construction and deep learning may need to adapt.

References

[1] Sha Chenming, Zhang Xiaojing, and Yue Yajie. 3d meshes deformation based on mean value coordinates. In *International Conference on Software Intelligence Technologies and Applications and International Conference on Frontiers of Internet of Things 2014*, pages 288–291, 2014.

[2] Michael Garland and Paul Heckbert. Surface simplification using quadric error metrics. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 1997, 07 1997.

[3] Uwe Hahne. Weighting in laplacian mesh editing, 2006.

[4] Shungang Hua, Qing Zhong, and Qiuxin Jiang. Direct manipulation of 3d mesh deformation. In *2010 Third International Symposium on Information Science and Engineering*, pages 202–206, 2010.

[5] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. GRAPHITE '06, New York, NY, USA, 2006. Association for Computing Machinery.

[6] J. Pan and W. Lu. An adaptive deformation method based on hierarchical progressive meshes. In *International conference on Networking and Services*, pages 375–380, Los Alamitos, CA, USA, mar 2010. IEEE Computer Society.

[7] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, page 175–184, New York, NY, USA, 2004. Association for Computing Machinery.

[8] Olga Sorkine. Laplacian mesh processing. *Eurographics (State of the Art Reports)*, 4(4), 2005.

[9] Sebastian Starke, Ian Mason, and Taku Komura. Deepphase: Periodic autoencoders for learning motion phase manifolds. *ACM Trans. Graph.*, 41(4), jul 2022.

[10] Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. Neural animation layering for synthesizing martial arts movements. *ACM Trans. Graph.*, 40(4), jul 2021.

[11] Jin Sun, Yu Ding, Zedong Huang, Ning Wang, Xinglong Zhu, and Juntong Xi. Laplacian deformation algorithm based on mesh model simplification. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 209–213, 2018.

[12] Ryo Tamura, Seiya Ito, Naoshi Kaneko, and Kazuhiko Sumi. Towards detailed 3d modeling: Mesh super-resolution via deformation. In *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–6, 2020.

[13] Genyuan Xia and Li Chen. 3d dental mesh repairing using template-based deformation. In *2014 7th International Conference on Biomedical Engineering and Informatics*, pages 410–414, 2014.

[14] Yong Zhao. Scalable mesh deformation with controllable stiffness. In *2013 International Conference on Computer-Aided Design and Computer Graphics*, pages 449–450, 2013.

[15] Dongdong Zhou, Xiaobing Chen, Chuangchuang Zhang, Shuxin Guo, Khadka Ashim, and Jianchu Lin. Research on topological deformation of 3d human image based on laplace optimization. In *2022 International Conference on High Performance Big Data and Intelligent Systems (HDIS)*, pages 329–332, 2022.

Managing Secrets in Cloud Applications

Zainab Ahmad
zainab.ahmad@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

Storing sensitive information in the cloud can pose inherent risks, making effective secret management crucial for ensuring the security and privacy of such data. This paper examines default secret management approaches used by Docker and Kubernetes, two widely used containerization and orchestration platforms. We explore their built-in security features and discuss best practices for secret management in cloud applications. These practices include secure storage with strong encryption, fine-grained access control, auditing capabilities, usage monitoring, versioning, and the ability to distribute and rotate secrets. Implementing robust security measures and best practices can help organizations mitigate the risks of storing sensitive data in the cloud.

KEYWORDS: cloud, secrets, secret management, Docker, Kubernetes

1 Introduction

In this rapidly progressing era, the use of cloud technologies is integral to many businesses, and the cloud computing market is experiencing rapid growth. The European cloud computing market growth trend [11] is in line with the global outlook for cloud computing, as Gartner predicts

significant growth in worldwide public cloud end-user spending [10]. Efficient and cost-effective storage, processing, and access to data are driving the growth of Cloud infrastructure usage.

As more organizations look to leverage these benefits, Venafi, a cybersecurity company, has reported a significant increase in the number of cloud security incidents over the past year [8]. The 2022 LastPass breach [20], where unauthorized access was gained to customer data, serves as a reminder of the risks associated with storing sensitive data in the cloud. The breach resulted from a DevOps engineer's hacked credentials, highlighting the importance of proper security practices for all individuals with access to sensitive data.

In the realm of Information Technology (IT), ***secrets*** act as keys that secure resources. Such resources include passwords, API keys, certificates, encryption keys, and other types of sensitive data. Secrets are utilized to authenticate access to systems and applications, and safeguard sensitive information in transit or at rest. Various methods are employed in the cloud to safeguard secrets to ensure their confidentiality, integrity, and availability.

In light of the growing reliance on cloud technologies and the increasing number of cloud security incidents, it is imperative for organizations to adopt robust security measures to protect their sensitive information. While third-party solutions provide viable options for managing secrets, the native capabilities of popular containerization and orchestration platforms, such as Docker and Kubernetes, should not be overlooked. Therefore, this paper particularly focuses on examining the default secret management approaches employed by Docker and Kubernetes [5, 7].

This paper is organized as follows. Section 2 presents a general discussion on managing secrets to maintain the security and privacy of information over the cloud. Section 3 discusses Docker secret management. Section 4 discusses Kubernetes and its key features in secret management. Section 5 presents an analysis of the two along with the influence of third-party resources in the context of secret management. Finally, Section 6 presents some concluding remarks.

2 Managing Secrets

Effective management of secrets is crucial for maintaining the security and privacy of information and systems, particularly when utilizing

cloud-based solutions. In order to safeguard data privacy and security, it is important to consider key factors such as protecting data in transit and addressing potential risks posed by cloud service providers. These risks may include:

- Unauthorized access to data
- Insecure APIs
- Identity and access management issues
- Technology vulnerabilities
- Data breaches

It should be noted that cloud-based technologies such as containers, virtual machines, and microservices can also pose risks if not configured or managed properly. The decentralized structure of cloud-based applications can increase the likelihood of vulnerabilities due to the complexity of cloud environments and shared infrastructure. Provisioning errors can result in misconfigured or insecure systems that could be exploited by attackers. Therefore, implementing proper security measures and strategies is essential in mitigating these risks, and protecting data within the cloud. Secrets managers play a critical role in solving some of these problems by securely storing credentials required for accessing sensitive information without exposing them directly to users or applications.

2.1 Secret Management Lifecycle

Secret management involves ensuring full protection at every stage of a secret's lifecycle, from creation to deletion [3, 18]. The four key phases of a secret's lifecycle can be seen in Figure 1.

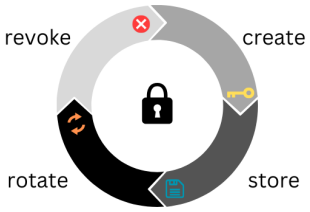


Figure 1. Key phases of a secret's lifecycle.

Creation: During the creation phase, secrets can be manually created by users, such as a password, or automatically generated, such as an encryption key for decrypting a protected database.

Storage: In the storage phase, secrets can be stored in various places in the cloud, including designated secret management services such as AWS Secrets Manager, HashiCorp Vault, or Kubernetes Secrets. They can also be stored in configuration files or environment variables in the application or container images.

Rotation: To improve overall protection, secrets can be changed or reset on a schedule, which is required by many regulations and standards, during the rotation phase.

Revocation: In case of a cybersecurity incident, or when an employee leaves or no longer needs access to certain resources, secrets can be revoked during the revocation phase to prevent unauthorized access to critical systems and data, limit negative consequences and prevent attackers from accessing critical resources.

