Proceedings of the Seminar in Computer Science (CS-E4000), Fall 2022

Antti Ylä-Jääski and Wencan Mao

Tutors for seminar topics

Ali Unlu, Antti Yla-Jaaski, Hamza Khan, Sanna Suoranta, Trung Trinh, Lorenzo Corneo, Blerta Lindqvist, Miika Komu, Jaakko Harjuhahto, Vesa Hirvisalo, Chris Brzuska, Seied Hoseini, Mario Di Francesco, Anton Debner, Linh Truong, Alex Hämäläinen, Petri Mähönen, Vishnu Raj, Timo-Pekka Heikkinen, Amin Babadi, and Jaakko Lehtinen

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

Adapting Kubernetes Pod Scheduling for Fog Computing

Aaro Isomäki
aaro.isomaki@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

Fog computing and fog orchestration are growing areas of research as controlling a geo-distributed system includes several challenges such as increased network latencies, bandwidth variability and deployment management. This paper reviews recent modifications to Kubernetes, a cloud-based orchestration tool, to solve the issue of geo-distributed orchestration. The paper reviews four specific implementations and analyzes their capabilities by comparing them to each other.

KEYWORDS: Kubernetes, Fog, Orchestration

1 Introduction

In recent years the need for decentralized computing has increased as a result of applications such as self-driving cars and IoT devices which benefit from having low-latency compute capabilities close to the end user. With the increase of decentralized computing has also increased the demand for a standardized way of controlling it. In cloud data centers orchestration, the control and management of computers, is more straightforward due to predictable and controllable variables and conditions. With decentralized computing, or fog computing, there are different orchestra-

tion challenges related to network capabilities, heterogenous hardware and geo-distribution [6]. Geo-distribution, as a characteristic of fog computing means that the compute nodes are spread out geographically in different places. This allows a fog based application to potentially provide better Quality of Service (QoS) to users that are based in different places and possibly far away from a cloud data center.

Kubernetes has been the de-facto standard for orchestration in the cloud [4]. The purpose of this paper is to examine how Kubernetes can be modified to work in fog computing from the perspective of orchestration. More specifically, how the geo-distributed system is controlled and what advantages and disadvantages are introduced with distributed scheduling.

Geographic distribution raises challenges regarding how to host and execute application instances in compute nodes [9]. Resource availability can differ and it may be difficult to orchestrate which nodes to host services in. Compute clusters can become full and need to relay capacity information to other clusters. The QoS requirements can further complicate the orchestration requirements by placing rules on how well the geo-distributed system must behave by reacting and adapting to changing network conditions. Varying network conditions and delays can quickly change the environment to which the orchestrator must adapt seamlessly. This paper presents different technologies that solve these issues, analyses how each of them approach the geo-distributed orchestration problem and compares them to each other. We will examine parameters such as scheduling and deployment times, adaptivity and reactivity to network changes, architectural approach for scheduling, and latencies.

The contents of this paper are as follows: Section 2 describes the background of Kubernetes and fog computing. Section 3 describes the methods of geo-distribution in Kubernetes scheduling. Section 4 describes the advantages and disadvantages of each method described in Section 3. Finally, section 5 describes the conclusions and findings of this paper.

2 Background

In this section, we will look at an overview of some of the core concepts of the paper.

2.1 Fog

Fog computing is a paradigm used to describe computing extending from the cloud data centers right until the edge of the network. Unlike the cloud, fog consists of heterogenous hardware with differing network conditions and geo-locations. Fog-based processing can have superior latency capabilities and QoS due to being on the network edge [1]. An example use case for a fog edge device would be a real time application such as processing sensor input from self-driving cars and returning data results to the vehicles. In this example the system cannot suffer from delays caused by distances to data centers.

2.2 Kubernetes

Kubernetes [3] is an open-source software system for managing containerized applications. It is widely adopted and extensible. Kubernetes has a plethora of features to help manage a containerized system, including: service discovery, load balancing and storage orchestration. A central feature in Kubernetes is the pod scheduler. To understand what the scheduler does a few definitions are needed. A Pod is a deployable unit which is essentially one or more containers. A Node is a Virtual Machine (VM) or physical machine where the pods are run. The purpose of the scheduler is to assign pods to nodes based on built-in principles such as scoring and filtering. The default Kubernetes scheduler is centralized and easily extensible to add custom scheduling logic or even to replace with an entirely different scheduler. Kubernetes uses Kubelets, agents running on the nodes to listen to the Kubernetes master for tasks to run on the nodes.

3 Control of a geo-distributed system

Masip et al. [8] describe three methods of orchestration control for the fog: centralized, decentralized and distributed, as visualized in Figure. 1.

In centralized control, one main node controls the system. This is the default scheduler control mechanism in Kubernetes. It works well in a data center environment where the scheduler has a holistic view of the entire system and the compute conditions are predictable. In fog however, an issue with this approach is that the main node is a single point of failure for the entire control system. Also, a fully centralized control system can

have difficulties in determining optimal placements for pods in nodes and making decisions regarding the entire system when the system is highly distributed.

In decentralized control there is no single main node to control the entire system, but rather several local main nodes that control smaller parts of the larger system and that communicate with each other. In this method exchanging information between nodes can be more difficult but the single point of failure is addressed. Additionally, decisions about pod placement and ranking is done closer to the fog nodes.

In distributed control there is no main node concept, but rather all nodes make decisions for themselves and communicate with other nodes.

In the following sections we will look at custom implementations of Kubernetes and its scheduler for fog environments and understand how they control a geo-distributed system.

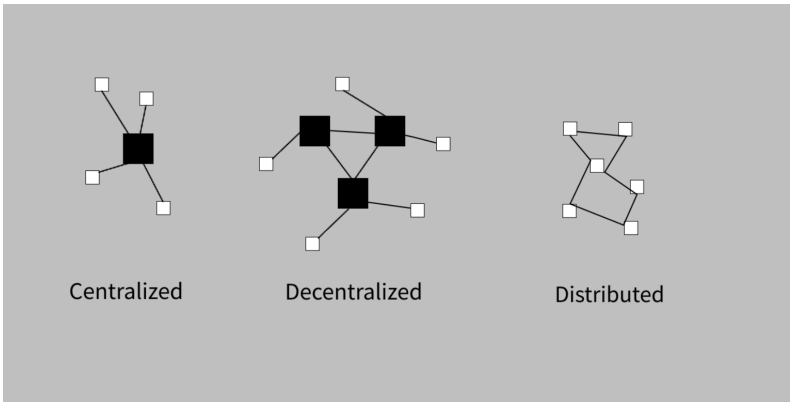


Figure 1. Three control methodologies where the black boxes represent main control nodes and the white boxes are edge nodes. The lines represent connections between nodes.

3.1 Agentified scheduler (MAS)

Casquer et al. [5] implemented a custom distributed scheduler for Kubernetes which uses the Multi Agent System (MAS) to distribute scheduling tasks among agents in the processing nodes. This prevents the pod management decisions from being made completely by a central entity. Here the MAS platform provides the deployment specification and tracks statuses and sends events. The Agentified scheduler works by modifying core steps in the scheduling process: filtering and ranking. Node filtering is the process of deciding whether a pod fits on a node or not. Node ranking is the process of giving a priority score for fitting the pods on the nodes. Both filtering and ranking in the Agentified scheduler are moved

to the MAS platform and away from the control plane. Node ranking result is based on negotiation between the worker nodes while filtering is based on data retrieved from the MAS database. The scheduler watches unscheduled pods, waits for the agents in the processing nodes to elect a winner node which will then be used to make the binding, i.e. select which node the pod is placed in.

To test the Agentified scheduler the researchers compared it to a centralized scheduler. The analysis was done based on the time it takes to save a scheduling result. The tests were conducted by 20 repetitions of differing pod count deployments on both scheduler types. The comparison revealed that especially for the single pod deployment the scheduling time is significantly lower for the Agentified scheduler. However this difference decreases with increasing pod count. Thus the researchers conclude that performance benefits can only be seen with low pod counts.

3.2 Ge-kube

Rossi et al. [9] proposed an orchestration tool called Ge-kube that extends Kubernetes providing functionality for geo-distributed deployments. Ge-kube implements a custom scheduler and deployment service to address the geo-distribution challenges. The custom scheduler is deployed as a pod. It can poll the status of other nodes and determine the placement of pods. The custom scheduler works in conjunction with the deployment service to determine placements for pods. In the case that new pods are needed the custom scheduler retrieves monitoring information from the system and sends a scheduling request to the deployment service. The deployment service then utilizes placement policies to determine the pod placements.

Ge-kube has self-adaptation and reactivity capabilities. The self-adaptation uses monitoring, placement and other principles to monitor and execute changes when needed. These changes are scaling actions that remove or add new pods. Ge-kube scheduler periodically evaluates the application deployment status and triggers placement policies if scaling is needed.

The researches evaluated Ge-kube by testing it in a geo-distributed environment. Utilizing the placement policies they had created and comparing them to the default placement policy of Kubernetes: kube-scheduler. The results show that the average number of used nodes was better in all of the custom placement policies ranging from just over three to over five nodes per pod. Kube-scheduler resulted in 2.85 as the average num-

ber of used nodes. Also the custom placement policies resulted in better adaptation percentages and pod CPU utilization in almost all cases.

3.3 KubeEdge

KubeEdge is an orchestration platform built on top of Kubernetes to solve the issue of deployment to cloud and edge [2]. KubeEdge uses a split design for managing nodes that are on the edge. First it uses a service called EdgeController which runs in the cloud on behalf of the edge nodes it represents and listens to the master node for actions that are targeted for the edge nodes. The EdgeController then sends the actions as metadata to the edge nodes via a messaging bus called KubeBus. A system called AppEngine then executes the actions on the node.

The researchers evaluated the KubeEdge platform with a setup consisting of two edge nodes and two nodes running on the Kubernetes Master. KubeBus was used to connect all of the nodes together. The analysis was done via measuring network latency between edge and cloud and deployment times. The researchers found no measurable impact to the network latency. Deployment times were tested by Kubernetes Master being deployed to the edge node resulting in a 3 second delay, which the researchers deem to be at an acceptable level without providing any reference number. Edge node to edge node and edge node to cloud latencies were also tested and resulted in negligible differences [10].

3.4 Geolocate

Vilaca et al. [7] propose a scheduler called Geolocate that is a generic scheduler that can extend for example the KubeEdge platform. It attempts to solve the geo-distribution issue with a custom scheduling algorithm. The Geolocate scheduler consists of multiple parts. The scheduler-core contains the algorithm for pod placement, which includes user-defined rules that are either mandatory or preferential. The scheduler-implementation integrates the scheduler-core with e.g. KubeEdge.

Geolocate was analysed with a setup of a data-producing service running at an edge node and a data-consuming node running in the cloud which fetches the data from the edge node. Scheduling overhead was compared between Geolocate and standard Kubernetes scheduler and the results were similar with a 300 microsecond delay with a test repeated 20 times. Location aware scheduling was tested with a setup of small

and large distances between data-producing and data-consuming applications. The experiment was repeated 5 times and it resulted in 62 percent gain for Geolocate in response time when compared to the regular scheduler. In conclusion the researchers found that Geolocate has similar scheduling planning times to the KubeEdge default scheduler but outperforms it in response time due to better geo-awareness. The researchers noted that adaptation to workload changes needs further research and was not tested.

4 Discussion

The Agentified scheduler solved the problem of centralized scheduling by utilizing the processing nodes and agents in them for electing a winner node. This scales well and addresses partially the single point of failure issue with a centralized controller. When filtering and scoring take place in the worker nodes, only the winner node result needs to be relayed to the control plane which improves performance. The scheduler which makes the pod binding still lies on the control plane and can act as a single point of failure, even if it doesn't have to make the ranking and filtering decisions. It is also noteworthy that while the binding time was reduced with the Agentified scheduler, the benefits were seen only with a single pod deployment which limits the size of the deployment in terms of performance.

Ge-kube solves the geo-distribution problem with the master-worker principle. It has an Elasticity Manager for determining whether new pods are needed and a Placement Manager which determines the place for the pod. With constant monitoring Ge-kube is well suited for agile adaptation to the network utilizing the self-adaptation and custom placement policies. Average node utilization, adaptation percentage and pod CPU usage was better with the custom placement policies when compared to default policy of Kubernetes. It is worth noting that the placement policies tested all had their own use cases and there was no general policy that works in all situations. Also, here it also appears the monitoring and placement components can be single points of failure.

KubeEdge boils the actual control logic into two parts. The EdgeController runs in the cloud and listens to the Kubernetes master for tasks to be executed for the edge nodes it represents. The EdgeController then relays the tasks via KubeBus to the Nodes.

This architecture is better suited for a geo-distributed deployment than

the native Kubelet logic in Kuberentes. If the edge nodes are far away from the Kubernetes master and behind poor bandwidth and network conditions the communication from the edge nodes to the Kubernetes master can be troublesome. This is what the EdgeController solves in KubeEdge by listening and executing actions on behalf of the edge nodes but on the cloud. The KubeEdge architecture is quite complex due to its cloud-edge split design. This could be a challenge when the system is wanted to be kept simple. Also the researchers evaluations and metrics provided were a bit lacking. Deployment times and latencies were said to be acceptable.

Geolocate solves the geo-distributed problem by implementing geo-distribution rules. A rule can be mandatory or preferential and determines where the pod is deployed geographically and what to do if it does not fit on a node. This is clearly a more configuration dependent solution that doesn't scale and adapt automatically but could be useful for geo-distributed deployment that has pre-set user-defined locations and rules. The 62 percent gain in response time that the researchers found Geolocate to have, could be significant reason to employ this system assuming the manual configuration requirement does not prevent this.

Table 1 presents the orchestrator technologies and their respective control methodologies.

5 Conclusion

The four implementations we reviewed all have some sort of master component for monitoring/deploying pods to the fog nodes. None of the implementations are thus completely distributed but rather distributed with centralized components. This could be argued to be a consequence of being a modification of Kubernetes which is inherently centralized. To be completely distributed, a system would have to work in a sense, peer to peer without any central mediator.

Based on this review The Agentified scheduler is the most distributed of the compared technologies. Geolocate is in turn the most centralized of the compared technologies, having no actual distribution of the management/monitoring logic. Geolocate however can be used as an extension of e.g. KubeEdge so it is not a clear distinction. KubeEdge and Ge-kube both could be considered partially centralized and partially distributed having elements of both.

Technology	Control Methodology	Control Methodology Breakdown
Agentified Scheduler	Distributed	Node filtering and ranking are decentralized to worker node agents
Ge-kube	Distributed/Centralized	Utilizes the master-worker architecture to have a centralized state and view of the system but the execution is decentralized
KubeEdge	Distributed/Centralized	Utilizes split design of having parts of control reside in worker node and parts in the cloud
Geolocate	Centralized	A centralized controller that solves the geo-distribution problem by user-defined geo-placement rules.

Table 1. Technologies and their control methodologies

References

[1] Firas Al-Doghman, Zenon Chaczko, Alina Rakhi Ajayan, and Ryszard Klempous. A review on fog computing technology. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 001525–001530, 2016.

[2] KubeEdge Project Authors. Documentation, 2022.

[3] The Kubernetes Authors. Overview, 2022.

[4] Carmen Carrión. Kubernetes scheduling: Taxonomy, ongoing issues and challenges. *ACM Comput. Surv.*, may 2022. Just Accepted.