Throughout each phase, unauthorized access, intervention, and manipulation of secrets must be prevented.

2.2 Microservices

Microservices are referred to as an approach to software architecture that organizes an application into a set of independent services [16]. These services are designed to be loosely coupled, enabling them to be deployed and scaled independently of one another [19]. Each service is focused on a specific business capability and is owned by a small, dedicated team. Characteristics of microservice architecture can be seen in Figure 2. The microservice architecture promotes high maintainability and testability, allowing for the rapid and reliable delivery of large, complex applications.

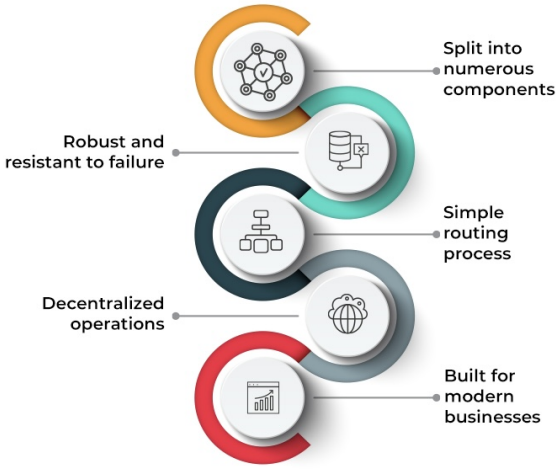


Figure 2. Characteristics of microservice architecture [1].

Containerization technologies, such as Docker, are used to deploy microservices [5]. Container orchestration platforms, such as Kubernetes, can be used to manage the deployment, scaling, and management of these containerized microservices [7]. Docker and Kubernetes are commonly used tools in the implementation and management of microservices-based applications.

2.3 Docker

Docker is an open-source platform that automates the deployment, scaling, and management of applications in containers [15]. Containers are lightweight, isolated environments that package an application and its dependencies together, allowing the application to run consistently on any infrastructure. Docker simplifies the management and maintenance of applications across various stages of development and production, by providing a uniform and repeatable environment for building, testing, and deploying applications. With Docker, developers can create, deploy, and manage applications more efficiently and effectively, reducing the time and effort required to set up and maintain the development environment.

2.4 Kubernetes

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications [7]. It provides a platform-agnostic way to manage containerized workloads and services, enabling teams to run and manage applications on any infrastructure. Organizations of various sizes, ranging from small startups to large enterprises, commonly use the platform to simplify application deployment and management. This results in improved scalability, reliability, and flexibility.

3 Docker Secret Management

Docker is a widely-used platform for containerization, and security is a critical aspect of its architecture. Docker security is based on three components [9]:

- The isolation of processes by the Docker daemon in the user space
- The enforcement of this isolation by the kernel
- The security of network operations.

Docker has several mechanisms in place to secure secrets by default.

3.1 Encrypted Secrets

One of the key methods is through the use of encrypted secrets. Docker allows developers to store secrets, such as passwords, API keys, and other sensitive data, in a separate file that can be encrypted. This file can then be mounted into the container as a read-only file system, ensuring that secrets are not exposed in clear text within the container.

3.2 Using environment variables

While storing configuration in environment variables is a common practice, it is also common to use .env variables for sensitive information [5]. However, this practice has a few drawbacks, such as increased vulnerability to accidental exposure and difficulty tracking access [14]. Debugging can also lead to the accidental printing of the entire collection of .env variables, and secrets can be shared with subprocesses without proper oversight. In order to overcome this problem, Docker developed Docker Secrets.

3.3 Docker Secrets

Storing sensitive data such as private keys becomes crucial when a cluster of containers communicate with each other or with external services. To address this need, Docker has introduced Docker secrets [14]. With Docker secrets, the sensitive data is encrypted and stored in a highly secure and controlled fashion. This information is only accessible by the services that are authorized to use it, and it is never stored in clear text on disk or transmitted over the network unencrypted. Docker secrets can be used to manage secrets within a swarm cluster, allowing to control access to sensitive data across the entire application stack. This feature is especially useful when managing a large number of containers or services, as it simplifies the process of storing, managing, and distributing secrets to the relevant services.

3.4 Docker Swarm

Docker Swarm provides a secure and scalable way to manage secrets for containerized applications [5]. TLS mutual authentication and

encryption are enforced by each node in the swarm, which ensures secure communication between nodes. Users have the choice to use self-signed root certificates or certificates issued by a custom root CA. In addition to using certificates for secure communication between nodes in a Docker swarm, Raft consensus algorithm can be used to establish a leader node in the swarm, which manages the swarm state and ensures consistency across all nodes [12]. The use of certificates also enables swarm locking features, which prevent conflicts and ensure data consistency in distributed systems by allowing only one node to perform write operations at a time. Secrets can be created, updated, and deleted through the Docker CLI or API, and are automatically encrypted and securely stored in the Swarm's Raft consensus store. When a container needs access to a secret, it is securely delivered to the container's runtime environment as an in-memory file system, isolated from the host and other containers. The secrets are automatically removed from the container's memory when the container stops, and access to the secrets can be restricted to specific services or nodes within the Swarm.

Docker Swarm offers a secure and scalable approach to managing containerized applications by utilizing a combination of certificates and Raft. It presents a convenient and reliable method for handling sensitive data in a distributed container environment.

These mechanisms ensure that Docker containers are secure by built-in features, without using any secret management services, whilst protecting sensitive data from unauthorized access and breaches.

4 Kubernetes Secret Management

Kubernetes is responsible for managing confidential data utilized by a cluster, including login credentials, access keys, and encryption keys [7]. It enables a secure and scalable way to manage containerized applications. Secrets can be managed independently of the pods, which are the smallest units that can be created and deployed, and can be made available to the pods as required.

Kubernetes secrets provide multiple mechanisms to enhance the security of its cluster [13]. One of them is ensuring the secure exchange of configuration data between a controller and its workers, such as the communication between a kubelet and a pod. Additionally, Kubernetes

provides an alternative approach for storing sensitive information within the Kubernetes cluster, such as login credentials used to access external applications. This is done through Kubernetes secrets, which are encrypted and only made available to the intended recipient.

By default, Kubernetes secrets are stored in an unencrypted manner in the server of the API. Therefore, some steps need to be taken in order to secure secrets in Kubernetes [7].

4.1 Ecryption at Rest

Kubernetes Secrets are stored in *etcd*, a distributed key-value database used to store configuration data and metadata of a Kubernetes cluster, as plain text and encoded in base64 format. However, storing Secrets in plain text can pose a risk as attackers can easily compromise them, and gain access to the system. Since etcd is not encrypted by default, Secret data needs to be encrypted at rest to prevent sensitive information from being accessed by unauthorized users. Without encryption, an attacker with file system access can read the Secrets. To address this issue, Kubernetes offers an encryption at rest feature that encrypts Secrets before storing them in etcd. This can be studied in detail in the 'Encrypting Secret Data at Rest' section of the Kubernetes documentation [7].

4.2 Configure RBAC rules

Kubernetes Secrets and RBAC, Role Based Access Contol, rules are closely related [13]. To configure RBAC rules, a cluster administrator defines roles, a set of permissions that define a user's or group's access, and role bindings, which can be applied at the cluster level or the namespace level. Roles can be customized to fit specific use cases, and they can include a combination of permissions for different resources. It is important to carefully manage RBAC rules to ensure that users have the appropriate level of access to Kubernetes resources and to prevent unauthorized access to sensitive data. RBAC controls access to creating, modifying, and deleting Secrets and limits it to specific users only.

4.3 Types of secrets

When generating a Secret in Kubernetes, its classification can be designated by using the "type" attribute of the Secret resource. The Secret type is employed to enable automated processing of the Secret

data. Kubernetes comes with various pre-defined types for typical use cases. These types differ in their validation procedures and the restrictions that Kubernetes applies to them. Some of the types, that are mentioned in the 'Secrets' section of the Kubernetes documentation [7], include Opaque Secrets, Basic Auth Secret, TLS Secrets, Dockercfg Secret, SSH Secrets, Bootstrap Token Secrets, and Immutable Secrets.

These methods can be used to accomplish secret management in Kubernetes without using any third-party solutions.

5 Analysis

To have a comprehensive secret management solution, it is essential to have features such as secure storage with strong encryption, fine-grained access control, auditing capabilities, usage monitoring, versioning, and the ability to distribute and rotate secrets. Docker and Kubernetes have several built-in security features to secure secrets.

Docker Swarm enables secure management of Secrets in distributed container environments via TLS mutual authentication and encryption. It supports secret distribution and access control, limiting access to authorized entities. Docker secrets lack versioning; new secrets must be created for any changes. However, rolling updates allow updating services and containers with new secrets. Secrets are encrypted at rest and managed via the Docker CLI or API. While Docker lacks built-in centralized secret management, third-party tools can provide this functionality.

Similarly, Kubernetes uses base64 encoding to store secrets in etcd. It also provides encryption at rest, enabling encryption of secrets before storing them in etcd. It uses RBAC to enforce access control and fine-grained authorization to secrets, enabling cluster administrators to define granular policies that restrict access to secrets based on roles, users, and groups. A secret rotation feature is provided by Kubernetes that enables users to update the contents of a secret without changing the secret name or breaking existing applications. Kubernetes also offers built-in support for centralized secret management through its Secrets API. Secrets can be created and managed at the cluster level, and accessed by individual containers or pods as needed.

Docker and Kubernetes provide a convenient API and client SDKs

for accessing and managing secrets, making it easy for developers to integrate secrets management into their applications. Table 1 aims to provide an overview of the differences between secret management of Docker, Kubernetes, and other secret managers.

Feature	Docker	Kubernetes (k8s)	Vault by HashiCorp	AWS Secrets Manager	Azure Key Vault	Google Cloud Secret Manager
Secret storage	Docker secrets	Kubernetes' etcd storage	Encrypted storage	Encrypted storage	Encrypted storage	Encrypted storage
Encryption at rest	Yes	Yes	Yes	Yes	Yes	Yes
Encryption in transit	Yes, with Docker swarm	Yes	Yes	Yes	Yes	Yes
Access control	Docker Swarm services	RBAC	ACLs	IAM policies	Azure RBAC	IAM policies
Versioning	No	No	Yes	Yes	Yes	Yes
Rotation	No	Yes	Yes	Yes, with Lambda functions	Yes, with Azure Functions	Yes, with Cloud Functions
Centralized management	No	Partially, with central cluster	Yes	Yes	Yes	Yes

Table 1. Secrets Management: Docker vs. Kubernetes vs. Other Secret Managers [5, 7, 4, 17, 2, 6].

In comparison to other secret managers, Docker and Kubernetes offer several advantages, such as tighter integration with containerization platforms and support for RBAC. However, other secret management platforms such as HashiCorp Vault and AWS Secrets Manager offer more comprehensive features such as dynamic secrets, secret revocation, and more advanced access control mechanisms. Choosing the right secret management platform depends on the specific needs of the organization, the level of integration required, and the complexity of the secret management workflow.

6 Conclusion

In conclusion, this paper emphasizes the importance of effective secret management in cloud applications to ensure the security and privacy of sensitive information. The native capabilities of popular containerization and orchestration platforms, such as Docker and Kubernetes, provide viable options for managing secrets, each with its strengths and limitations. However, it is also important to consider third-party solutions in the context of secret management. By adopting robust security measures and best practices for secret management, organizations can mitigate the risks associated with storing sensitive data in the cloud.

References

[1] Hossein Ashtari. *What Are Microservices? Definition, Examples, Architecture, and Best Practices for 2022*. https://www.spiceworks.com/tech/devops/articles/what-are-microservices/. Accessed Feb 2023.

[2] Azure Authors. *Azure Key Vault Documentation*. https://azure.microsoft.com/en-us/products/key-vault/. Accessed April 2023.

[3] Ekran System Authors. *Secrets Management: Importance, Challenges, Best Practices*. https://www.ekransystem.com/en/blog/secrets-management. Accessed March 2023.

[4] HashiCorp Vault Authors. *Manage Secrets Protect Sensitive Data with Vault*. https://www.vaultproject.io/. Accessed April 2023.

[5] The Docker Authors. *Docker Documentation*. https://www.docs.docker.com/. Accessed Feb 2023.

[6] The Google Cloud Authors. *Google Cloud Secret Manager Documentation*. https://cloud.google.com/secret-manager. Accessed April 2023.

[7] The Kubernetes Authors. *Kubernetes Documentation*. https://kubernetes.io/docs/home/. Accessed Feb 2023.

[8] Shelley Boose. *81% of Companies have had a Cloud Security Incident in the Last Year*. https://venafi.com/blog/81-companies-have-had-had-cloud-security-incident-last-year-venafi-research/. Accessed Feb 2023.

[9] Theo Combe, Antony Martin, and Roberto Di Pietro. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3(5):54–62, 2016.

[10] Gartner. *Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly $600 Billion in 2023*. https://www.gartner.com/en/newsroom/press-releases/2022-10-31-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023. Accessed March 2023.

[11] Justus Haucap, Daniel Fritz, and Susanne Thorwarth. *The Economic Impact of Cloud Computing in Europe*. https://www.europeancloudalliance.com/wp-content/uploads/2022/11/Cloud-Computing-in-Europe-fin.pdf. Accessed Feb 2023.

[12] Luc Juggery. *Raft logs on Swarm mode*. https://medium.com/lucjuggery/raft-logs-on-swarm-mode-1351eff1e690. Accessed March 2023.

[13] Eric Kahuha. *Best practices for Kubernetes Secrets management*. https://snyk.io/blog/best-practices-for-kubernetes-secrets-management/. Accessed March 2023.

[14] Allan MacGregor. *The Complete Guide to Docker Secrets*. https://earthly.dev/blog/docker-secrets/. Accessed March 2023.

[15] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal*, 2014:2, 2014.

[16] Chris Richardson. *What are microservices?* https://microservices.io/. Accessed Feb 2023.

[17] Amazon Web Services. *AWS Documentation*. https://aws.amazon.com/secrets-manager/. Accessed April 2023.

[18] OWASP CheatSheets Series Team. *Secrets Management Cheat Sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html#26-secret-lifecycle. Accessed Feb 2023.

[19] Johannes Thönes. Microservices. *IEEE Software*, 32(1):116–116, 2015.

[20] Karim Toubba. *LastPass: Notice of Recent Security Incident*. https://blog.lastpass.com/2022/12/notice-of-recent-security-incident/. Accessed Feb 2023.

Monolithic vs Microservices: A Comparative Analysis of Architectural Approaches for Application Development and Migration

Zainab Khan
zainab.khan@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

In the traditional approach, systems and applications were developed as a single unit, known as the monolith approach. However, this approach resulted in complications as user demands increased with the widespread acceptance of digitization. To address issues such as availability, fault tolerance, responsiveness, and consistency, various companies, such as Amazon, Google, Netflix, and eBay, have shifted towards distributed cloud computing. This involves migrating systems from a monolithic architecture to a microservices-based architecture, which enables modular and independent expansion of horizons. This paper aims to analyze both architectural approaches while outlining their respective advantages and disadvantages. In addition, it will discuss case studies conducted on migration methods and highlight important areas that need to be considered during the migration of any application from a monolithic architecture to a microservices-based architecture.