[5] Oskar Casquero, Aintzane Armentia, Isabel Sarachaga, Federico Pérez, Darío Orive, and Marga Marcos. Distributed scheduling in kubernetes based on mas for fog-in-the-loop applications. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1213–1217, 2019.

[6] Breno Costa, Joao Bachiega, Leonardo Rebouças de Carvalho, and Aleteia P. F. Araujo. Orchestration in fog computing: A comprehensive survey.

ACM Comput. Surv., 55(2), jan 2022.

[7] Ricardo Vilaca Joao Vilaca, Joao Paulo. Geolocate, 2022.

[8] Xavi Masip, Eva Marín, Jordi Garcia, and Sergi Sànchez. *Collaborative Mechanism for Hybrid Fog-Cloud Scenarios*, pages 7–60. 2020.

[9] Fabiana Rossi, Valeria Cardellini, Francesco Lo Presti, and Matteo Nardelli. Geo-distributed efficient deployment of containers with kubernetes. *Computer Communications*, 159:161–174, 2020.

[10] Ying Xiong, Yulin Sun, Li Xing, and Ying Huang. Extend cloud to edge with kubeedge. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 373–377, 2018.

Rootless containers

Aleksi Hirvensalo
aleksi.hirvensalo@aalto.fi

Tutor: Mario Di Francesco

Abstract

Containers are used to isolate applications at the operating system level. However, that isolation can be escaped and currently the threat is amplified by how containers are being run. Most containers run as root by default, which means that if they can escape the isolation, they are a root user of the host machine. To tackle the issue, rootless containers have been created. Rootless means to run the container and its runtime as non-root.

This paper introduces rootless containers and gives a general overview of the topic. Furthermore, a practical example is given on how to attack poorly configured containers and how to prevent the attack by enabling rootless containers.

KEYWORDS: container, container security, docker, rootless containers, sandboxing

1 Introduction

Containers realize application virtualization at the operating system level [6]. In particular, containers provide a lightweight mechanism to ensure isolation, allowing multiple applications to run on the same host. They also enable a specific application to always run the same, regardless of

the infrastructure. On a higher level containers do offer the same kind of flexibility in deploying and running applications as virtual machines [19]. However, containers are different in a number of ways.

Virtual machines are an abstraction of the hardware whereas containers are an abstraction at the application layer [19]. In fact, containers share the kernel of the host which allows for multiple containers to run on the same operating system. By sharing the kernel, containers can use the same resources and functions as the host machine, for example, the filesystem and networking. However, the visibility and accessibility of these resources are controlled. This is how the aforementioned lightweight isolation is achieved.

Before defining containers more in depth, it must be noted that the focus of this paper is Linux containers. This simplification is made as most often containers run on a Linux environment [10]. Containers are essentially a technology that uses features of the Linux kernel, mainly control groups and namespaces. Containers are nothing more than processes running on a Linux operating system. Control groups limit the resources that a certain process is assigned to, for example, memory and CPU [10]. Namespaces limit the visibility of operating system resources for a given process [10]. These resources can for example be file system structure and process.

Containers have rapidly become popular in software companies due to their benefits in performance compared to other means of virtualization [12]. Containers allow for agile development and efficient operations [18] and have been largely adopted in DevOps practices and building microservices. One of the most popular tools for containers is Docker [1]. an open-source project that can be run on Linux, macOS and Windows. On Linux and macOS, Docker is mainly used through VMs. Moreover, on Windows the user can choose between Windows containers or Linux containers through Docker desktop [8]. Docker is an open platform that can be used throughout softwares lifecycle e.g. to develop, to deploy and to run the software [5]. Docker also enables users to separate the applications from the infrastructure, allowing quicker software delivery.

As new technologies are developed and introduced, new security concerns and vulnerabilities often arise. Containers are no exception [10]. Compared to virtual machines, containers provide a weaker level of isolation, as containers share the same kernel as the host machine. A stronger level of isolation is preferred because it mitigates the risk of malicious

applications interacting with the host or other containers. This paper focuses on rootless containers, a method to mitigate the aforementioned threat. The goal is to introduce rootless containers to the reader and give a technical overview of the topic. Also, a tutorial is given on how to use rootless containers with Docker.

The paper starts with the background, where containers are described and a motivation for rootless containers is given. Then it covers how rootless containers are made and how to enable them in practice. Finally, sandboxing is briefly covered followed by concluding remarks.

2 Background

Operating containers involves three important elements, the container runtime, the container image and the container itself. The container runtime takes care of running the containers, it makes sure that the container is isolated accordingly and necessary resources are created for the container to use, for example the filesystem and the container user are configured. The runtime is able to create the needed resources by instructions that the container image provides. In fact, the container image's only responsibility is to describe how to run the container i.e. the container environment. Regarding Docker, the container runtime is called Docker daemon which is responsible for building, running and distributing containers [5]. Docker registry takes care of storing the images. A Docker registry takes care of storing the images; Docker Hub is a public registry, used by default [4].

In general, most containers run as root, which corresponds to the host machine's root [10]. This is the case in Docker as well [10]. In Docker, one can configure the container to run as non-root but the runtime will by default run as root.

Running containers as root is clearly a security risk; if the container user manages to escape its isolation, it will be in control of the host machine. Therefore, it would be convenient to run containers as non-root if possible. That is indeed possible through the so-called rootless containers, a term which broadly refers to running the containers and also the runtime as non-root [11].

At a first glance, rootless containers seem like an unnecessary concept. Why do containers even need to run as root by default? There are a few reasons. One reason is that they introduce some level of complexity, es-

pecially with networking. Rootless containers cannot bind to privileged ports and have limited access to the filesystem.

Another reason is that the container image needs to install software using package managers. This is reasonable but only during the build phase of the image [10]. After the dependencies have been installed the container can be configured to run as non-root.

The last reason is that for some applications it makes sense to run as root outside of containers, for example apps that bind to privileged ports [10]. This has been then translated to containers as well where it does not make much sense as the application can bind to any port and then the port can be mapped to privileged ports. To conclude, the aforementioned reasons can be circumvented at the cost of increasing complexity and few restrictions with rootless containers. Besides complexity, performance can be an issue with rootless containers, it has been shown that creating a rootless container takes longer than a normal container [16].

3 Achieving rootless

How can one achieve rootless containers? It can be simply divided into two main categories, *creating* containers as non-root and *running* containers as non-root. As mentioned previously, the containers should be created and run as non-root on the host machine. In the rest of the paper, non-root user will specifically mean a non-root user on the host machine.

3.1 Creating rootless

Before going into creating rootless, Linux capabilities should be introduced, they are an essential feature when considering rootless containers at all as certain capabilities are required to create namespaces and control groups. Capabilities can be assigned to threads and they dictate what operations the thread can perform. Capabilities can also be assigned to files. Certain capabilities are also needed to create containers. For example, *CAP_SYS_ADMIN* capability allows to create mount namespaces [2]. Typically processes started by a non-root user do not have special capabilities [10], whereas processes started by a root user basically contains all of the capabilities [2]. To grant capabilities, one needs to have *CAP_SETFCAP* capability which is automatically granted to root. Essentially, capabilities offer a much more fine-grained access control compared

to just being root or non-root.

One will notice a problem when trying to create containers as non-root because non-roots do not have the required capabilities. However, this can be circumvented with the use of user namespaces. The user namespace allows for a given process to have a different view of user and group IDs. In practice this means that a pseudo-root can be created inside a specific container. In this context pseudo-root means that the root user has admin rights inside a specific container but on the host machine, wherein it is actually a non-root user [10]. Within the user namespace the pseudo-root user can then apply capabilities to the processes as a typical root user would. However, these capabilities do not apply outside of the specific user namespace [15]. This means that even though the user has a wide range of privileges inside the namespace, it does not have them if it manages to escape.

However, using the pseudo-root introduces some complexity and limitations. One cannot expect an image that runs successfully as a root in a normal container to do the same in a rootless container, even though the rootless container might perceive itself to be running as root [10]. For example, the *CAP_NET_BIND_SERVICE* capability allows processes to bind to low-numbered ports but inside the rootless container the capability just does not work [10]. That would require the container to exercise the capability against host machines resources that it cannot see or have access to. Finally, it is good to realise that most of the applications running in normal containers will run successfully inside rootless containers as well [10].

3.2 Running rootless

How to run containers as non-root? A straightforward solution is to use rootless image and/or to specify a non-root user to run a container. Regarding rootless container images, Bitnami maintains a broad collection of non-root images [3]. Note that one can specify a pseudo-root to run the container as well. The important aspect is that the user is not root on the host machine.

As previously mentioned, running the containers as non-root is not rootless; if the container runtime is operated by root there is still a potential for outbreak. The running rootless section is quite irrelevant if the runtime operates as non-root. As the runtime is responsible for running the containers, it cannot elevate the containers to have root privileges if it is

running as non-root itself. To summarise, if the runtime is non-root, one cannot run the container as root. Running the container as pseudo-root is still possible.

3.3 Rootless Docker

Docker required to run its daemon with root privileges. This has recently changed as Docker introduced rootless mode since version 19.03, and is considered stable since version 20.10 [13]. Docker still runs as rootful (as opposite to rootless) by default and just by running `docker` without `sudo` does not mean one is running rootless. Therefore, one needs to install the rootless mode and switch to it explicitly.

With rootless mode a new user namespace is created for the Docker daemon to operate in. As of Linux 3.8, user namespaces can be created by non-roots [15]. This effectively allows non-root users to work with Docker completely without the root users help.

Running containers as non-root in Docker can be achieved by using the USER directive to specify a non-root user to be used. The specified user is then used for RUN command which will start the container. One can also use the `-user` flag with `docker run` and then after the flag, specify a non-root user id or group id. If one does not specify the user, it will be the root. Rootless images can also be used through Docker Hub by searching, for example Bitnami images [4].

Rootless mode can be observed with Docker, for example by running `docker run -it alpine sh` and then in the terminal `whoami`. It will tell that the user is root. However, running `sleep 100` and then `ps -fC sleep` on the host machine will reveal the truth. From the output it can be observed that the process is not assigned to the actual root on the host machine. Whereas if the previous commands are run in rootful docker, it can be seen that the root inside the container is also the root on the host machine.

Rootless docker contains some limitations [13]. For example, AppArmor, checkpointing and overlay network cannot be used. Also, only certain types of storage drivers are supported. Cgroup are only supported with cgroup version 2 and systemd . However, one can use `ulimit` and `cpulimit` to limit resources in a more traditional way.

3.4 Networking

Much of the rootless containers complexity comes from networking requirements [7]. As mentioned previously, binding to a low-numbered port cannot be done with non-root users as it requires capabilities that apply at the host machine level. Incoming internet connections cannot directly react to network namespaces. vEth pairs cannot be created without privileges. Also, lower network performance occurs as the networking system needs additional components to function as in privileged environments. Aforementioned problems are part of the reason networking tools as RootlessKit and slirp4netns exist. Those help to setup rootless containers while providing the same functionality as in privileged environments. RootlessKit is an open-source project created to run Docker as pseudo-root, leveraging user namespaces [9]. slirp4netns provides networking for unprivileged network namespaces by creating a tap device that acts as a default route [9]. Slirp4nets is a component used in rootless containers to provide communication from inside a container to the outside [7]. It achieves this by using a tap device.

Network interfaces cannot be implemented with user namespaces alone because vEth pairs need to be created between a container and a host [7]. RootlessKit helps with this as it relays communication from the outside to an intermediate network namespace. Intermediate network namespace has a bridge that containers can connect to and create its vEth pair. RootlessKit is also used for example, to make /etc writable.

4 Sandboxing

Running containers as root is not a problem itself, unless the root user is able to escape the container and access the host machine. One approach to tackle such problem is to use sandboxing. Sandboxing means to isolate an application so that it has a limited access to resources [10]. The definition overlaps with containers quite a bit as they also limit the access and visibility of resources. However, limiting resources with sandboxing means to limit what the actual code can do on the host machine.

There exists many sandboxing mechanisms [17]. But in this paper, sandboxing is briefly covered through only one application, called gVisor. gVisor is an application kernel developed by Google [10]. It can be viewed as a layer between a running application and the host operating system.

Its task is to intercept system calls of the application and then to replace or modify them to be executed on the host machine.

gVisor does not allow the application itself to control the system calls [14]. This provides security as the system call API can occasionally be exploited due to bugs and race conditions. gVisor does not just directly relay the system call in the host machine but rather it might make modifications or limit the call. It is important to note that using system calls through gVisor will still result in a system call on the host machine.

gVisor does still come with limitations. Mainly reduced performance and incomplete features [10, 20]. It does not implement all of the Linux system calls which obviously is a drawback as some of the system calls are simply not available. If the sandboxed application makes frequent system calls, it might significantly impact the performance, gVisor has been measured to have approximately 50% overhead in system calls compared to bare-metal [17]. Finally, the gVisor is large and complex which increases the likelihood of having vulnerabilites in itself [10].

5 Container outbreak

As mentioned previously, the main motivation for rootless containers is to protect the host machine in the event of container outbreaks. Container outbreak is an event where the container user is able to access or see resources on the host machine. This might happen in multiple ways, often due to poor configuration [10].

This section introduces a real life example of breaking out of a container and taking malicious action. Before going into the example, `confidential.sh`-script has been added as root under the `/usr/bin`-folder. Running the script is only allowed by root and running it echoes "This is confidential. This file cannot be modified.". Groups or others do not have any permissions enabled for the file. For the rest of the section, a non-root user is used.

An example of poor configuration is to mount a host directory to be available inside a container [10]. This can achieved with `-v` flag. One can run `docker run -it -v /bin:/rootbin ubuntu bash` in host machine's root folder to map `/bin` on the host machine to rootbin-folder in the container. Running `ls` inside the container shows indeed a `rootbin`-folder being present. Inside the rootbin a `confidential.sh`-file is found and it can be read with `cat confidential.sh`. The container was started with non-root user and

it is still able to read the file. This happens because the container user is actually mapped to the root user on the host machine. This would also mean that the user is able to edit the file, which is indeed possible and it can be edited with, for example, `cat » confidential.sh echo "This file has been modified."`. Now executing the file outputs `This file has been modified` on the last line. To be clear, the aforementioned actions have been made inside the container.

On the host machine's terminal `confidential.sh` can be executed again and it can be observed that the file output has changed, on the last line it reads `This file has been modified`. This is only one example of what could be done with poor configuration. The container user could also modify other files in the bin file, for example, hide some scripts inside some common executables like `ls`-command. The possibilities are limitless and very severe damage can be done.

Now observe what happens when rootless context is enabled. Container is run with the same command as before and `rootbin`-folder is created and inside it is `confidential.sh`-file. However, executing the file outputs: `permission denied`, the same applies when trying to read or write to the file. Running `whoami` outputs `root` but this is not actually root on the host machine and therefore the container user has no permissions on the file. Still, mapping the bin-folder to the container reveals the existence of `confidential.sh`-file, it would be best not to map the folder altogether if not needed.

Note that gVisor does not help preventing the above exploit as it does not concern about namespaces or cgroups. gVisor is only allowed to block malicious system calls. It is advised to run rootless containers paired with gVisor for maximum protection.

Mounting critical files to a container's environment poses a very serious threat but an even bigger threat can be introduced with Docker's `privileged` flag. In Docker's default environment root user is used to run containers but it does not contain all capabilities [10]. The `privileged`-flag effectively gives a much more wider range of capabilities for the container user, the capabilities include CAP_SYS_ADMIN which can be used to modify namespaces and mount filesystems. This means that a malicious user is able to mount the `/bin`-folder by itself.