KEYWORDS: Monolithic architecture, Microservices architecture, Distributed applications, modular approaches.

1 Introduction

In contemporary times, the widespread usage of software-based services and applications has emphasized the importance for system designers to anticipate future demand trends. Selecting an appropriate architecture for the system in question is crucial to withstand high loads while simultaneously maintaining availability and reliability. The two prevalent architectures being discussed currently are monolithic architecture and microservice architecture.

In the context of software architecture, the monolithic approach refers to the implementation of an application in which all functionality is tightly coupled and integrated into a single unit. In contrast, microservice architecture is based on the deployment of multiple independent services, each representing a separate functionality in the system, with independent business logic, data access, and database layers, as illustrated in Figure 1. While the monolithic approach has been the traditional method of building software systems, numerous legacy systems are now transitioning towards a microservice-based architecture due to its benefits over the monolithic approach.

The selection of an appropriate architecture for a system remains dependent on the specific needs and intricacy of the implementation. Both monolithic and microservice architectures have their respective strengths and weaknesses. The development, deployment, and initial testing of monolithic applications are relatively straightforward; however, their maintenance becomes challenging as they grow, and a single error can bring down the entire application, thus raising concerns for scalability, maintainability, and reliability [1]. On the other hand, microservice architecture ensures scalability, maintainability, and reliability due to its loosely coupled nature. Additionally, dividing work among teams assigned to each microservice is convenient. However, this approach has its own drawbacks, including complex deployments, the complexity of communication, the need for more resource allocation for each service, and global testing [1].

This paper aims to provide a comprehensive survey of monolithic and microservice architectural styles, highlighting the challenges and advantages of each approach. In addition, the paper investigates various approaches that legacy systems adopt to migrate from monolithic architecture to microservice architecture.

The structure of the paper is organized as follows: Section 2 presents an analysis of previous research studies, which compares the two architectures. Section 3 identifies several crucial factors that must be considered during the transition from a monolithic approach to microservices. Section 4 provides an in-depth analysis of the survey results. Finally, Section 5 concludes the study.

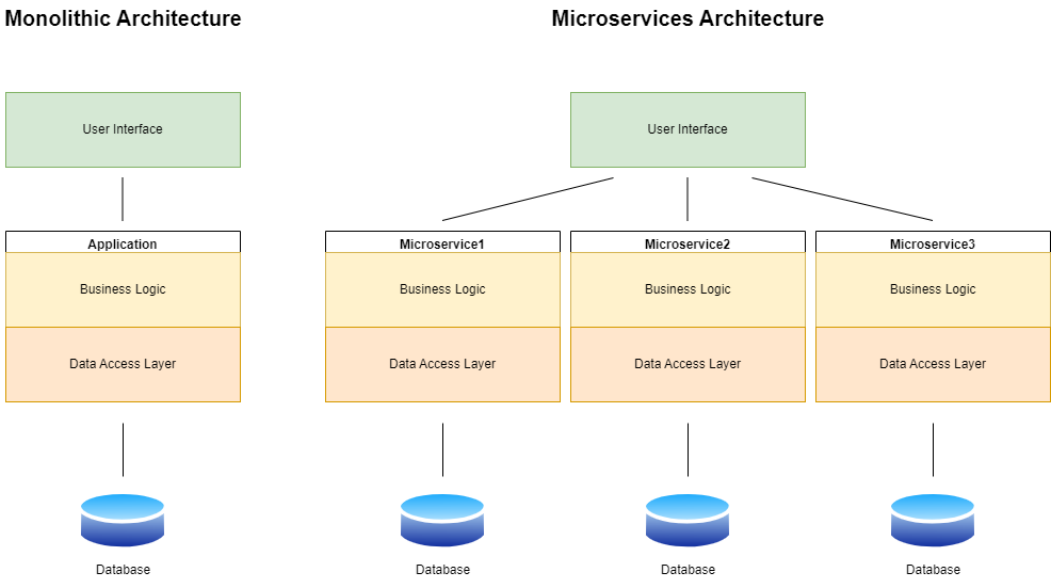


Figure 1. Monolithic VS Microservice architecture

2 Architectural Comparison and Migration Methods Based on Some Experiments

Al-Debagy and Martinek [2] compared microservices with monolithic architecture by evaluating the performance of a development platform under various scenarios and tests. The three testing scenarios were load testing, concurrency testing, and endurance testing. The load test revealed that both architectures exhibit similar performance under average load, but, with small-scale load (i.e., less than 100 users), monolithic application performed better than microservices-based application. On the other hand, concurrency testing showed that the monolithic application exhibited higher throughput (i.e., the requests were handled faster in it). The final test scenario was conducted on two microservices applications having different service discovery technologies (i.e., Eureka and Consul). This test showed that the application with Consul service discovery provided better throughput than Eureka.

One of the main challenges in migrating a monolithic application to microservice architecture is the identification of potential modular candidates to be converted to microservices. Mazlami et al. [3] addressed this issue algorithmically, which consists of three formal coupling strategies namely, logical, semantic, and contributor coupling strategies. These strategies were then provided to a graph-based clustering algorithm to generate suggestions for microservice candidates. They obtained their observations by prototyping the strategies on various open-source projects to assess the performance (execution time) and quality of the generated microservices. With respect to performance, their approach scaled with the size of revision history in logical and contributor coupling. However, the quality metric showed that the proposed model can reduce the team size for a microservice-based implementation to a quarter of the team size in a monolith approach.

Kazanavicius and Mažeika [4] also discussed the migration process of a monolithic system to microservices. They discussed some infrastructure requirements for the migration which should be considered whilst taking the decision. The requirements encompassed continuous integration (CI) and continuous deployment (CD) (i.e., how the microservices will be developed, tested, and deployed), execution environment (containers and cloud technologies), monitoring, and logging. They also discussed various migration methods, each of which had specific models based on different parameters.

The work in [5] compared the two architectural styles, by prototyping on a car sharing application built on Java. They tested the application under two test cases, one with 30,000 requests made by 30 users simultaneously, and the other one with 300,000 requests. The microservice architecture was tested with three forms of replication, i.e., without replication, replicated twice, and replicated four times. The performance metric for the requests (HTTP GET and HTTP POST) showed that monolithic architecture responded to a larger number of requests when the load was light (30,000 requests case), but when the load increased to 300,000 requests, the microservice (with replication of two times) performed better than the rest. The response time metric also showed that the microservice was relatively slower than monolithic under the first test case, but as the requests increased, the microservice architecture outperformed monolithic.

Romani et al. [6] proposed a data-centric approach to migrate legacy systems from monolithic to microservice architecture. The two-phase pro-

cess included microservice identification and refactoring of code. They partitioned the data models from legacy system to more related and cohesive sub-models and packaged them along with their business logics to corresponding number of microservices. Next steps in their work were code refactoring, data migration, and writing Application Programming Interfaces (APIs) to support the newly separated business logic but it was out of the scope of current paper.

3 Considerations and Challenges during Migration to Microservice Architecture

The following are key factors that should be carefully considered by developers before initiating the transition process from monolithic architecture to microservices architecture. [7].

3.1 Division of services

Identification of modules that are capable of independent standalone services is the most fundamental as well as a critical step in migration towards microservice architecture. Technical teams must be sufficiently familiar with the business requirements, domain and scope of the application, and future expectations before converting the existing monolithic application or starting the development from scratch towards microservice-based architecture. New features and functionalities should also be considered towards expanding the microservice-based architecture instead of appending them together in monolithic fashion. This way, all the existing code refactoring as well as new functionalities will promote the adoption of a distributed and loosely coupled architecture, which is the prime goal of microservice architecture.

3.2 Automated Testing

When an application is implemented in microservice-based architectural style, it produces several services which are developed, maintained and deployed independently in production environment whenever they need to be released. This makes it difficult for manual testing approaches to test all the services in their own scopes as well as their integrations with other services. Hence, it is a good idea to move towards automated testing coverage to address the role and feature-specific testing requirements of

each microservice. Therefore, practices, such as continuous integration and continuous deployment go hand in hand to ensure frequent release and maintenance of multiple services on their own timeline.

3.3 Integration with other Services

In a microservice-based application individual teams can opt for different programming languages or technology stacks to implement the service assigned to them. This makes integration of various services together challenging. To avoid this problem, it is recommended that the services should not be tied together through specific technology. Instead, integration should be achieved by technology-independent means of communication, such as Representational State Transfer (REST) APIs and Google Remote Procedure Call (gRPC) APIs [8] based communication mechanisms.

3.4 Automated Deployments

With microservices, multiple deployments can occur per day to the production environment. Traditional deployment processes cannot handle the challenges caused by such scenarios. This raises a need for the usage of tools to automate deployment and management of these services in real-time, such as Gitlab CI/CD [9], Docker [10] and Kubernetes [11]. These tools can help developers in pipelining the deployment into several jobs, as well as containerization of services by making them isolated and free of dependency conflicts. Developers can imitate the production environment locally in containers before releasing a deployment while these tools can also help with load balancing, service discovery and horizontal scaling.

3.5 Monitoring and Logging

Monitoring and logging important events and data is an important aspect of any application and can benefit the organization in the longer run. When different microservices are released independently with their specific functionalities, it becomes challenging to keep their events tracking and logging centralized in order to obtain meaningful information from those scattered logs. The logging should also be searchable and aggregated together to ease the process of fault tracking. Additionally, good, automated tools are required to be put in place to notify exactly which

microservice is causing the failure and which team is required to fix it.

3.6 Fault Tolerance

An application whose execution depends on the proper functioning of all the microservices involved in it must incorporate fault tolerance mechanisms in its design and implementation. Under different circumstances such as heavy load and network congestion, any of the microservice can fail to respond timely. This is where the circuit breaker pattern comes into use. It monitors and counts for failure events taking place in a specified time period and allows microservices to respond according to the load. If the count of failure events exceeds a certain threshold value, the circuit breaker will transit into open state [12]. The circuit breaker returns either the default data or an error to the user immediately, instead of sending more requests to the service, this way, the microservice under severe load can get time to recover. Hystrix [13] is a library commonly used these days to provide latency and fault tolerance in order to ensure resilience in distributed systems.

3.7 Team Structure

To proceed with the development of a microservice-based application, the organization should also structure itself in a similar manner. This means that, having large teams with well-defined roles work well for a monolith product, but with microservices, those teams must split into smaller teams working autonomously.

4 Analysis

This paper presents a comparative analysis of two popular approaches for application development, namely monolithic architecture and microservice architecture. While monolithic architecture has been a long-standing choice for numerous tech companies, it has become increasingly challenging to maintain the applications in a unified monolithic way. As a result, companies have started to migrate their systems towards a microservice-based architecture, which involves breaking down the application into separate, independent modules based on their respective functionalities and feature sets.

The analysis of migration methods and experiments in Section 2 has

revealed that monolithic architecture is suitable for systems with limited functionality and scope for growth, and also when the system is not under constant heavy load. Additionally, it is a suitable choice for companies whose business models are based on large teams handling the complete application development lifecycle, from design and development to testing and maintenance.

As the system under development grows exponentially and demands for newer features increase, monolithic architecture can become inefficient due to its limited scalability. On the other hand, microservice architecture has been proven to be more efficient in such situations, owing to its modular nature. In a microservice architecture-based system, the application is divided into multiple independent microservices, which makes development, deployment, testing, and maintenance significantly easier than a monolithic application. This architectural style is better suited for business models that are structured around having multiple small teams, each responsible for delivering a particular piece of functionality in the system, rather than the entire system.

The migration from monolithic architecture to microservice-based architecture requires careful consideration of various technical, organizational, and infrastructure-based factors. Section 3 of this paper discusses some of these critical factors, such as the assessment of candidacy for service division, automated testing, integrations, deployment, logging, fault tolerance, and team structure. The evaluation of these factors is crucial in determining whether a system should migrate to microservice-based architecture or not. Additionally, modern cloud-based tools and technologies must be used throughout the entire migration process, such as technology-independent APIs, automated testing, continuous integration and deployment (CI/CD), version control systems, and containerization.

5 Conclusion

This paper surveyed the two popular architectural styles of developing applications: monolithic and microservice architecture. The analysis of previous migration studies led us to conclude that the selection of the appropriate architecture cannot be determined by a universal rule; rather, it is highly dependent on the system's requirements and future prospects. Before adopting a specific architecture, companies must define their expectations from the application and align them with their business structure.

Additionally, this paper highlighted various crucial aspects that need to be taken into account when migrating from monolithic to microservice-based architecture. Nonetheless, prior to initiating the actual migration process, several other technical and organizational factors must be carefully observed.

References

[1] C. Harris, "Microservices vs. monolithic architecture," URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith.

[2] O. Al-Debagy and P. Martinek, "A comparative review of microservices and monolithic architectures," in *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 000149–000154, 2018.

[3] G. Mazlami, J. Cito, and P. Leitner, "Extraction of microservices from monolithic software architectures," pp. 524–531, Institute of Electrical and Electronics Engineers Inc., 9 2017.

[4] J. Kazanavičius and D. Mažeika, "Migrating legacy software to microservices architecture," in *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1–5, 2019.

[5] K. Gos and W. Zabierowski, "The comparison of microservice and monolithic architecture," in *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pp. 150–153, 2020.

[6] Y. Romani, O. Tibermacine, and C. Tibermacine, "Towards migrating legacy software systems to microservice-based architectures: A data-centric process for microservice identification," pp. 15–19, Institute of Electrical and Electronics Engineers Inc., 2022.

[7] M. Kalske, N. Mäkitalo, and T. Mikkonen, "Challenges when moving from monolith to microservice architecture," in *Current Trends in Web Engineering* (I. Garrigós and M. Wimmer, eds.), (Cham), pp. 32–47, Springer International Publishing, 2018.

[8] *gRPC: A high performance, open source universal RPC framework*. URL: https://grpc.io/.

[9] *GitLab CI/CD*. URL: https://docs.gitlab.com/ee/ci/.

[10] *Docker: Develop faster. Run anywhere*. URL: https://www.docker.com/.

[11] *Kubernetes for Application Orchestration*. URL: https://montel.fi/kubernetes/?gclid=CjwKCAiAmJGgBhAZEiwA1JZoll4Jq4W5y2nV5Iy6iwcifKU2AKYVYrVPS6E14i_Ef3t2viinnhjMIRoCHhQQAvD_BwE.

[12] C. Dulanga, "Circuit breaker pattern in microservices," URL: https://blog.bitsrc.io/circuit-breaker-pattern-in-microservices-26bf6e5b21ff.