6 Conclusion

Containers often run as root by default, for example, this is the case considering Docker's default environment. Running containers as root is not an issue in itself but it does leave a possibility of malicious user being able to escape the container environment with root access on the host machine.

Rootless containers is a technology that mitigates the threat of container outbreaks by running containers as pseudo-root. In the event of an outbreak the container user is not able to leverage root privileges on the host machine. Rootless allows for unprivileged users to create, run and manage containers [11]. They do have added complexity compared to rootful containers but this is rarely visible to the end-users as transferring rootful containers to rootless usually only requires making sure that the runtime is run by non-root on the host machine.

Rootless containers are relatively new technology and their large-scale adoption is still underway. It is encouraged for everyone to use rootless containers instead of rootful as there really is no reason for one to predispose their infrastructure to container outbreaks.

References

[1] Donnie Berkholz. *Docker Index Shows Continued Massive Developer Adoption and Activity to Build and Share Apps with Docker - Docker*. URL: https://www.docker.com/blog/docker-index-shows-continued-massive-developer-adoption-and-activity-to-build-and-share-apps-with-docker/ (visited on 10/19/2022).

[2] *capabilities(7) - Linux manual page*. URL: https://man7.org/linux/man-pages/man7/capabilities.7.html#DESCRIPTION (visited on 10/20/2022).

[3] *Containers*. URL: https://bitnami.com/stacks/containers (visited on 10/20/2022).

[4] *Docker Hub Container Image Library | App Containerization*. URL: https://hub.docker.com/ (visited on 10/23/2022).

[5] *Docker overview*. Docker Documentation. Oct. 19, 2022. URL: https://docs.docker.com/get-started/overview/ (visited on 10/19/2022).

[6] Jorge Gomes et al. "Enabling rootless Linux Containers in multi-user environments: The udocker tool". In: *Computer Physics Com-*

munications 232 (2018), pp. 84–97. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2018.05.021. URL: https://www.sciencedirect.com/science/article/pii/S0010465518302042.

[7] Naoki Matsumoto and Akihiro Suda. "Accelerating TCP/IP Communications in Rootless Containers by Socket Switching". In: (July 2022).

[8] nishanil. *What is Docker?* URL: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-define (visited on 10/19/2022).

[9] Alex Rapatti. "Rootless Docker Containers in Continuous Integration". In: (Nov. 2021). URL: http://www.theseus.fi/handle/10024/498794.

[10] Liz Rice. *Container security : fundamental technology concepts that protect containerized applications*. 1st. Sebastopol, CA: O'Reilly Media, 2020. ISBN: 1-4920-5669-3.

[11] *Rootless Containers | Rootless Containers*. URL: https://rootlesscontaine.rs/ (visited on 10/02/2022).

[12] Bowen Ruan et al. "A Performance Study of Containers in Cloud Environment". In: *Advances in Services Computing*. Ed. by Guojun Wang, Yanbo Han, and Gregorio Martínez Pérez. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 343–356. ISBN: 978-3-319-49178-3. DOI: 10.1007/978-3-319-49178-3_27.

[13] *Run the Docker daemon as a non-root user (Rootless mode)*. Docker Documentation. Oct. 21, 2022. URL: https://docs.docker.com/engine/security/rootless/ (visited on 10/23/2022).

[14] *Security Model*. URL: https://gvisor.dev/docs/architecture_guide/security/ (visited on 10/22/2022).

[15] *user_namespaces(7) - Linux manual page*. URL: https://man7.org/linux/man-pages/man7/user_namespaces.7.html (visited on 10/23/2022).

[16] Guillaume Everarts de Velp, Etienne Rivière, and Ramin Sadre. "Understanding the performance of container execution environments". In: *Proceedings of the 2020 6th International Workshop on Container Technologies and Container Clouds*. WOC'20. New York, NY, USA: Association for Computing Machinery, Jan. 11, 2021, pp. 37–

42. ISBN: 978-1-4503-8209-0. DOI: 10.1145/3429885.3429967. URL: http://doi.org/10.1145/3429885.3429967.

[17] Xingyu Wang, Junzhao Du, and Hui Liu. “Performance and isolation analysis of RunC, gVisor and Kata Containers runtimes”. In: *Cluster Computing* 25.2 (Apr. 1, 2022), pp. 1497–1513. ISSN: 1573-7543. DOI: 10.1007/s10586-021-03517-8. URL: https://doi.org/10.1007/s10586-021-03517-8 (visited on 11/11/2022).

[18] *What are containers?* Google Cloud. URL: https://cloud.google.com/learn/what-are-containers (visited on 10/02/2022).

[19] *What is a Container? - Docker*. Nov. 2021. URL: https://www.docker.com/resources/what-container/ (visited on 10/02/2022).

[20] Ethan G. Young et al. “The True Cost of Containing: A gVisor Case Study”. In: *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*. Renton, WA: USENIX Association, July 2019. URL: https://www.usenix.org/conference/hotcloud19/presentation/young.

Passive IoT solutions for 6G

Anton Sulkava
anton.sulkava@aalto.fi

Tutor: Hamza Khan

Abstract

With the increasing demand for faster and more efficient mobile networks, research has been conducted on the sixth generation of mobile communications. 6G has been envisioned to solve the short comings that have been noticed with current mobile communications. These include data rates, energy efficiency, dead zones and so on. With 6G networks, researchers have noticed a challenge with highly varying wireless channels. To combat this, research has been conducted on modifying the propagation environment. A key technology for this is reconfigurable intelligent surfaces (RIS). These surfaces can be used to modify the passing signals. A large aspect of these surfaces is their energy consumption, as it is lower than traditional amplifiers for example. This paper reviews the related key technologies of 6G and RIS. Then it studies how these RISs can be utilized in 6G and how the deployment of RISs can be conducted. Finally, we discuss some of the possible challenges and look at the future of RIS and 6G.

KEYWORDS: Iot, 6G, Passive IoT, RIS, IRS

1 Introduction

In today's world, the demand for increasingly faster mobile networking has grown as people consume and share more data constantly. In the past few years, the use of wireless communication networks has been increasing worldwide. Even though the fifth generation (5G) of mobile communication is not yet even fully used, the world is already looking at what comes next. The next step in mobile networks is 6G. The 6G network is planned to be more oriented towards connecting things and devices. Applications related to things and devices, such as the Internet of Things (IoT) and the larger Internet of Everything (IoE) are highly related to 6G. [1]

IoT plays a key role in providing opportunities in 6G and beyond networks [2]. IoT is a system with devices which have some levels of sensors that can measure data. The system also has the ability to communicate data to other systems. With the evergrowing energy consumption and the current world's energy problems, researchers have come up with unique solutions, namely passive IoT solutions. The key feature of these devices is that, they do not require an internal power source, rather they harness energy from the wireless signals in there surroundings. These devices are an essential part in building a power efficient wireless system, as power consumption poses a huge issue with larger scale IoT. [3]

With the next generation of mobile networks, a key aspect to be studied is how the radio waves and signals travel through the different mediums. As there are big variations in these mediums, the results can be slightly altered depending on the location of the transmitter and the receiver. To combat this, a new technology called reconfigurable intelligent surfaces (RIS) is being studied. The key idea behind RIS is to manipulate the signal phase or amplitude in such a way that the received signal strength is maximized. These surfaces consist of several passive reflective elements and a controller, which enable altering the incident signal. The RIS's reflective elements reflect electromagnetic waves in a manner that can be controlled. Controlling the surfaces is achieved with programmable controllers inside the surface. As the surfaces are configurable and controllable, they can be attuned to specific environments, and they can be responsive in regard to changes in the environment. [4] [5]

The aim of this paper is to study the use of RIS in wireless communication networks, mainly in 6G. Namely, how the surfaces can be imple-

mented and how can they improve the existing communications technologies. The paper is organized in sections. Section 2 presents the two main technologies involved in the subject. Section 3 reviews the implementation of RIS into beyond 5G and 6G communication systems. Section 3 also discusses the challenges and the future related to RIS. Finally, Section 4 includes a conclusion.

2 Technologies

In this section, we review the key technologies related to 6G networks and reconfigurable intelligent surfaces. Sec. 2.1 emphasizes important points of 6G and the current issues that are being addressed. Sec. 2.2 elaborates on technical and general information about RIS. Sec. 3 studies the different use cases and advantages of using intelligent surfaces in 6G networks.

2.1 6G

The use of the fifth generation of mobile networking is increasing and a few short comings and limitations have been noticed. These limitations are not necessarily related to personal communication and device use, rather they have been noticed in upcoming use cases related to IoT, IoE, augmented reality (AR), extended reality (XR) and so on. As some 6G driving applications have been speculated to revolve around these concepts [6]. Some key limitations in the current 5G networks revolve around latency, capacity, energy consumption, reliability and coverage, these limitations will be addressed with upcoming 6G networks [1] [4] . Some solutions to this limitation are delayed behind upcoming complex network applications. These technologies, such as 6G, have a highly varying wireless channel, i.e., the propagation environment. [7]

To combat the problems with the propagation environment, another key technology has awoken interest in research. Earlier the propagation of signals in the environment only depended on the transmitter and receiver and no study has been conducted to alter the signal behavior while propagating, however current research has been done in possibilities to configure the signals in a way which better combats environmental effects. This could be achievable with intelligent surfaces. With the help of these surfaces the passing signals could be modified to suit the environment better

[4]. Thus, eliminating some of the problems that structural or natural objects might cause to signals. As mentioned before, the world's energy consumption is growing rapidly and the ever growing communication networks are contributing to this consumption as well. In this regard, the use of intelligent surfaces that are targeted to be passive, can be used to combat the excessive need for energy. These surfaces can also be used to replace existing power amplifiers and repeaters that consume larger amounts of energy. [8]

2.2 Reconfigurable intelligent surfaces

A variety of intelligent surfaces are coming up, as more efficient ways to amplify and modify signals are being researched. RIS, intelligent reflecting surfaces (IRS) and large intelligent surfaces (LIS) are the key candidates in combating problems in communications networks. RISs and IRSs are the same technology but with different callings, and LISs are the same technology but in a larger scale. [2][9] For this paper we will focus on the intelligent surface technology as a whole, and we will use the RIS abbreviation for clarity.

RISs have many potential use cases and environments. They can potentially be installed almost anywhere, the installations can be done outside, inside, on moving objects and even on people's wearables. A few use cases are improving cellular connections, indoor communications, and potentially large scale IoE networks. [10] However, this paper will focus on the subject of improving cellular networks.

The RISs work by having electromagnetic surfaces that have meta-atoms in them, meta-atoms electromagnetic units that compose the surface. These surfaces are also known as metasurfaces. The concept of metasurfaces is that each of these surface elements can be modified individually where by the elements reflection amplitude and phase shift can be changed. This real-time configurability is a requirement for wireless network communications. The reconfigurable surfaces are composed from three layers, the front layer contains the metasurface, the second layer is a back plate made of copper. This copper plate's function is to prevent signal leakage. The last layer is the control circuit board. This layer is linked to the RIS controller as well, the controller and circuit board work together to configure each of the metasurface's elements properties according to the changing needs.[2] [11]

One key aspect of the surfaces is the passivity of them. This requires

optimization of the passive beamforming techniques, which however has some challenges to it. Beamforming is elaborated more in section 3.2. These challenges relate to the tuning parameters of each element, which is out of the scope of this paper [11]. The surfaces are not fully passive, as they need some power source for the controller [2]. However, this energy can be harnessed from the environment with solar or other passive sources. In addition, if the RIS does not need be configured in real time, the surface can be fully passive [9]. Still the power need of these nearly passive surfaces are significantly lower than the traditional methods that use power amplifiers or repeaters.

3 Passive IoT with 6G

Current research in 6G has many different subjects, but in this paper, we will focus on the wireless propagation environment research done for beyond 5G and 6G communications. The propagation environment is currently an issue as it can be unpredictable. The RISs are envisioned to combat this problem. The RIS deployment can combat many issues as well. Such as, structural blockages, blind spots from signals and increased signal strength. [7]

This section will be divided into subsections. Sec. 3.1 will elaborate on the deployment of RIS and how they can be implemented to current network architecture. Sec. 3.2 will focus on the possible problem areas of implementing RISs into beyond 5G and 6G communications. Sec. 3.3 will focus on the future of RIS.

3.1 RIS deployment for network communications

The research in RIS deployment strategies is already ongoing, and the prospect of RISs being deployed for next generation network communications is very probable. There are two types of communication environments where intelligent surfaces can be beneficial, terrestrial and non-terrestrial.[7] Fig. 1 is a simplified image of deployment scenarios.

The deployment strategies for terrestrial environments include a single RIS and multi RIS deployment. A single RIS deployment is deployed with a single surface that hold all of the reflective elements, a multi RIS deployment is the opposite. Multiple RISs are used separately, these deployments have differences in coverage, costs and optimization. [7]

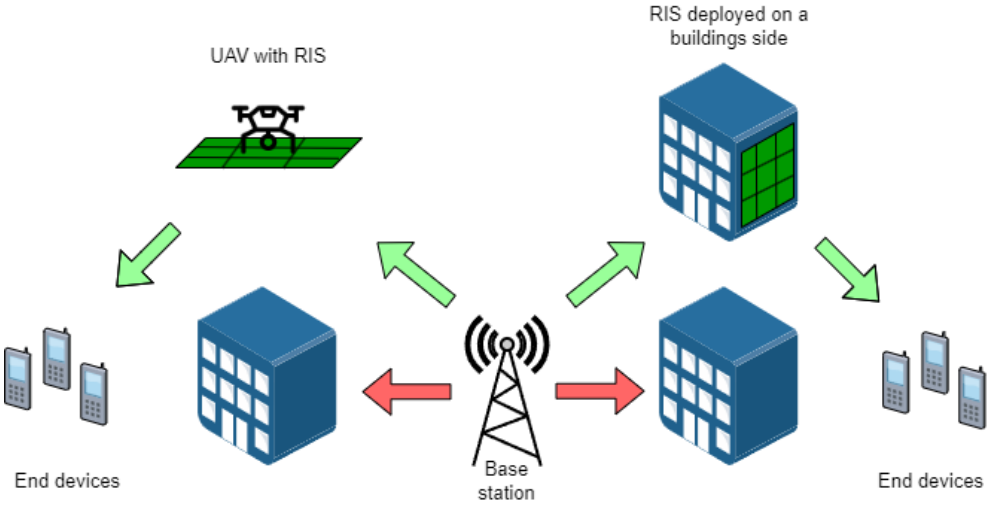


Figure 1. Simplified terrestrial and non-terrestrial RIS deployments, red arrows indicate blocked signals and green vice versa.

In addition, the locations of the RIS systems need to be considered. The RISs can be deployed user-side, base station (BS) side, or as a hybrid deployment. In a hybrid deployment the RISs are deployed in both user and base station sides. User-side deployment is designed to enhance the coverage in certain locations. In user-side deployment the surfaces are deployed near the users, so locations where coverage is worse or the coverage is nonexistent, or the network needs to be strengthened. Base station side deployment is located near the base stations, rather than near the users. This is used to strengthen and redirect the signals leaving the base station. In a hybrid deployment, both of the deployments are taken advantage of. [7][12]

Terrestrial deployment strategies are centered around unmanned aerial vehicles (UAV). UAVs can be used to create an air-borne wireless coverage, and they can be used as a sort of base station. However, for the scope of this paper we will look at using an actual RIS with the UAV. These kinds of movable aerial surfaces can be convenient for sudden network drops or coverage issues. [7]

3.2 RIS challenges

The road to deploying RISs for 6G communications still has many challenges to overcome. This section lists some of the challenges that might arise in the future of RIS assisted communication networks.