[13] *Hystrix: Latency and Fault Tolerance for Distributed Systems*. URL: https://github.com/Netflix/Hystrix.

Comparative analysis of Container Network Interface (CNI) implementations

Zsombor Takács
zsombor.takacs@aalto.fi

Tutor: Tuomas Aura

Abstract

Containerization is an essential part of cloud software development. The application source code with all its dependencies is packaged into a container, and orchestrator systems, such as Kubernetes, are used to deploy them into a cloud environment. Container Network Interface (CNI) consists of a specification and a set of core plugins (programs) that provide a standard interface for pod network configurations. There are various plugin CNI-compliant implementations and this paper evaluates the most widely used ones: Calico, Cilium, Flannel, Kube-router, and Weave. The analysis concerns technical implementations, performance, and potential use cases. The investigation shows that they provide similar networking models, but differ in packet forwarding and filtering methods. Performance is highly dependent on the chosen networking model, and it can be improved by executing forwarding and filtering decisions closer to the hardware. The analyzed plugins are suitable for general use, but once enterprise-grade features and performance is required, Calico and Cilium stand out. Based on the analysis, CNI plugin technologies are moving towards eBPF implementations for routing and network policies, considering its extended developer audience, security, and flexibility.

KEYWORDS: Kubernetes, Container Network Interface, CNI, Container networking, Containers, Calico, Flannel, Weave, Kube-router, eBPF

1 Introduction

In recent years, software development has adopted various virtualization technologies. Hardware virtualization (i.e., virtual machine) is an effective method to create multiple fully isolated computing environments on the same physical host by full emulation of a physical computer. However, with the advent of microservices and function-as-a-service applications, the need for more lightweight virtualization has emerged [10]. Containers provide virtualization on the operating system (OS) level by sharing the kernel of the host OS. This creates a more lightweight, isolated environment with all the required dependencies encapsulated in one container image. Containers (i.e., deployed images) serve as building blocks of microservice-based applications, and to manage the increased number of containers, an orchestrator system is used, e.g., Kubernetes. Such a system automates numerous tasks, including the deployment of containers across several host machines, application scaling, load balancing, and container self-healing. Containers implement microservices and need to connect to each other and to external networks to form a cloud application. This networking is provided by Container Network Interface (CNI) plugins, such as Calico, Cilium, Flannel, Kube-router, and Weave. There are a large number of plugins, each using different underlying technologies, and the aim of this paper is to analyze the internal design of the most widely used ones and understand their implementation, performance, and potential use cases.

The rest of the paper is structured as follows. Section 2 introduces CNI, Section 3 introduces various design solutions, Section 4 presents a performance comparison, Section 5 discusses possible use cases of the selected plugins, and Section 6 shows the results of the document. Finally, Section 7 concludes the paper.

2 Container Network Interface (CNI)

CNI is a Cloud Native Computing Foundation project that comprises a specification and a set of libraries for writing plugins that offer various solutions for Linux containers to connect to a network. In addition, it provides a set of core reference plugins that serve as a starting point to understand the purpose of CNI.

A CNI plugin is an executable between the container runtime and the container itself. The runtime calls it to execute a network configuration, such as adding a network interface to the container, connecting it to the host network, or assigning an IP address.

The reference plugins are centered around single-host container network interface configuration. For example, the bridge plugin creates a virtual switch in the host network namespace and connects the container to it, thus setting up a connection between the host and the container [1].

When it comes to a cluster consisting of multiple hosts, a number of third-party plugins are available, which advance the capabilities of the reference plugins, providing more extensive, out-of-the-box network solutions for container networking across multiple nodes. The most widely used Kubernetes container runtimes (containerd, CRI-O) make use of CNI plugins to implement the desired container network configuration. In Kubernetes, a pod is the smallest deployable unit of computation that consists of one or more containers sharing the same network namespace.

The specification states directives for container runtime and plugin developers in order to provide a unified method container networking configuration. The specification defines five directives as follows [2]:

1. *A network configuration format*

   The network configuration must be specified in a JSON file where all the necessary plugins along with their configuration are listed. The runtime processes this file and converts it to a format that can be passed to the plugin executable.

2. *A protocol for runtimes to make requests to network plugins*

   It defines a protocol for the communication between the runtime and the plugin executable. It contains protocol parameters that the runtime passes when calling a plugin to execute a certain network configuration, including CNI_COMMAND, CNI_CONTAINER_ID, CNI_NETNS, and

CNI_IFNAME. Furthermore, four basic operations are outlined: ADD, DEL, CHECK, and VERSION, which are denoted by CNI_COMMAND during plugin execution. E.g., the ADD operation means the creation of interface CNI_IFNAME inside container CNI_CONTAINER_ID in network namespace CNI_NETS.

3. *A procedure for runtimes to execute plugins according to the configuration*

   The network configuration JSON file contains a list of plugins to call. The specification defines how the runtime should understand the file and convert it to a form the plugin binary can process. For example, the provided network configuration might translate to a sequence of ADD and DEL operations, and the specification defines the order in which the runtime should invoke these operations. Furthermore, it is forbidden to call parallel operations for the same container.

4. *A procedure for plugins to delegate functionality to another plugin*

   A plugin might call another already existing plugin, for instance, in case of IP address assignment. To manage the IP address allocation of the interface and add the corresponding routes, the IP Address Management (IPAM) core plugin can be called. CNI specifies rules, such as the required parameters the delegating plugin should pass to the delegatee plugin and the way error messages should be forwarded.

5. *Data types for plugins to return their results to the runtime*

   CNI defines the result response format the plugin should return after executing an operation. The specification consists of three values: Success, Error, and Version, and the respective keys to return. For example, after ADD operation the plugin should return the CNI version, the respective interfaces, IP addresses, routes, and DNS information in the JSON formatted result message.

3 Design considerations

This section compares the technical solutions of the most common CNI implementations.

3.1 Layer of operation

Regarding a Kubernetes cluster with several hosts containing multiple pods, communication can be classified into two broad categories: intra-host and inter-host communication [11]. Intra-host communication takes place on the host level, between two or more pods on the same host, whereas in the case of inter-host communication, a pod on one host should be able to reach a pod on another host.

Intra-host communication is simpler since the messages do not leave the hosts where the pods reside. However, there are multiple strategies to achieve it, depending on the layer of operation. Fig.1 (based on [10]) depicts two widely used approaches.

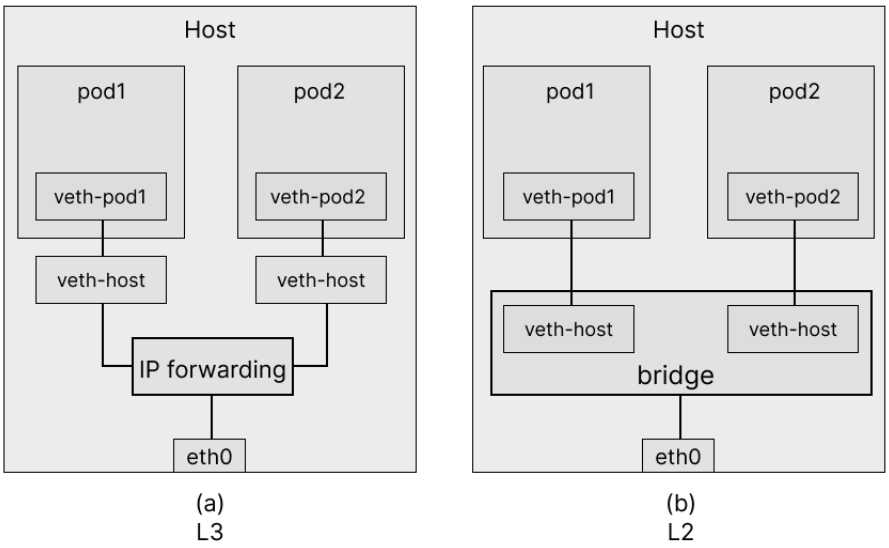


Figure 1. Intra-host pod-to-pod communication approaches

The L3 (Fig.1(a)) relies solely on the routing capability of the host machine and forwards every packet based on the IP address of the pods. The L2 (Fig.1(b)) implements L2 switching capabilities by taking advantage of the bridge reference plugin. In this setup, multiple virtual links, "veth-pairs", are created, where one end is connected to the respective pod, and the other is inserted in the virtual switch. This way, all inter-pod communication traverses through the virtual switch.