RIS deployment

As mentioned before there are two strategies for the locations of RIS deployment, user-side and base stations side. Both of these have their own challenges. User-side placement leads to a high number of RISs that all require controllers, that increase the hardware and energy requirements. However, when placing on BS-side, the line of sight for the channels are not that optimal. The locations of the RISs need to be studied to identify the optimal placement strategy .[13]

Channel state information

Channel state information (CSI) is critical knowledge for optimizing the RISs coefficients and beamforming gains. Beamforming is a technique that enables combining a "beam" of wireless signals, the signals are formed to a beam that is focused to specific receivers. This technique results in an enhancement of the signal directivity. With the passive nature of the RISs, the surfaces cannot receive or send pilot signals on their own, which means that channel estimation needs to be done with some assistance or with estimation calculations. A optimal method needs to be researched to combat this as CSI is critical for communications channels.[7][14]

Higher frequency bands

6G network requirements are expected to support higher data rate requirements, which leads to the large increase in frequency bands versus 5G. It is envisioned that the frequency bands will expand to THz bands and even above. [6][10] The increase in frequency leads to challenges with channels without a line of sight (LOS). As the frequency of the channel is higher, the signals are more prone to blockage by obstructions and obstacles. This can cause large challenges in environments that lack LOS. In addition the RISs need to be modified to suite the higher frequency bands, which will lead to challenges in the manufacturing and costs of the RISs. [7][10]

3.3 The future of 6G and RIS

The future of RIS in 6g networks is still very much in research. Despite the advances that have been made in recent years, there are several challenges and open issues that need to be researched. Sec 3.2 mentioned some of these but there is a lot more to research as well. Such as artificial intelligence (AI) empowered optimization, power consumption, hardware

complexity and so on. There are many possible research directions for different applications and use cases that are not inherent challenges, but possibilities. One aspect to look at is how other technologies can be used to empower RISs with 6G, and how other technologies can be empowered instead. Some of these technologies are included in the next sub sections.

Unmanned aerial vehicles

The use of UAVs has grown increasingly in many different industries, such as agriculture, military, logistics and so on. With increasing performances in 5G and 6G networks the UAVs can be utilized in different and more efficient manners. With RIS and 6G the LOS from the ground to the UAVs can be very precise, this can be achieved with flight plans and the beamforming gains from the RISs. [13] In addition to better connections between the ground and the UAV, introducing RISs also amplifies the energy efficiency of the usage of UAVs[5].

AI-empowerment

As mentioned, there are a lot of challenges involved in optimizing different aspects of RIS assisted networks, such as beamforming, channel estimation and phase-shift optimization. All of these optimization issues should be researched with possible AI and deep learning methods as they could prove beneficial. Channel estimation with AI based techniques has already been researched with different techniques. The authors in [15] adopted a neural network channel estimation with positive performances, using the normalized mean square error as a performance metric. Adopting a deep learning method for the phase shift reconfigurations in the RISs has also shown positive signs [16]. This shows a promising road ahead for AI-empowered optimization.

4 Conclusion

In this paper, we reviewed and studied the use of RIS in network communications. Firstly, we studied the key technologies, 6G and the RIS itself. We presented an overview of the upcoming 6G technology. This overview pointed out several short comings in the current networking systems. As a solution to some of these short comings, the RIS technology was presented as an option. RISs can replace existing technology, such as power amplifiers or repeaters, with additional benefits areas such as performance and

energy consumption. After reviewing the key technologies, we presented different deployment strategies for introducing RISs to network communications. These consisted of different options for RIS locations, amounts and so on. Finally, this paper discusses some of the potential challenges for RISs, and presents some possible research directions for the future of RISs and 6G networks.

References

[1] Chamitha De Alwis, Anshuman Kalla, Quoc-Viet Pham, Pardeep Kumar, Kapal Dev, Won-Joo Hwang, and Madhusanka Liyanage. Survey on 6g frontiers: Trends, applications, requirements, technologies and future research. *IEEE Open Journal of the Communications Society*, 2:836–886, 2021.

[2] Sarah Basharat, Syed Ali Hassan, Aamir Mahmood, Zhiguo Ding, and Mikael Gidlund. Reconfigurable intelligent surface-assisted backscatter communication: A new frontier for enabling 6g iot networks. *IEEE Wireless Communications*, pages 1–8, 2022.

[3] Martin Beale, Hiromasa Uchiyama, and John Chris Clifton. Iot evolution: What's next? *IEEE Wireless Communications*, 28(5):5–7, 2021.

[4] Ying-Chang Liang, Jie Chen, Ruizhe Long, Zhen-Qing He, Xianqi Lin, Chenlu Huang, Shilin Liu, Xuemin Sherman Shen, and Marco Di Renzo. Reconfigurable intelligent surfaces for smart wireless environments: channel estimation, system design and applications in 6g networks. *Science China Information Sciences*, 2021.

[5] Teena Sharma, Abdellah Chehri, and Paul Fortier. Reconfigurable intelligent surfaces for 5g and beyond wireless communications: A comprehensive survey. *Energies*, 14(24), 2021.

[6] Walid Saad, Mehdi Bennis, and Mingzhe Chen. A vision of 6g wireless systems: Applications, trends, technologies, and open research problems. *IEEE Network*, 34(3):134–142, 2020.

[7] Faisal Naeem, Georges Kaddoum, Saud Khan, and Komal S. Khan. Intelligent reflective surface deployment in 6g: A comprehensive survey. 2022.

[8] Chongwen Huang, Alessio Zappone, George C. Alexandropoulos, Mérouane Debbah, and Chau Yuen. Reconfigurable intelligent surfaces for energy efficiency in wireless communication. *IEEE Transactions on Wireless Communications*, 18(8):4157–4170, 2019.

[9] Wenjing Yan, Xiaojun Yuan, Zhen-Qing He, and Xiaoyan Kuai. Passive beamforming and information transfer design for reconfigurable intelligent surfaces aided multiuser mimo systems. *IEEE Journal on Selected Areas in Communications*, 38(8):1793–1808, 2020.

[10] Rui Chen, Mengjie Liu, Yilong Hui, Nan Cheng, and Jiandong Li. Reconfigurable intelligent surfaces for 6g iot wireless positioning: A contemporary survey. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[11] Qingqing Wu and Rui Zhang. Towards smart and reconfigurable environment: Intelligent reflecting surface aided wireless network. *IEEE Communications Magazine*, 58(1):106–112, 2020.

[12] Changsheng You, Beixiong Zheng, Weidong Mei, and Rui Zhang. How to deploy intelligent reflecting surfaces in wireless network: Bs-side, user-side, or both sides? *Journal of Communications and Information Networks*, 2020.

[13] Shapagataiym Balgabay, Sultangali Arzykulov, and Galymzhan Nauryzbayev. Semi-Passive Reconfigurable Intelligent Surfaces Enabled Communications: A Comprehensive Survey. *Nazarbayev University*, 5 2022.

[14] Xianfu Lei, Mingjiang Wu, Fuhui Zhou, Xiaohu Tang, Rose Qingyang Hu, and Pingzhi Fan. Reconfigurable intelligent surface-based symbiotic radio for 6g: Design, challenges, and opportunities. *IEEE Wireless Communications*, 28(5):210–216, 2021.

[15] Shicong Liu, Zhen Gao, Jun Zhang, Marco Di Renzo, and Mohamed-Slim Alouini. Deep denoising neural network assisted compressive channel estimation for mmwave intelligent reflecting surfaces. *IEEE Transactions on Vehicular Technology*, 69(8):9223–9228, 2020.

[16] Sarah Basharat, Maryam Khan, Muhammad Iqbal, Umair Sajid Hashmi, Syed Ali Raza Zaidi, and Ian Robertson. Exploring reconfigurable intelligent surfaces for 6g: State-of-the-art and the road ahead. *IET Communications*, 16(13):1458–1474, 2022.

Container sandboxing

Antti Nousiainen
antti.h.nousiainen@aalto.fi

Tutor: Mario Di Francesco

Abstract

This paper considers the usage of technologies such as Linux Security Modules (SELinux, AppArmor) and system call filtering (Seccomp) in improving container isolation of applications via Docker and Kubernetes. This paper also considers the support of other container runtimes for these technologies and Kubernetes.

KEYWORDS: containers, sandboxing, SELinux, AppArmor, Seccomp, Docker, Kubernetes, syscalls, LSM

1 Introduction

With software that is publicly accessible, security is very important to ensure that the runtime environment in which the program is executed does not get compromised by an users malicious or unintentional actions for example.

One way to improve security is containerization. Containers provide an easy way to isolate projects and their dependencies from the host machine on which they run. However, the same security risks still exist for example in the form of attackers that target access into private data [16, Chapter 1]. Container sandboxing is one way to mitigate some of these possible security concerns by improving the security of the container runtime overall.

This paper will focus on three different ways of increasing container isolation: SELinux, Seccomp and AppArmor. It will also focus on how these can be implemented in containerized environments. The main focus will be on Docker and Kubernetes.

In addition to this the paper will also consider the container runtime support of Kubernetes and how different, common, container runtimes support access limitation with Linux Security Modules or system call filtering.

The goal of this paper is to provide an overview of how well the previously mentioned technologies are supported in containers (especially Docker) as well as how an user can implement them in their project. This is done in the form of summarizing the technologies and providing examples.

2 System calls and file permissions

Two possible ways to limit access to resources center around file permissions (Linux Security Modules: SELinux, AppArmor) and system calls (Seccomp) [16, Chapter 8].

In Unix, traditional file permissions are implemented in a way where a file has permissions for three different permission classes: for the owner, for the group and for all other users. It is possible to give these classes three kinds of permissions: read, write and execute [18]. This means that the possible permissions that can be given are somewhat limited in how they can be configured. However SELinux or AppArmor can be used to

add an additional layer of file permissions on top of this permission model [16, Chapter 8].

System calls on the other hand are the interface between applications and the Linux kernel. They allow the application to use kernel functions and are usually hidden from the developer by library wrapper functions [2]. Seccomp acts on these calls and filters them based on the currently active Seccomp profile. Through this, Seccomp can block unwanted actions from happening [16, Chapter 8].

3 Sandboxing containers

Container sandboxing is important due to the way in which containers are isolated.

Containers use two features of Linux to isolate their contents from the host: cgroups and namespaces. Namespaces can be used to create multiple isolated instances of available resources and cgroups (control groups) can be used to limit resources [15, p.3]. Currently Docker for example uses multiple namespaces to achieve process, filesystem, ipc and network isolation and cgroups to achieve device isolation [15, p.4].

Containers often run as root by default and make system calls to the same kernel as the host. This means that if the container isolation can be broken, be it deliberately or by accident, code running within the container can gain unintended permissions [16, Chapter 9]. This would then lead to the host system being compromised.

3.1 Seccomp

Seccomp (secure computing mode) functions by filtering the systemcalls the application can make [17, p.9]. When it was initially intruduced, it was extremely restrictive allowing only very few system calls to be made, and some of them only under special conditions. For example, file access requires file descriptors to be opened before switching to secure mode [16, Chapter 8].

Some years later a new implementation Seccomp-BPF was created to allow system calls based on a Seccomp profile specific to the application. This works so that when an system call is made, the filter can check based on the profile whether the call is allowed or not, with different kinds of possible actions depending on the result [16, Chapter 8].

For example, the default Docker Seccomp profile blocks tens of system calls [17, p.10] with next to no ill effect on an average application. However this profile is not enabled by default on Kubernetes [16, Chapter 8].

3.1.1 Implementing Seccomp for an application

Seccomp profiles consist of an default action as well as defined exceptions with different actions. These can either log (SCMP_ACT_LOG), prevent (SCMP_ACT_ERRNO) or allow (SCMP_ACT_ALLOW) system calls. It also supports limiting architectures. [1, 12]

A Seccomp profile has the following structure and might for example contain limitations like this:

```
{
  "defaultAction": "SCMP_ACT_ALLOW",
  "architectures":
  [
    "SCMP_ARCH_X86_64",
    "SCMP_ARCH_X86",
    "SCMP_ARCH_X32"
  ],
  "syscalls":
  [
    {
      "names":
      [
        "add_key",
      ],
      "action": "SCMP_ACT_ERRNO"
    }
  ]
}
```

This example allows all other system calls but prevents the container / application from using the kernel keyring. This limitation is done through the names section (add_key) which in this case is set to use the action SCMP_ACT_ERRNO [12, 11].

Using Seccomp with Docker requires very little setup as there is already an default profile available. However, it is possible to override it with the

```
--security-opt seccomp=/path/to/seccomp/profile.json
```

parameter [12].

Kubernetes supports Seccomp since version 1.19 and profiles can be automatically applied through configuration files. It is also possible to use the default Seccomp profile by running the kubelet with the

```
--seccomp-default
```

parameter [11].

3.2 SELinux

SELinux (Security-Enhanced Linux) is a Linux Security Module (LSM) that allows limiting the rights of a process in relation to its interaction with resources like other processes or files [16, Chapter 8].

The file permissions of SELinux also differ from the classic Linux file permissions by being based on file specific labels instead of user or group specific permissions [15, p.5]. When using SELinux, an application would have to pass both permission checks to function correctly. SELinux can also be set to only track and log violations instead of blocking them [16, Chapter 8].

SELinux can be difficult to use due to requiring an application specific profile with comprehensive knowledge on what resources the application in question needs for it to work properly and effectively. However some vendors provide pre-made profiles, which somewhat mitigates this [16, Chapter 8].

3.2.1 Implementing SELinux for an application

In Kubernetes it is possible to assign SELinux labels to containers through the seLinuxOptions field of the securityContext. Currently, by default, this causes the container runtime to apply the given label for all files within the containers volumes [3].

3.3 AppArmor

AppArmor is also a Linux Security Module. It works in a similar way as SELinux. It allows using profiles for limiting application access to resources [20, p.130], with the difference to SELinux being that it uses profile specific constraints instead of file specific labels [16, Chapter 8].

AppArmor also has a default Docker profile, but it is also not in use by default in Kubernetes [16, Chapter 8].

3.3.1 Implementing AppArmor for an application

AppArmor profiles are written in a profile language that has an extensive syntax for writing access rules. These are not only limited to file access

but also support restricting capabilities and networking to some extent [13]. An example profile might look like this:

```
profile example-profile flags=(attach_disconnected) {
  deny /** w,
}
```

This profile denies all file writes: `deny` is the rule modifier, `/**` matches the files and `w` sets the permission the rule relates to [13, 14].

In Docker, a default profile is enabled by default. This provides an acceptable security baseline for applications out of the box, but it is also possible to load a different profile with Docker [6].

To do this, the profile has to be loaded into AppArmor on the target system, with:

```
 apparmor_parser -r -W /path/to/your_profile
```

After that, the profile can be loaded with the `-security-opt` parameter as follows:

```
--security-opt apparmor=your_profile
```

[6]

In Kubernetes AppArmor is not enabled by default. Support for AppArmor is present in Kubernetes starting from version 1.4 [20, p.130] and its options are set via annotations, older versions ignore AppArmor annotations silently. Currently the support is still in beta state, which is the reason for annotations being used to set the required options. In the future, they will be replaced with first class fields once stable [10].

Prerequisites for using AppArmor include enabling the AppArmor kernel module on the node and ensuring that the container runtime that is in use supports AppArmor [20, p.130].

Currently Kubernetes has no native mechanisms for loading AppArmor profiles on nodes, therefore other methods such as SSH have to be used. When the profile has been copied to the node, it needs to be loaded to AppArmor in a similar way as when using Docker [10].

3.3.2 Automatic AppArmor profile generation

Generating new AppArmor profiles for containers by hand can be very time consuming and result in profiles that are too permissive. Tools such as Kub-Sec, Lic-Sec and Sysdig's K8s Policy advisor exist and can help with rule generation [20, p.1].

Lic-Sec, which is based on Docker-sec and LiCShield [19, p.3] works by

automatically generating AppArmor profiles for local Docker containers based on the container's behavior while the K8s Policy advisor generates rules based on the Kubernetes pods configuration. [20, p.1].

However, Lic-Sec or the K8s Policy advisor do not help in generating AppArmor rules based on the behavior of a container deployed with Kubernetes. For this a solution named Kub-Sec has been proposed which uses AppArmor audit logging and an separate API server running an automatic policy generator as well as other needed services to create and load rules on a pod [20].

4 Kubernetes container runtime support

Kubernetes supports container runtimes that conform with its CRI (Container Runtime Interface) [4] which is an API based on the gRPC protocol [5].

As of Kubernetes 1.25 the default CRI version used is v1, with support for the depricated version v1alpha2 having still been retained as a backup [4].

The supported list of container runtimes for example includes runtimes such as containerd, CRI-O, Docker Engine and Mirantis Container Runtime [4].

4.1 Container runtimes and access limitation

Currently the support for AppArmor on Kubernetes is not limited to only Docker. In addition, the other common supported container runtimes for Kubernetes such as CRI-O and containerd also support AppArmor [10]. These container runtimes also support Seccomp, but the default profiles of each container runtime differs [11].

For instance, containerd and docker seccomp profiles are by default quite similar, but might temporarily get out of sync due to the timing of updates in the future, while the cri-o seccomp profile is loaded (with some modifications) directly from the Seccomp library files [9, 7, 8]. Therefore it can be said that in some cases issues can arise from switching to another container runtime due to these differences if Seccomp is being used.

5 Conclusions

Overall, it can be said that Kubernetes supports multiple common container runtimes such as containerd, which in turn support access limitation through AppArmor, Seccomp and SELinux.

It can also be concluded that using additional layers of security such as AppArmor, Seccomp or SELinux with containers improves the overall isolation of the container thus increasing its security from the host's point of view.

However, enabling such solutions does come with the cost of needing additional configuration to be done, with the amount needed depending on which solution or solutions are chosen to be enabled, and on which platform (local or cloud environments etc.) the containerized application runs.

The possibility to use automatic profile generation tools is something that should also be considered when generating new profiles. It could improve the quality of profiles while reducing the time needed for their generation compared to doing it manually. Setting up such automatic profile generation could also help in the future as the base configuration needed for setting the tools up would not need to be done again in order to leverage the tools with new projects.

Many platforms also provide decent default profiles that can improve security with minimal need for additional reconfiguration. Enabling the default AppArmor or Seccomp profile for example could be enough to stop an security breach from happening, and considering the minimal time needed to enable these default profiles would therefore be a good place to start from if additional container sandboxing is desired in an environment.

References

[1] Seccomp rule add linux manual page. https://man7.org/linux/man-pages/man3/seccomp_rule_add.3.html.

[2] Syscalls linux manual page. https://man7.org/linux/man-pages/man2/syscalls.2.html.

[3] https://kubernetes.io/docs/tasks/configure-pod-container/security-context/, Sep 2022.

[4] https://kubernetes.io/docs/setup/production-environment/container-runtimes/, Nov 2022.

[5] https://kubernetes.io/docs/concepts/architecture/cri/, Nov 2022.

[6] Apparmor security profiles for docker. https://docs.docker.com/engine/security/apparmor/, Oct 2022.

[7] Default seccomp profile for containerd. https://github.com/containerd/containerd/blob/main/contrib/seccomp/seccomp_default.go, Oct 2022.

[8] Default seccomp profile for crio. https://github.com/cri-o/cri-o/blob/main/internal/config/seccomp/seccomp.go, Nov 2022.

[9] Default seccomp profile for docker. https://github.com/docker/docker/blob/master/profiles/seccomp/default.json, Aug 2022.

[10] Restrict a container's access to resources with apparmor. https://kubernetes.io/docs/tutorials/security/apparmor/, Oct 2022.

[11] Restrict a container's syscalls with seccomp. https://kubernetes.io/docs/tutorials/security/seccomp/, Aug 2022.

[12] Seccomp security profiles for docker. https://docs.docker.com/engine/security/seccomp/, Oct 2022.

[13] Steve Beattie and John Johansen. Apparmor core policy reference. https://gitlab.com/apparmor/apparmor/-/wikis/AppArmor_Core_Policy_Reference, Mar 2021.

[14] Christian Boltz and Steve Beattie. A quick guide to apparmor profile language. https://gitlab.com/apparmor/apparmor/-/wikis/QuickProfileLanguage, Aug 2021.

[15] Thanh Bui. Analysis of docker security. 2015.

[16] Liz Rice. *Container Security*. O'Reilly Media, Inc, 2020.

[17] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7:52976–52996, 2019.

[18] Jack Wallen. Understanding linux file permissions. https://www.linuxfoundation.org/blog/blog/classic-sysadmin-understanding-linux-file-permissions, Sep 2022.

[19] Hui Zhu and Christian Gehrmann. Lic-sec: an enhanced apparmor docker security profile generator. https://arxiv.org/abs/2009.11572, 2020.

[20] Hui Zhu and Christian Gehrmann. Kub-sec, an automatic kubernetes cluster apparmor profile generation engine. In *2022 14th International Conference on COMmunication Systems and NETworkS (COMSNETS)*, pages 129–137, 2022.

Survey of Modern Generative Modelling

Christian Montecchiani
christian.montecchiani@aalto.fi

Tutor: Vishnu Raj

KEYWORDS: Generative Models, Deep Learning, Generative Adversarial Networks, Variational Autoencoders, Energy-Based Models, Autoregressive Models, Normalizing Flows, Diffusion Models

1 Introduction

Generative Models are a class of **unsupervised** learning techniques of Machine Learning. The objective of this class is, given training data, to generate new samples from the same distribution.
However, it is not always possible to learn the exact data distribution, especially in the case of *high dimensionality*, such as images. For this reason, the most recent approaches leverage the power of Deep Neural Networks to learn a distribution $p_{model}(\cdot)$ similar to $p_{data}(\cdot)$.

There are many applications in which a generative model can be used:

- **Generate synthetic data** to increase the size of the dataset. An example of this application is StyleGAN a generative model proposed by Nvidia [9].
- **Image inpainting** that consists in reconstructing the missing regions in an image. In particular, the reconstruction must be realistic and context-based. An example of it is DeepFill [17].
- **Denoising** that consists in removing noise from a picture without losing the important features. Important generative models for im-

age denoising is Autoencoders [15].

This survey will summarize the most known and used techniques in this field. In Section 2, Generative Adversarial Networks will be presented. Section 3 will go through the Variational Autoencoders. Autoregressive Models and Normalizing Flows will be presented in Section 4 and 5, respectively. Section 6 will introduce the Diffusion Models.

2 Generative Adversarial Networks (GAN)

Generative Adversarial Nets (GAN) is a framework in which two models are trained simultaneously:

Generator $\boldsymbol{G}$ is a Neural Network that has the goal to generate new plausible sample from the real data distribution p_{data}.

Discriminator $\boldsymbol{D}$ is a Neural Network that estimates the probability that a sample comes from the training data (p_{data}) rather than being generated by $\boldsymbol{G}$.

The training procedure for $\boldsymbol{G}$ is to maximise the probability of $\boldsymbol{D}$ to commit an error, while the training for $\boldsymbol{D}$ is to minimize it. These opposite goals define the term *adversarial nets*. Both networks try to beat each other and, doing so they are improving in their objective.

2.1 Mathematical Details

The mathematical formulation of GAN training, used in the original paper [7], is discussed below. The generative network G is fed with a random input $\boldsymbol{z}$ sampled from $p_{\boldsymbol{z}}$ and $G(\boldsymbol{z})$ is the output that should follow the *target* distribution. On the other side, the discriminative network D is fed with a sample that can be drawn from the real distribution or from the generated one. The output is the probability of that sample to be a sample of the true distribution. Fig. 1 summarized the aforementioned procedure, which, from game theory, can be formulated as a *minimax* two-player game of the following value function $V(\boldsymbol{G}, \boldsymbol{D})$:

$$\min_{G} \max_{D} V(\boldsymbol{G}, \boldsymbol{D}) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})} \left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \left[\log(1 - D(G(\boldsymbol{z})))\right]$$

The **Nash equilibrium** of the mentioned minimax game is obtained when:

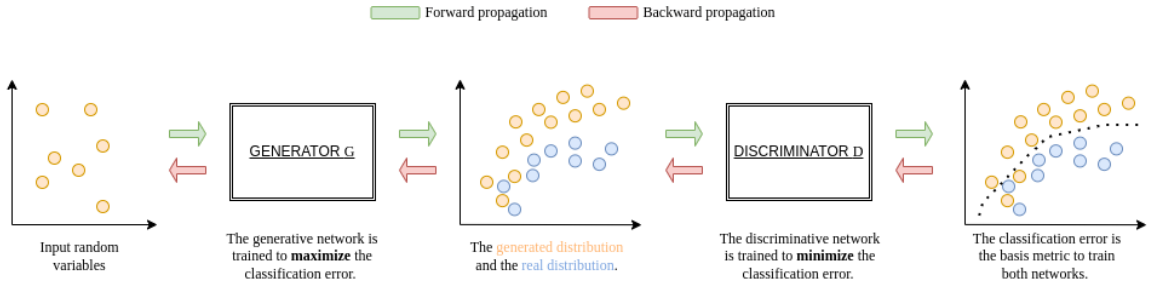


Figure 1. Summary of the training procedure. This figure was taken from [13].

- The data produced by the generator looks very similar to the data of the training set.
- The discriminator classifies real and fake images completely randomly: each class has 0.5 probability to be selected.

3 Variational Autoencoders (VAE)

The main component of Variational Autoencoders is an Autoencoder, which is composed of two Neural Networks:

Encoder E It takes a high dimensional vector, $x \in \mathbb{R}^n$, and compress it in a smaller representation, $E(x) \in \mathbb{R}^m$ where $m < n$.

Decoder D It is the *reverse* process. So, it takes as input the encoded representation and tries to reconstruct the original data, $D(E(x)) \in \mathbb{R}^n$.

The goal is to find the pair of *encoder-decoder* that keeps the **minimum** reconstruction error when decoding and the **maximum** information when compressing.

The new aspect that the Variational Autoencoders introduce with respect to the aforementioned Autoencoders is that: instead of encoding an input as a single vector, encode it as a *distribution* over the latent space. Fig. 2 shows a summary of the architecture of VAEs.
The **encoded distributions** are used for the generative purpose and are chosen to be Gaussian so that the encoder can be trained to return the mean and the covariance matrix. Then the loss function of the neural networks is composed of two terms:

Reconstruction Error which is used to make the encoding-decoding scheme as performant as possible.

Regularisation Term which generally is the Kullback-Leibler Divergence (KL) [6], which is a measure of distance between distributions. In VAEs it is used to make the distributions returned by the encoder close to a Standard Normal $\mathcal{N}(\bar{0}, \mathbf{I})$.

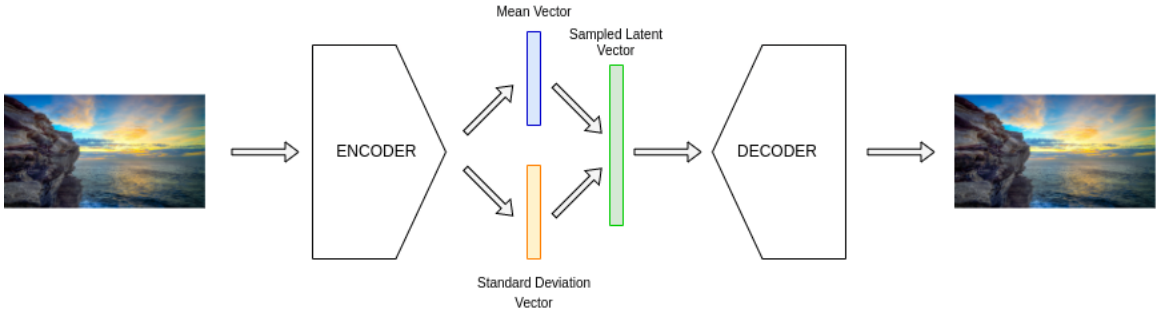


Figure 2. Architecture of VAEs.

The following equation represents the Loss used to train Variational Autoencoders:

$$\overbrace{||x - D(E(x))||^2}^{Reconstruction\ Error} + \underbrace{KL[\mathcal{N}(\mu_x, \sigma_x), \mathcal{N}(0, 1)]}_{Regularisation\ Term}$$

This loss is called *Evidence Lower Bound* (ELBO) [16], is a core component of the *variational infererence*, which is a paradigm to estimate the posterior distribution when the direct computation is intractable.

4 AutoRegressive Models

As mentioned in the previous sections to generate data we find the joint distribution. Another way to do that is to use autoregressive models. They do that by using the observations of the previous time steps to predict the value at the current time step, this is called **autoregressive property** [2]. Suppose we start with a data set $\mathcal{D}$ composed by a n-dimensional data points and by the *chain rule* of probability:

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i | x_{<i})$$

Where $x_{<i}$ is a vector of random variables with index less than i. This equation can be expressed by the Bayesian Network (BN) [4] in Fig.3. In deep autoregressive generative model, the conditionals are specified as parameterized functions, so they can be model using neural networks.

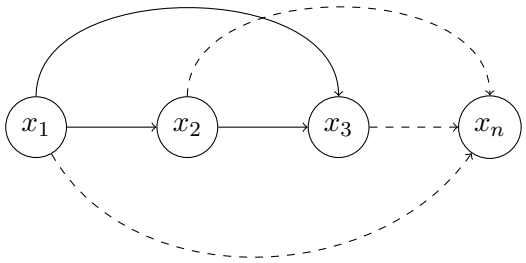


Figure 3. Representation of p_{data} using Bayesian Network.

4.1 Learning and Sampling

The main objective is to minimize the divergence between training distribution $p_{data}(\boldsymbol{x})$ and the model distribution $p_\theta(\boldsymbol{x})$:

$$\min_\theta D_{KL}(p_{data}, p_\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log p_{data}(\boldsymbol{x}) - \log p_\theta(\boldsymbol{x})]$$

Since the minimization is with respect to the parameters of the neural network, the previous equation is equivalently to the following:

$$\max_\theta \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log p_\theta(\boldsymbol{x})]$$

If points are independent and identically distributed drew from the dataset $\mathcal{D}$, then it corresponds to the Maximum Likelihood Estimators:

$$\max_\theta \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}} \log p_\theta(\boldsymbol{x}) = \mathcal{L}(\theta|\mathcal{D})$$

Since the *chain rule* of probability holds the loss of the neural network is:

$$\max_\theta \frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{x}\in\mathcal{D}}\sum_{i=1}^{n} \log p_{\theta_i}(x_i|x_{<i})$$

The disadvantage of this model consists in the sampling procedure because it is a sequential procedure. It consists in starting with a first sample x_0, then sample x_1 conditioned on x_0 and so on. For *high dimensional* data produce a synthetic sample is an expensive and time consuming procedure.