Inter-host communication can also be divided into two general categories. Figures 2 and 3 (based on [10]) illustrate underlay and overlay networking modes, respectively. In the case of underlay networking, packets transit via the underlying IP network between the hosts. The routing protocol of choice depends on the data center. However, Border Gateway Protocol (BGP) is commonly used since it can route across different

autonomous systems, which can be desirable for cloud sites. The other widely used approach is to use an overlay network, i.e., a tunnel, on top of the physical network to create virtual connections between the hosts. Essentially, packets are encapsulated using a protocol other than that of the underlying network. The most commonly used tunneling protocols are Virtual Extensible LAN (VXLAN) and IP-in-IP.

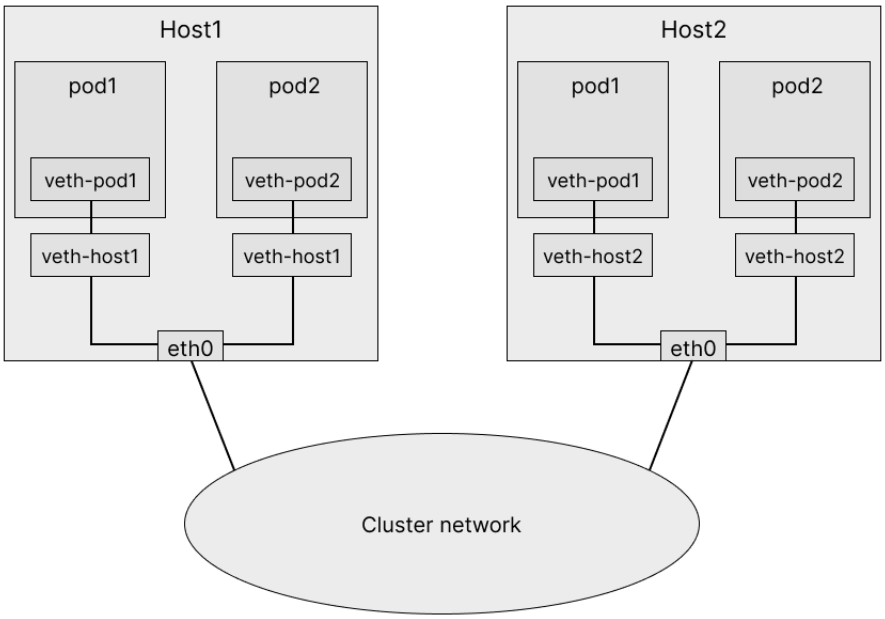


Figure 2. Inter-host pod-to-pod communication via the underlying cluster network

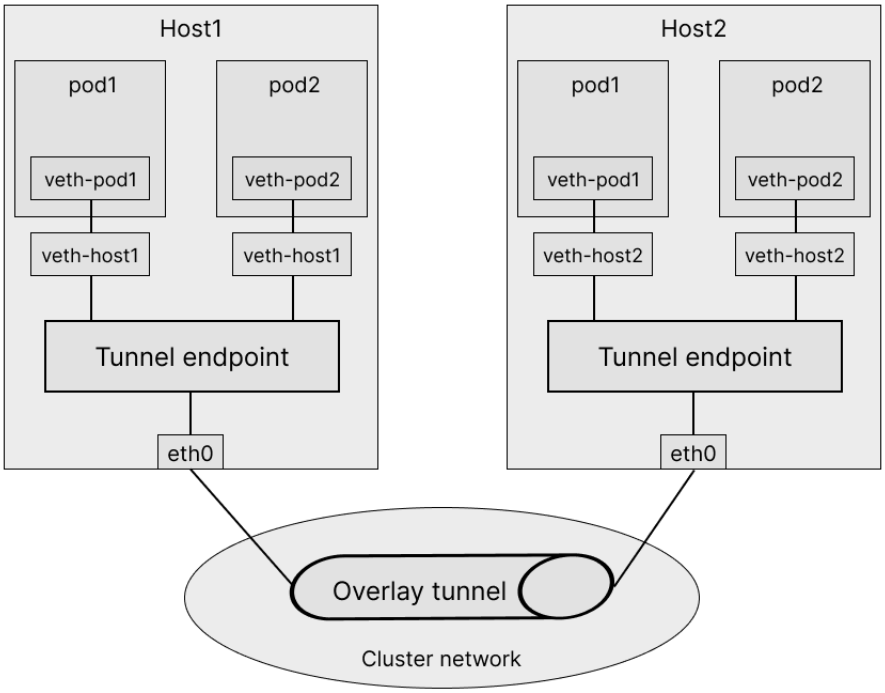


Figure 3. Inter-host pod-to-pod communication via an overlay tunnel

As to third-party plugins, a widely used approach, implemented by Flannel and Kube-router, is to use L2 virtual switching for intra-host communication and support both underlay and overlay modes for inter-host communication. Alternatively, Calico and Cilium only operate on L3, without utilizing a virtual switch for intra-host communication, while supporting underlay as well as overlay modes between different hosts.

3.2 Packet forwarding and filtering

In the L3 CNIs (Cilium and Calico), a widely used approach for packet forwarding has been to rely on kernel IP functions [11]. However, recently, Cilium and Calico have added support for extended Berkeley Packet Filter (eBPF) datapath, which allows attaching kernel-level microprograms to certain networking events (hooks) [4]. eBPF enables efficient and secure execution of user-defined code in the kernel space and is applied in different computational areas, including networking, security, observability, and tracing. A Just-In-Time compiler converts the eBPF microprogram to kernel byte code, thus allowing natively compiled kernel code efficiency during execution. For enhanced security, the program goes through a verification, which ensures that it runs to completion and does not try to access memory out of bounds [12]. As an additional hardening step, the verified byte code is made read-only at execution time, and any modification attempt will result in a kernel crash [4]. These eBPF programs can be attached to the veth-pairs to route packets inside the host or to another.

Most of the discussed plugins support the standard Kubernetes network policy [3], based on iptables, which operates on L3 and L4 [11]. While Flannel does not implement any network policy technologies, Calico and Cilium utilize alternative solutions, along with the standard L3-L4 approach. These enable more fine-grained network policies covering L3-L7. Calico achieves this by implementing its own iptables-based solution, whilst Cilium relies on eBPF microprograms for this purpose, as well. eBPF programs are more customizable than iptables. For instance, they can be written using high-level programming languages, and packets can be filtered based on the contents of the packet, rather than just the source or destination IP address [4].