5 Normalizing Flow Models

The aim of Normalizing Flow Models is to start from *simple* distribution, which are easy to sample and evaluate, to get *complex* one which are learned via data [3]. This is done by using the **change of variables** formula as explained in Fig. 4.

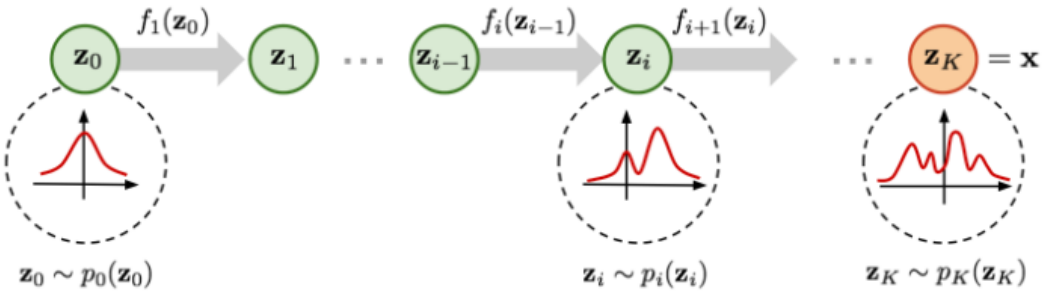


Figure 4. Sequence of changing variables. [14]

If Z and X be *random variables* which are related by a mapping: $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that $X = f(Z)$ and $Z = f^{-1}(X)$. Then:

$$p_X(x) = P_Z(f^{-1}(x)) \underbrace{\left| det \left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|}_{\text{Jacobian Matrix}}$$

The *Jacobian Matrix* is responsible to normalize the distribution of Z after applying the transformation f. It can be seen as **change of volume**.

Sequence of invertible transformations, $f_k : \mathbb{R}^D \rightarrow \mathbb{R}^D$. Starting with a known distribution $\pi(z_0) = Z_0 \sim \mathcal{N}(0, I)$ and apply sequentially the invertible transformations to obtain a flexible distribution.

$$p(x) = \pi(z_0 = f^{-1}(x)) \prod_{i=1}^{K} \left| det \frac{\partial f_i(z_{i-1}}{\partial z_{i-1}} \right|^{-1}$$

5.1 How to model the Invertible Transformation?

Neural Networks can been used, but they must have two characteristics:

1. The transformations must be invertible.
2. Computation of $\sum_{i=1}^{K} \ln |J_{f_i}(z_{i-1})|$ can be intractable for an arbitrary sequence of invetible transformations.

In this survey I will present one of the most popular RealNVP [5]. It is composed by two main building blocks: a **Coupling Layer** and **Permutation Layer**.

Coupling Layer

Consider an input that is divided into two parts $X = [X_A, X_B]$, where $X_A = X_{1:d}$ and $X_B = X_{d:D}$. The transformation is defined as follows:

$$Y_A = X_A$$
$$Y_B = exp(S(X_A)) \odot X_B + T(X_A)$$

where $S(\cdot)$ and $T(\cdot)$ are arbitrary neural networks. This transformation is invertible by design:

$$X_A = Y_A$$
$$X_B = (Y_B - T(Y_A) \odot exp(-S(Y_A))$$

The problem is that we process only *half* of the input, therefore, we must think of an appropriate additional transformation a coupling layer could be combined with.

Permutation Layer
An effective transformation that could be combined with a coupling layer is permutation layer. It is done to change the order of the variables.

5.2 Training of Normalizing Flows

As the AutoRegressive models the objective function is, given a set of parameters $\mathcal{M}$, maximize the *expected* log-likelihood:

$$\arg\max_{\theta \in \mathcal{M}} \mathbb{E}_{x \sim p(x)}[\log p_\theta(x)]$$

which can be transformed in the following equation by applying the change of variables formula:

$$\min_{\theta \in \mathcal{M}} \frac{1}{\mathcal{D}} \sum_{x \in \mathcal{D}} \left[\frac{1}{2} ||G_\theta(x)||_2^2 - \log |det \nabla_x G_\theta(x)| \right]$$

Where $G_\theta(x)$ is the output of the normalizing flow network.

5.3 Advantages of Normalizing Flows

There are two main advantages introduced by the Normalizing Flows with respect to the GANs and VAEs:

1. The training process is more *stable*, with respect to find the Nash Equilibrium of the adversarial training.
2. The convergence is *easier* and *faster*.

6 Diffusion Models

This section will present another kind of generative model which are the Diffusion Models [8]. The general idea behind this new model is very simple and it is basically can be represented by two steps:

1. Take a *real* image, x_0, from the training distribution and successively add Gaussian noise. This first step can be graphically visualized by Fig. 5. The $q(x_t|x_{t-1})$ is conditional probability of obtaining image x_t given image at step $t-1$.

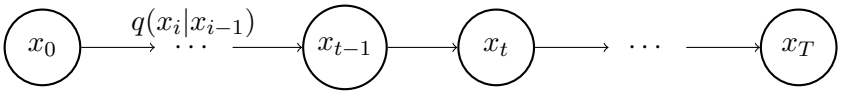


Figure 5. Injection of noise.

2. Then lean to recover the data any **reversing** the noising process. The reverse process is often indicated with the following probability $p_\theta(x_{t-1}|x_t)$, as represented in Fig. 6.

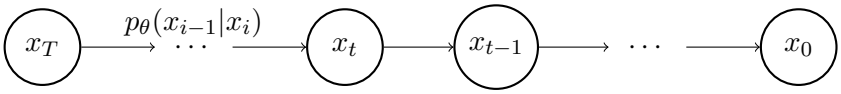


Figure 6. Denoising process.

After that the Diffusion Model can be used to generate data by simply passing randomly sampled noise though the learned denoising process.

This "**destroying** process" continue until the image is asymptotically transformed to pure Gaussian noise. When the added noise is sufficiently low and it is drawn from a Gaussian distribution, combining this fact with the Markov Chain rule to learn the reverse process leads to a simple repatametrization.

The training process of the diffusion models consists in minimizing the Variational upper bound on the negative likelihood of the training data, which can be mathematically expressed as:

$$\mathbb{E}[-\log p_\theta(x_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] = \mathcal{L}_{vlb}$$

6.1 Example

A recent example of diffusion model is DALL-E 2 [12], which is a machine learning model produced by OpenAI which is able to generate new synthetic images starting from natural languages captions. It is a diffusion model conditioned on the robust representation given by separate model called CLIP (Contrastive language Image-Pre training) [11].

7 Energy Based Models (EBM)

In this section I will introduce the *energy based models* (EBMs) [10] which leverage the power of the neural networks to learn an energy function $E_\theta(\boldsymbol{x})$ that assign low energy values to inputs $\boldsymbol{x}$ in the data distribution.

Given a training datapoint $\boldsymbol{x}$, let $E_\theta(\boldsymbol{x}) \in \mathbb{R}$ be the **energy function**, it defines a probability distribution through the use of Boltzmann distribution [1]:

$$p_\theta(x) = \frac{\exp(-E_\theta(\boldsymbol{x}))}{Z_\theta}$$

where $Z_\theta = \int exp(-E_\theta(\boldsymbol{x}))dx$ is the **partition function** which is used to ensure that the probability integrates to 1.

As in the previous methods the goal is to maximize the maximum likelihood estimator:

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim p_{data})}[-\log p_\theta(x)] = \mathbb{E}x \sim p_{data}\left[-\log(E_\theta(x) - \log(Z_\theta))\right]$$

This equation is well known to have the following gradient:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{x^+} \sim p_{data}[\nabla_\theta E(x^+)] - \mathbb{E}_{x^-\sim p_{data}}[\nabla_\theta E_\theta(x^-)]$$

This gradient decreases energy of the positive samples x^+, while increasing the energy of the negative samples x^- from the model p_θ.

The Energy Based Models have a simple and stable training. Despite this advantage, EBM models suffer of the *curse of dimensionality*, so they are difficult to train in high dimensional data domains.

8 Conclusion

In this survey I have summarized the main generative models techniques that can be used to generate synthetic data, including GANs, VAEs, Normalizing Flows, AutoRegressive and Diffusion models. Each of these models has its own *strengths* and *weaknesses*, and it is important to choose the right model for the task at hand.

GANs are powerful generative models, but they can be difficult to train. Since, they try to find the Nash Equilibrium of a minimax problem.

VAEs are trained to maximize the likelihood of the data and they use a Gaussian prior for the latent variables, while GANs do not have a specific prior.

Normalizing Flows are a type of generative model that is similar to VAEs. However, instead of using a Gaussian prior, normalizing flows use a transformation of the data to make it more Gaussian-like.

AutoRegressive models are another type of generative model. They are similar to VAEs in that they use a latent variable model. However, AutoRegressive models are trained to predict the next value in a sequence.

EBMs use a energy function which defines the probability of a data point x being generated by the model. A low energy means that the data point is likely to be generated by the model, while a high energy means that the data point is unlikely to be generated by the model.

Diffusion models have the ability to handle high-dimensional data and the ability to learn complex distributions Additionally, diffusion generative models are relatively efficient to train, and can be trained on small datasets.

References

[1] J. S. Rowlinson *. The maxwell–boltzmann distribution. *Molecular Physics*, 103(21-23):2821–2828, 2005.

[2] Aditya Grover and Stefano Ermon. Autoregressive models. `https://deepgenerativemodels.github.io/notes/autoregressive/`, 2018.

[3] Aditya Grover and Stefano Ermon. Normalizing flow models. `https://deepgenerativemodels.github.io/notes/flow/`, 2018.

[4] Devin Sonin. Introduction to bayesian networks. `https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e`, 2018.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

[6] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *ICCV*, volume 3, pages 487–493, 2003.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020.

[9] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition, pages 4401–4410, 2019.

[10] Yann Lecun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. *A tutorial on energy-based learning*. MIT Press, 2006.

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021.

[13] Joseph Rocca. Understanding generative adversarial networks (gans), Jan 2019.

[14] Lilian Weng. Flow-based deep generative models. *lilianweng.github.io*, 2018.

[15] Wikipedia contributors. Autoencoder — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=1109263923`, 2022. [Online; accessed 1-October-2022].

[16] Wikipedia contributors. Evidence lower bound — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Evidence_lower_bound&oldid=1117816922`, 2022. [Online; accessed 24-October-2022].

[17] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

Adoption of Service Mesh Solutions for Microservice Management

Dennis Marttinen
dennis.marttinen@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

Adopting a microservice-based application architecture introduces many challenges around networking. In a Kubernetes cluster, features, such as load balancing and end-to-end encryption, must be implemented for all microservice components to ensure security and scalability. However, implementing this functionality separately for each microservice is inefficient to maintain and challenging to secure, as every component will accumulate subtle implementation differences. This paper explores service meshes as a solution that is able to consolidate and unify complex networking functionality in a Kubernetes environment. The paper analyzes the rising adoption of and technology behind service mesh solutions, such as Istio and Cilium, and identifies future development patterns for both adoption and the technologies.

KEYWORDS: Service Mesh, Istio, Cilium, Kubernetes, Networking, CNI, SOA, Microservice

1 Introduction

The rising popularity of microservice-based application development models has brought new challenges in deploying and managing software, par-

ticularly in the cloud. While the shift from monolithic approaches to distinct software components has implications for software development practices, an often overlooked aspect of this migration are the changes required to the infrastructure and platforms running these applications.

Compared to monolithic applications, an application consisting of multiple components, such as a database, a web backend, and a web frontend, requires a platform that provides resource management, networking and orchestration. A common solution is to package the application components in software containers that can be orchestrated using container orchestration software, such as Kubernetes [1]. While these orchestration platforms can provide the components with resources, networking and scheduling, this support has traditionally been inflexible out of the box, with extensibility being a main focus [2].

Particularly when desiring to scale the application, each component must start integrating features that allow for example service discovery, traffic encryption, load balancing, circuit breaking, and monitoring. Implementing this functionality separately in each component of the application leads to significant development overhead, duplicate effort, and potentially even security vulnerabilities, as the component attack surface is covertly increased [3].

To alleviate this issue, specialized *service mesh* software has been created [4, 5] that offloads all service related tasks from the individual components. This allows application developers to focus on the core business logic of each component in isolation without worrying about about the requirements and implications of direct service-to-service communication. Service meshes can provide functionality for various purposes [4], including transparent traffic encryption, automatic discovery of available components and resources, monitoring of the health of the service, and load balancing of service traffic without needing to adapt any of the individual services to the current scale and demands of the application. Despite service meshes having broad compatibility with applications and environments, adopting a service mesh implementation might still require adaptation from existing systems [6]. Additionally, careful planning is required to avoid microservice applications becoming tightly coupled with a particular service mesh implementation. However, as evidenced by the increasing adoption of service meshes [7], the advantages of adoption seem to outweigh these drawbacks.

This paper describes the rationale for the mass adoption of service mesh

backed microservice architectures as seen in the industry, and details the internals of popular service mesh architectures. Section 2 defines the core concepts and technologies related to a service-oriented architecture, while Section 3 details the cloud native application development methods and how service meshes can help solve key challenges. Furthermore, it details the integration of service mesh solutions with the networking layer. The Istio [8] and Cilium [5] projects are used as case examples of practical service mesh solutions with differing architectures. Finally, Section 4 summarizes the core remarks of the paper and presents the future outlook on service mesh adoption and development from the perspective of the author.

2 Service-Oriented Architecture

In this section, the concept of a service-oriented architecture (SOA) is introduced. Sec. 2.1 describes the need for modular applications in SOA solutions. Sec. 2.2 generalizes the concept of modular applications to modern microservices. Finally, Sec. 2.3 describes how Kubernetes is used to orchestrate microservices and compose SOA services.

2.1 Modular Applications

Service-Oriented Architectures (SOA) have recently attracted attention in cloud software development. Instead of concentrating on the implementation of a monolithic software product, a service-oriented architecture focuses on the development of services [9]. In order to enable fast iteration and a short time to market, the services must consist of easily malleable components that are modular and *loosely coupled*. Loose coupling denotes that the components have no hard dependencies on each other, i.e., they can be both developed and even fully replaced with other equivalent components independently of each other. This requires shifting focus on developing refined interfaces for inter-component interaction [9].

With standardized interfaces, it is also straightforward to re-use the modular software components across different projects and even across different organizations. By leveraging the high-quality open source software ecosystem, SOA solutions can utilize existing software to provide the functionality of many of the requires components of a modular application, and focus on implementing the core business logic of the application

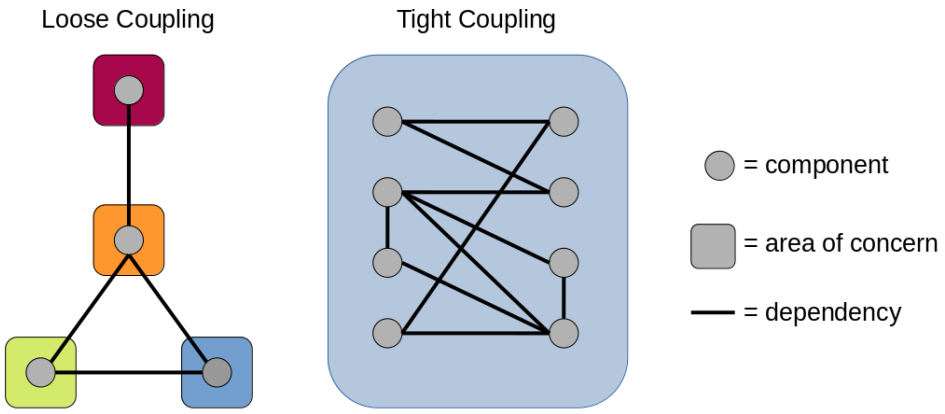


Figure 1. An illustration comparing loose and tight coupling

instead [10]. For example, the Apache Software Foundation [11] maintains a wide range of open source projects [12] that can perform tasks, such as web serving or big data analytics, as part of an SOA solution. These advantages have been a major driver in the adoption of SOA and modular application architectures [13].

2.2 Microservices

Traditional monolithic applications are self-contained, indicating that they bundle all of the application components into a single package that can be deployed on a server without major platform or runtime requirements. In contrast to loose coupling, monolithic applications heavily leverage the induced *tight coupling*, where all components are able to directly interact with each other and access the information provided by other components in a direct way, without utilizing well-defined interfaces. This creates a dense network of internal dependencies, which makes separating an individual component from the bundle non-trivial. This key difference between loose coupling and tight coupling is visualized in Figure 1.

The term *microservice* was coined to describe a single software component that cooperates with other microservices to perform the tasks of a traditionally monolithic application [14]. In a larger application, a single microservice may fulfill the role of, for example, access control, business logic, or data storage. For this purpose, the microservice is equipped with the minimal amount of dependencies, often containing only the necessary software libraries, with all other data and commands being externally sourced. Microservices are thus by construction modular and loosely coupled, as they only provide functionality and process data that is directly

related to the role, relying on standardized interfaces for all external communication. In the example representation of a loosely coupled architecture in Figure 1 each node in the graph could be realized as a microservice, since each of them addresses a separate concern. While it is technically possible to implement this kind of *separation of concerns* approach in the tight coupling example, with each component depending on information provided by almost all other components, the principle of loose coupling starts to loose its meaning. As updating the interface exposed by one microservice requires similar changes at the receiving ends of other microservices, updating a single microservice would in this kind of architecture mandate changes in almost all components of the system.

2.3 Kubernetes

Compared to monolithic services, microservices carry an inherent complexity in their architecture. With each of the components of a traditional web service divided into separate *software containers*, managing the resource needs and communication between the components becomes a challenge [15]. To manage this complexity, platform orchestration solutions, such as Docker Swarm [16] and Kubernetes [1] were introduced. Their purpose is to allow holistic deployment of microservice applications by abstracting away resource management, networking, and workload scaling. Kubernetes is currently the most widely used container orchestration platform [7]; consequently, this paper will focus on the deployment and development of service mesh architectures targeting Kubernetes.

In order to be orchestrated by Kubernetes, a software container must be distributed using images that follow the Image Format Specification [17] specified by the Open Container Initiative (OCI) [18]. Subsequently, Kubernetes manages containers sourced from these images by following the OCI Runtime Specification [19]. Furthermore, inter-container communication is enabled by following the Container Network Interface Specification [20].

These specifications denote the lowest common denominator between Kubernetes and the containers, but the platform provided by Kubernetes provides a set of abstractions to better adapt containers for SOA solutions. In addition to providing platform level features, such as grouping, resource management, and extensibility using APIs, Kubernetes provides primitive abstractions for workload and service management. Instead of a single container, the primitive form of workload in Kubernetes is called

a *Pod* [21]. A Pod consists of one or more software containers that share a network namespace, i.e., to the workloads running the containers the environment looks like they are running on the same host, and they can, for example, access the ports exposed by each other directly. Containers within a single Pod can thus be considered as tightly coupled [21], and are most suitable for running individual sub-components of a single microservice, which is represented by the Pod as a whole.

Kubernetes concerns itself with scaling Pods within and across a cluster of compute nodes, such as physical servers. At the same time, downtime should also be avoided if, for example, any server spontaneously shuts down due to a hardware failure. The natural way to fulfill these requirements from the perspective of a Pod is to run multiple instances of it in parallel, potentially across different availability zones [22]. As state management in a microservice architecture is typically delegated to a small amount of microservices, such as the ones responsible for managing a database, most microservice Pods can be replicated without side effects. If the components of the application are sufficiently loosely coupled, this replication is trivial, as any of the replicated instances can respond to a query addressed to the component.

To expose the replicated pods through a single interface, Kubernetes provides the *Service* abstraction [23]. A Service defines a single network name or address for a *Deployment* [24] of Pod replicas. Crucially, a Service is an abstraction for a set of Pods that run the *same* microservice, resulting in a mismatch between the definitions of a Kubernetes Service and a service in the SOA model. Multiple Kubernetes Services, each abstracting one microservice, are needed to compose a service fit for an SOA application. This composition is performed by a dedicated software component called a *service mesh*.

3 Service Meshes

Section 3 details the cloud native application development methods and how service meshes can help solve key challenges. Furthermore, it details the integration of service mesh solutions with the networking layer.

In this section, the cloud-native application deployment model is detailed in the context of service management. Service meshes are presented as a solution to management challenges, and different service mesh architectures are compared. Sec. 3.1 briefly describes the modern microservice-

based application development and deployment workflow. Sec. 3.2 focuses the challenges involved with microservice networking and how the use of service meshes can alleviate the them. Sec. 3.3 presents the high-level architectures used by current service mesh implementations. Finally, Sec. 3.4 and Sec. 3.5 and present the Istio and Cilium projects as service mesh case examples, and briefly analyze the reasons for their popularity.

3.1 Cloud-Native Application Deployment

In order to realize how service meshes ease the management of cloud native applications, that is, applications built to run in cloud environments, it is crucial to first understand how these microservice-based applications are developed and deployed. Thanks to the loose coupling of microservices, multiple different teams in an organization may work on different application components simultaneously in parallel. When, for example, the frontend team of a web application project developes a new UI feature and the backend team adds support for that feature in the business logic, both teams may independently build their solutions as microservices that get independently tested and packaged into software containers. From here, a Kubernetes Deployment is created with the desired new versions, combined with other configuration that defines Services as well as collections of Services to form the services expected by the SOA model. The end result is an updated application that can now be rolled out customers. Note that this is an oversimplification that is sufficient for the purposes of this paper. There are various tools and technologies involved in the deployment and application composition pipeline, such as GitOps [25], that are not discussed in detail here.

3.2 Service Management Challenges

Consider the above example of a backend and frontend microservice. By definition, the microservices should be loosely coupled, and independent of each other to the point that they could be individually replaced with new components providing equivalent functionality. Since software container environment provided by Kubernetes fundamentally only offers raw networking capabilities to each microservice running in a Pod, a lot of network-related functionality must be duplicated between the frontend and backend microservice. Desired network features that should be implemented by every microservice include, but are not limited to, encrypt-

ing incoming and outgoing communications using TLS to prevent sniffing, load balancing between different instances of the microservice to prevent overloading a single instance, automatic retries and failover within the microservice to avoid downtime, and applying policies that, for example, restrict communication to the intended subset of neighboring microservices to avoid data exfiltration.

Notably, most of these networking features should be identical between every microservice in the application, with potential differences only in their configuration, such as rate limits and failover thresholds. This realization is the core motivation behind service meshes [8]. Thus, the primary purpose of service mesh software is centralizing this functionality to avoid the need for implementing it separately for each microservice. This reduces the development and maintenance overhead of the different teams working on separate microservices. As a consequence, the attack surface of the whole application is reduced, as the amount of subtle differences that could introduce vulnerabilities in the networking code of each microservice is reduced by simply having less code and simpler implementation requirements.

Service meshes can be though of as implementing *separation of concerns* on the networking level, where the concern of network management is delegated from the individual microservices to the service mesh.

3.3 Service Mesh Architecture

Since Kubernetes provides direct IP layer networking to each Pod, service mesh software running on top of Kubernetes must utilize clever tricks to apply the high-level networking features, such as traffic encryption, to microservice Pods. Since Kubernetes mandates the use of CNI to implement its network model [26] and the CNI specification only focuses on enabling network connectivity between containers [20], injecting a service mesh in between to provide high-level networking features is not trivial. Currently, intercepting Pod communication to proxy its container traffic through the service mesh involve either the deployment of *sidecar proxies* or utilizing a Linux kernel feature known as *eBPF*.

Deploying sidecar proxies involves creating a new sidecar container for each container in a Pod. The sidecar container, as the name implies, attaches to the side of the original container, and redirects all network traffic of the original container through itself. Inside the sidecar container resides a proxy, such as Envoy [27], that implements the advanced network-

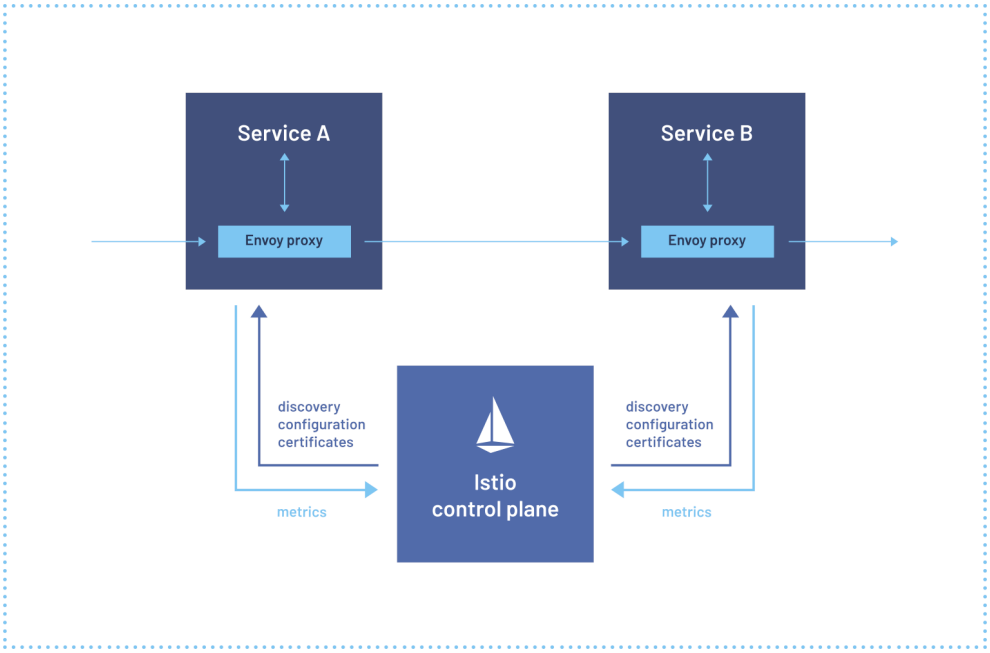


Figure 2. A diagram depicting the architecture of the Istio networking solution [4]

ing functionality as requested by the service mesh configuration. This approach is used by, for example, the Linkerd service mesh [28] as well as the popular Istio service mesh [4].

The other approach of utilizing eBPF on the other hand requires no sidecar injection or other invasive modifications to the application. Currently, this approach is solely implemented by the Cilium networking solution [29]. Consequently, this paper will focus on the use of eBPF from the perspective of Cilium. eBPF itself is a sandboxed execution environment in the Linux kernel that essentially enables the implementation of application logic at kernel level [30]. By implementing the proxy system from the sidecar model in eBPF instead, Cilium can provide advanced network functionality significantly more transparently.

To analyze these two approaches in more detail, Sections 3.4 and 3.5 will detail the design of Istio and Cilium respectively. Importantly, these sections highlight the key service mesh features behind their popularity, as well as discuss future aspects of development in these service mesh solutions.

3.4 Case Example: Istio

The Istio service mesh [4] is currently the most widely deployed service mesh solution with about a 33% usage rate in production environments [7]. Istio is at its core divided into a *control plane* and a *data plane* [4].

The control plane is implemented using the Kubernetes controller pattern [31], where the control plane applications of Istio run in containers inside the Kubernetes cluster and constantly watch for (changes in) Kubernetes objects [32] that are relevant for configuration of the service mesh. The control plane is then responsible for applying this configuration to the data plane, which consists of the individual Envoy [27] proxies running in the sidecars. The whole architecture is visualized in Figure 2. For example, when enabling mutual TLS (mTLS) traffic encryption between two endpoints in the service mesh, the control plane shares the relevant secrets with the proxies, enabling end-to-end security between the endpoints.

In addition to streamlining the microservice development workflow, Istio has seen a rise in popularity due to the *observability* it enables [8]. Observability in the context of cloud environments is the ability to observe and monitor the health and state of the individual components of a cloud system. Observing network data flows is a crucial indicator for workload characteristics, such as service health, performance, and load, especially for distributed microservice-based applications. Since observability features offered by CNI implementations are often limited due to the simplistic focus of the specification, service meshes in general have a unique position to provide this type of additional functionality in a global context in addition to the networking extensions.

3.5 Case Example: Cilium

Cilium, as mentioned in Section 3.5, is not a service mesh in the traditional sense. Instead, it is more akin to an advanced CNI networking implementation that contains the functionalities expected from a traditional service mesh, such as transparent encryption, load balancing of services and high-level service traffic observability [5]. Compared to Istio, Cilium leverages eBPF [30] for providing proxy-like functionality in the Linux kernel context instead of relying on sidecar containers. Due to this architecture, which depicted in detail in Figure 3, leveraging the advanced features provided by Cilium mandates also mandates that it acts as the CNI implementation of the Kubernetes cluster. This makes adoption of Cilium more complicated compared to Istio, which can be installed on top of any existing CNI solution the target cluster may have.

However, the unified networking and service mesh architecture provided by Cilium also offers unique advantages, notably the lack of sidecar

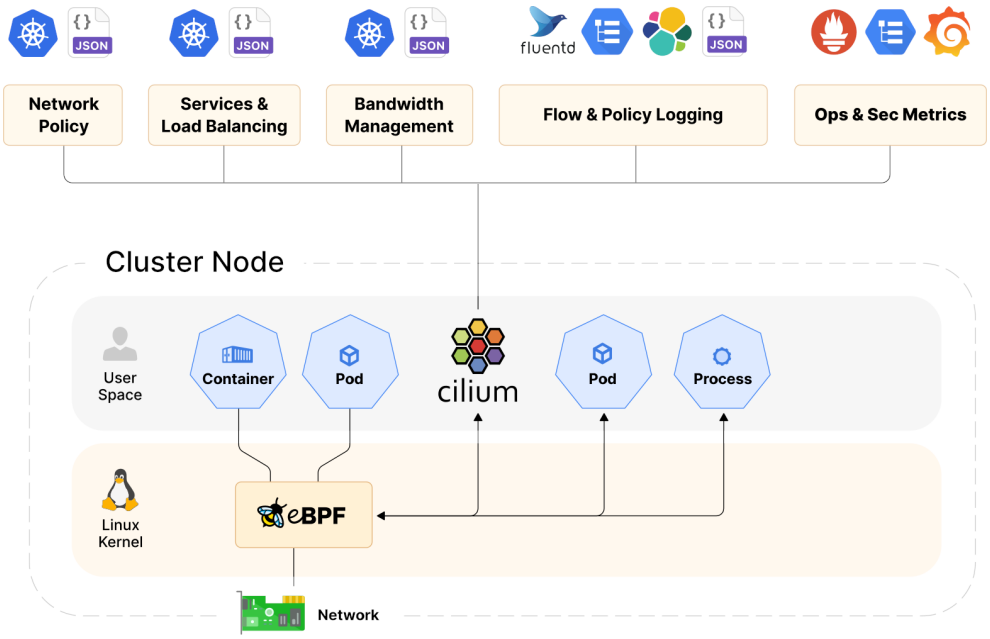


Figure 3. A diagram depicting the architecture of the Cilium networking solution [29]

containers. Injecting the sidecar containers as performed by Istio requires network administration privileges in the target pod [33]. This causes issues in, for example, clusters that enforce strict security policies. In contrast, providing all networking and service mesh functionality through CNI sidesteps these issues entirely as the native Kubernetes networking behavior is left unchanged from the perspective of containers.