4 Performance

Container network performance can be measured by several metrics, including the amount of data that can be transferred from one container to another in a unit of time (i.e., throughput), and the time it takes for a packet to travel from the source to the destination (i.e., latency). [10], [11], [6], and [13] evaluated the performance of widely used CNIs using various tools such as iperf, netperf, sockperf and sparkyfish. In addition, open-source Kubernetes benchmark tools are available for this purpose, such as knb [7].

In intra-host communication, eBPF approaches outperform the L2 bridge and kernel IP forwarding ones in both throughput and latency [10]. While bridging, IP forwarding kernel calls and processing iptables rules incur overhead, eBPF consumes less CPU power by the kernel microprograms handling both packet forwarding and filtering, thus achieving lower latency and higher throughput.

For inter-host communication, overlay mode performs poorly compared to native underlay networks [13]. In addition to the overhead of encapsulation and decapsulation, the lack of hardware acceleration, (i.e. tunnel offloading) can play a role in the underperformance [11]. Tunnel offloading is a mechanism that enables more efficient handling of overlay network traffic by delegating the task of encapsulation and decapsulation to the host Network Interface Controller (NIC), thus allowing the host CPU to perform other tasks. Not all NICs support every tunneling protocol of the CNI, therefore some CNIs might achieve degraded performance with a set of overlay tunneling protocols.

Numerous technical solutions can improve container network performance, including memory sharing, Remote Direct Memory Access (RDMA), and Data Plane Development Kit (DPDK). Sharing the host memory and enabling Inter-Process Communication between containers on the same host would eliminate the overhead incurred by network transmission [15]. RDMA allows direct memory-to-memory data transfer between hosts by enabling the NIC to directly communicate with the host memory controller, thus decreasing the computational burden of the CPU [9]. Note, these approaches degrade the isolation of containers, which might be undesirable for certain applications. DPDK enables containers to access the NIC directly without involving the kernel network stack, thereby achieving lower latency and higher throughput [5].

5 Use cases for different CNIs

This section compares the intended uses of the analyzed CNI implementations. It is partly based on the stated goals of the plugin developers and partly on the technical analysis of their implementation, and performance above.

While Flannel, Kube-router, and Weave only serve as standalone CNI plugins, Calico [14], and Cilium [8] provide consumer-grade as well as enterprise-scale Kubernetes networking solutions. The enterprise offerings include customer support, managed clusters, and monitoring features. Therefore, if these features are critical for the given containerized applications, Calico and Cilium are feasible options.

As mentioned in Section 3, CNIs differ in network policy rule specificity. Unlike the others, Flannel does not support any network policy solution. This results in fewer configuration options, and eliminating the overhead of rule processing can also improve performance. Thus, Flannel could be a suitable option for a quick, basic network without extensive security measures. Weave and Kube-router can be used if no packet filtering is needed above the transportation layer. However, for more control, Calico and Cilium could be suitable.

As pointed out in Section 3, most of the CNIs support both underlay and overlay networking for intra-host communication. [13] measured the performance of both modes based on packet size. When it comes to small-sized packets, the overlay approaches of Flannel, Weave, and Calico demonstrated comparable performance. However, in larger-sized data transfer, underlay Calico outperformed the overlay alternatives. As a result, for small packet traffic, e.g., message queue services, both modes seem to be viable. In contrast, for bulk data transfer, such as file sharing, overlay mode could be more efficient.

6 Discussion

CNI plugins fit very well in the Kubernetes ecosystem since they provide standard, pluggable interfaces to configure inter-pod network connectivity. There are technologies moving routing and filtering decisions closer to hardware, and CNI plugin development seems to be going towards eBPF microprograms due to their efficiency, security, and customizability. Cilium has completely adopted eBPF, and Calico has integrated it as one of

their datapath offerings. As opposed to changing the kernel source code or writing a kernel module, these programs can be developed at a much faster rate. Furthermore, since eBPF is not limited to networking applications, it spans a larger developer community. For basic, general use, the analyzed plugins (Calico, Cilium, Flannel, Kube-router, and Weave) prove to be sufficient. Most of them support both underlay and overlay networks with multiple tunneling options, therefore users can flexibly choose depending on the type of application. However, Calico and Cilium seem to be standing out with their fine-grained network policies and enterprise offerings.

7 Conclusion

This paper has reviewed the technical solutions of the most widely used CNI plugins and analyzed their design, performance, and possible use cases. The CNI specification defines rules about the network configuration format, the plugin-runtime communication protocol, plugin delegation and return data types. The base plugins configure essential single-host networking and serve as a good starting point for plugin developers. Generally, in intra-host communication, most of the plugins operate on L3 (kernel IP forwarding) or L2 (virtual switch). In addition, eBPF microprograms (implemented by Cilium and Calico) are more flexible and outperform the other solutions. In inter-host communication, by eliminating the overhead of encapsulation and decapsulation, underlay solutions provide more efficient communication. Regarding packet filtering, while Flannel does not implement network policies at all, Calico (using its own iptables implementation) and Cilium (relying on eBPF) allow rules on higher layers than L3 and L4, therefore achieving more fine-grained network policies.

References

[1] Bridge plugin README.md. https://www.cni.dev/plugins/current/main/bridge/. accessed 12-04-2023.

[2] CNI specification. https://github.com/containernetworking/cni/blob/main/SPEC.md. accessed 12-04-2023.

[3] Kubernetes Network Policy API. https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/. accessed 12-04-2023.

[4] What is eBPF? An Introduction and Deep Dive into the eBPF Technology. https://ebpf.io/what-is-ebpf/. accessed 12-04-2023.

[5] Ubaid Abbasi, El Houssine Bourhim, Mouhamad Dieye, and Halima Elbiaze. A Performance Comparison of Container Networking Alternatives. *IEEE Network*, 33(4), July 2019. Conference Name: IEEE Network.

[6] Alexis Ducastel. Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network. https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49. accessed 12-04-2023.

[7] InfraBuilder. k8s-bench-suite. https://github.com/InfraBuilder/k8s-bench-suite. accessed 12-04-2023.

[8] Isovalent. Isovalent Cilium Enterprise: Observability, Security, Networking. https://isovalent.com/product/. accessed 12-04-2023.

[9] Jacob Nelson and Roberto Palmieri. Understanding RDMA Behavior in NUMA Systems. In *2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, February 2019.

[10] Shixiong Qi, Sameer G Kulkarni, and K. K. Ramakrishnan. Understanding Container Network Interface Plugins: Design Considerations and Performance. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks*, Orlando, FL, USA, July 2020. IEEE.

[11] Shixiong Qi, Sameer G. Kulkarni, and K. K. Ramakrishnan. Assessing Container Network Interface Plugins: Functionality, Performance, and Scalability. *IEEE Transactions on Network and Service Management*, 18(1), March 2021.

[12] Dominik Scholz, Daniel Raumer, Paul Emmerich, Alexander Kurtz, Krzysztof Lesiak, and Georg Carle. Performance Implications of Packet Filtering with Linux eBPF. In *2018 30th International Teletraffic Congress (ITC 30)*, volume 01, September 2018.

[13] Kun Suo, Yong Zhao, Wei Chen, and Jia Rao. An Analysis and Empirical Study of Container Networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018.

[14] Tigera. Calico Cloud Documentation. https://docs.tigera.io/calico-cloud/. accessed 12-04-2023.

[15] Tianlong Yu, Shadi Abdollahian Noghabi, Shachar Raindel, Hongqiang Liu, Jitu Padhye, and Vyas Sekar. FreeFlow: High Performance Container Networking. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, New York, NY, USA, November 2016. Association for Computing Machinery.