Despite Cilium still being a CNCF incubation level (first stage of maturity) project, it is already widely used by large-scale cloud vendors, such as Google and AWS [5]. As it resides on a comparatively low abstraction level in the networking stack, Cilium has detailed visibility into the traffic flowing between workloads in the cluster, which enables detailed observability metrics required for large-scale deployments. Performance measurements also indicate Cilium to be competitive with other CNI solutions, particularly when considering the inter-host routing performance.

4 Discussion

This section presents the author's view and opinions.

The continually increasing adoption of microservice architectures paired with the rising popularity of cloud applications presents many new challenges related to deploying, orchestrating and observing software. However, rather than abandoning these daring new concepts, the community

and companies continue advancing the technologies and finding solutions to the incurred problems. Kubernetes and its networking model CNI have established a scalable platform for deploying cloud-native containerized software, although the exposed interfaces and interactions demand more depth to be practical for many real-world environments. Service meshes enrich these interface while simultaneously delegating the complexity of network management to themselves, enabling individual microservices to focus on the core application logic. The advantages of this approach are evident on both the organizational and cluster level, where the service mesh can reduce development and maintenance overhead for individual microservice development teams and provide them with better observability in the cluster.

The introduction of the Cilium service mesh with its eBPF-based architecture might be the start of a paradigm shift in service mesh technology. Unifying the currently distinct network and service mesh components in a Kubernetes cluster eliminates the need for sidecar injection, which, in addition to being a “hack”, introduces issues in security-critical environments. With the distinct advantages of making service mesh features, such as load balancing and transparent encryption, unconditionally available to the whole cluster, not only is the overhead of enabling secure and scalable deployments reduced, the cluster configuration is also simplified and the number of separate components is decreased. Assuming that unified CNI and service mesh implementations can prove themselves as secure and performant, companies around the world will likely begin mass adopting them in the near future.

References

[1] Kubernetes – Production-Grade Container Orchestration. https://kubernetes.io/, September 2022.

[2] Kubernetes – Extending Kubernetes. https://kubernetes.io/docs/concepts/extend-kubernetes/, November 2022.

[3] C. -A. Chen. With Great Abstraction Comes Great Responsibility: Sealing the Microservices Attack Surface. In *2019 IEEE Cybersecurity Development (SecDev)*, https://doi.org/10.1109/SecDev.2019.00027, pages 144–144, September 2019.

[4] Istio – The Istio service mesh. https://istio.io/latest/about/service-mesh/, November 2022.

[5] Cilium – Linux Native, API-Aware Networking and Security for Containers. https://cilium.io, October 2022.

[6] W. Li, Y. Lemieux, J. Gao, Z. Zhao, and Y. Han. Service Mesh: Challenges, State of the Art, and Future Research Opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, https://doi.org/10.1109/SOSE.2019.00026, pages 122–1225, April 2019.

[7] CNCF – CNCF Annual Survey 2021. https://www.cncf.io/reports/cncf-annual-survey-2021/, November 2022.

[8] Istio – Istio Documentation. https://istio.io/latest/docs/, September 2022.

[9] Derek T. Sanders, J. A. Hamilton, and Richard A. MacDonald. Supporting a service-oriented architecture. In *Proceedings of the 2008 Spring Simulation Multiconference*, SpringSim '08, ISBN 978-1-56555-319-4, pages 325–334, San Diego, CA, USA, April 2008. Society for Computer Simulation International.

[10] Jeff Davis. *Open Source SOA*. Manning Publications Co., USA, ISBN: 978-1-933988-54-2, 1st edition, 2009.

[11] The Apache Software Foundation. https://apache.org/, October 2022.

[12] Apache – Apache Projects List. https://projects.apache.org/projects.html, October 2022.

[13] Dusko Vukmanovic and Damir Kalpic. Key practices for SOA adoption. https://www.researchgate.net/publication/262152641_Key_practices_for_SOA_adoption, pages 20–25, May 2012.

[14] Keith D. Foote. A Brief History of Microservices. https://www.dataversity.net/a-brief-history-of-microservices/, April 2021.

[15] Daniel Muresu. *Investigating the Security of a Microservices Architecture : A Case Study on Microservice and Kubernetes Security*. http://urn.kb.se/resolve?Urn=urn:Nbn:Se:Kth:Diva-302579. 2021.

[16] Docker – Swarm mode key concepts. https://docs.docker.com/engine/swarm/key-concepts/, September 2022.

[17] Open Container Initiative – Image Format Specification. https://github.com/opencontainers/image-spec, October 2022.

[18] Open Container Initiative. https://opencontainers.org/, October 2022.

[19] Open Container Initiative – Runtime Specification. https://github.com/opencontainers/runtime-spec, October 2022.

[20] Container Network Interface – Specification. https://www.cni.dev/docs/spec/, October 2022.

[21] Kubernetes – Pods. https://kubernetes.io/docs/concepts/workloads/pods/, October 2022.

[22] Kubernetes – Running in multiple zones. https://kubernetes.io/docs/setup/best-practices/multiple-zones/, October 2022.

[23] Kubernetes – Service. https://kubernetes.io/docs/concepts/services-networking/service/, October 2022.

[24] Kubernetes – Deployments. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/, October 2022.

[25] WeaveWorks – GitOps what you need to know. https://www.weave.works/technologies/gitops, November 2022.

[26] Kubernetes – Network Plugins. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/, November 2022.

[27] Envoy Proxy. https://www.envoyproxy.io/, November 2022.

[28] Linkerd. https://linkerd.io/, November 2022.

[29] Cilium – Get Started. https://cilium.io/get-started/, November 2022.

[30] eBPF – What is eBPF? An Introduction and Deep Dive into the eBPF Technology. https://www.ebpf.io/what-is-ebpf, November 2022.

[31] Kubernetes – Controllers. https://kubernetes.io/docs/concepts/architecture/controller/, November 2022.

[32] Kubernetes – Understanding Kubernetes Objects. https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/, November 2022.

[33] Istio – Application Requirements. https://istio.io/latest/docs/ops/deployment/requirements/, November 2022.

Recent advances in Reinforcement Learning: Upside Down Reinforcement Learning

Elias Peltonen
elias.m.peltonen@aalto.fi

Tutor:Anton Debner

Abstract

Upside Down Reinforcement Learning (UDRL) is recent Reinforcement Learning (RL) method, which turns the reinforcement learning upside down. This is achieved by rather than predicting rewards, UDRL maps rewards to actions. Experiements with UDRL look promising, as it already can out perform other RL methods in some machine learning problems. As a recent advance in the field, the material about the subject is limited.

KEYWORDS: Reinforcement Learning, Upside Down Reinforcement Learning, Machine Learning, Q-learning

1 Introduction

Commonly machine learning can be categorized to supervised-, unsupervised- and reinforcement learning. Reinforcement learning (RL) is an area of machine learning where the agent to be trained attempts to find the best action that maximizes rewards. Data sets that the agent uses for learning, are not labeled like in supervised learning, nor does the agent get feedback to indicate whether a decision is correct or not [3]. Rather the agent learns by interacting with the environment. Training the agent to make correct decisions in different situations can be difficult. And as RL

focuses on maximizing short term rewards, it in its standard form is not enough in some cases. Upside down machine learning is recent alteration of RL where the agent is not trained to just find highest rewards, but to follow commands such as obtain X amount of reward in Y time [7].

This paper first explains what reinforcement learning is and where it can be applied. This is followed by description of UDRL, explanation of how agents are trained in UDRL and brief look where it can be applied. The research was done as a literature review.

2 Reinforcement Learning

One of the most common ways of learning, is by trial and error. This means attempting repeated and varied actions until success is achieved, or the practicer gives up. Process requires the practitioner to learn from failures, and improve their future actions. Luckily for every action, there exists a wealth of information about cause, effect and consequences of that action [8]. Such effects and consequences can be roughly categorized into positive rewards and negative punishments. By interacting with the environment, one can learn optimal actions that lead to most positive outcomes. Furthermore, one can predict future outcomes based on actions taken in similar states of the environment.

In this section, the paper explains the basic idea of Reinforcement Learning, and some notable applications.

2.1 Basic Idea

Decision making environments are characterized by three main elements [2]. First: state space, which describes the current state of the environment. For example, the current locations of chess pieces on the chess board. Second: actions, these are the possibilities of how decision-maker can affect the state space, and they produce an outcome [2]. On a chessboard, actions would be the legal moves players pieces can make. Third: outcomes, they are the consequences of actions that are assumed to have numerical utilities that can change according to the motivational state of the decision-maker [2]. In the chess board example, capturing pieces are the positive outcomes, and losing pieces the negative. Value of those gains and losses depend on the piece, but also the strategy the player is going for (motivational state). Reinforcement learning algorithm then takes a

state, and learns the best possible action that maximizes the reward. In essence RL is "how to map situations to actions" [8]. In RL process, the learner is not told which actions to take, but instead discover actions with most rewards by trying them [8].

2.2 Q-learning and memory replay

Most RL problems can be formulated via a Markov Decision Process (MDP). Given an MDP environment, one can use dynamic programming algorithms to compute optimal policies, which lead to the highest possible sum of future rewards at each state. Dynamic programming then can be split into two main branches: value iteration and policy iteration. As UDRL sources which are referenced later focus on value iteration, it is more suitable to explain. One of the most common reinforcement learning algorithm is the Q-learning. Q-learning is based on value iteration, and is defined by:

$Q^{new}(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$ [8].

Where S is state, A is action, $Q(S_t, A_t$ is the current value, α is learning rate, r reward, γ discount factor $max_a Q(S_{t+1}, a)$ estimate of optimal future value. In other words, Q function finds the action which maximizes the immediate reward plus maximum future reward for the next state. Q-learning is usually most suitable for smaller discrete problems.

Also worth noting, one possible way to improve reinforcement learning is to add technique Experience replay [4]. This allows the RL agent to learn from earlier memories to decrease the learning time, and the chances of encountering undesirable situations resulting in a more stable learning process [4]. In standard RL algorithms previous experiences (e.g. tuple of (state, action, reward, new state)) are removed from memory once the algorithm reaches the next state. Experience replay adds a memory buffer to the algorithm that stores n amount of experiences before they are discarded.

2.3 Applications

Reinforcement learning can be applied to a variety of fields such as: recommender systems, finance, games, healthcare systems, robotics and more. Although applying RL to real world problems can be difficult. Li [3] presents 7 step procedure of how to utilize RL:

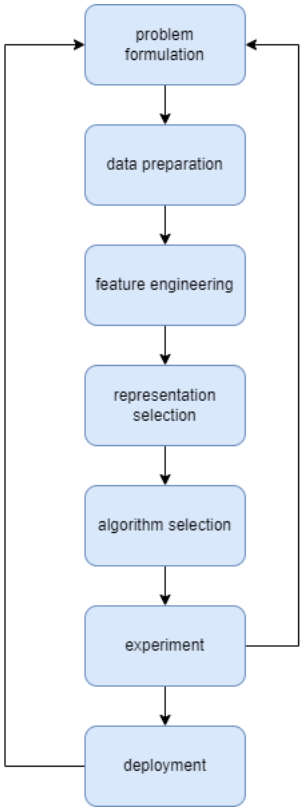


Figure 1. [3]

To apply RL to real life application, first the RL problem needs to be formulated. Environment, the agent, states, actions and rewards must be properly defined. Necessary amount of data also must be provided for the learning to happen. If collecting real world data becomes difficult or infeasible, it is possible to gather data through simulations. If needed data preparation is followed by feature engineering, in which data is transformed to form which is easier to interpret. Followed by how to represent the problem. Choices such as: Should neural networks be used, or what kind of neural networks represent the value functions or the policy the best [3]. After this, the most suitable algorithm is selected. There are many RL algorithms to choose from, online or offline, on- or off-policy, model-free or model-based, etc. [3]. Once these steps are completed, experiments are conducted to tune parameters in order to improve perfor-

mance. Previous steps can also be altered iteratively to improve results. Once decent performance is reached, a trained RL model is deployed. In this step, it is also possible to iterate steps before it to improve the model.

Personalized web services are one example where RL models can be applied. To increase relevance of news articles or advertisements shown to users, RL model can be trained to use users' interests and preferences for better recommendations. This problem has been formalized as a contextual bandit problem, with objective to maximize total number of clicks on a website, and so apporiate policies exists. Although policies derived from contextual bandit formulation are greedy in the sense that they only care about short term results [8]. Issue being that the policy treats every user of certain website as new user, while the user may be revisiting user. For example, it could be profitable for enterprise to alter discounts or displayed items based on whether the user is new user, or is frequent user of their online store. Theocharous et al. [9] compared the results of two algorithms. First algorithm was greedy algorithm that maximized probability of immidiate clicks. Second one was a reinforcement learning algorithm based on a Markovian decision problem formulation, which was aimed to take into account multiple visits to a website [9]. The greedy algorithm was based on mapping click probability with user features. The mapping was learned by using random forest (RF) algorithm with supervised learning. The RL algorithm used fitted Q iteration (FQI) and the same random forest algorithm. RF was used for greedy optimization, and the FQI evaluated best policies using a validation training set [9]. Final policy was then the one with best policy produced by FQI with the initialaction-value function set to the mapping produced by the RF for the greedy optimization approach [9]. Result proved to be promising as Adobe announced in 2016 that the second algorithm would be a standard component of the Adobe Marketing Cloud as the policy was more likely to yield higher returns then the one focusing on short term results [8].

3 Upside Down Reinforcement Learning

Upside Down Reinforcement Learning is one recent alteration on Reinforcement Learning. While standard RL focuses on maximizing rewards, UDRL can be modified to focus on different commands. As UDRL is very recent, there is not a lot of existing literature on the subject. This section

discusses basics of UDRL and provides an application example.

3.1 Basic Idea

As already mentioned, standard RL takes actions and observations as inputs, and transforms those into predictions of expected rewards of said actions [5]. RL agents get feedback about how useful actions are, but no information about which actions are best for a given situation [7]. Upside Down Reinforcement Learning (UDRL) instead takes observations and commands (desired rewards and time horizons) as inputs, and outputs actions [5]. This could mean in the chess board example gaining value lead of R (desired reward), in T turns (time horizon). UDRL would then output actions that lead to desired outcomes. Compared to standard RL this allows the agent to follow commands, rather than attempting to maximize rewards in any given situation [7]. Usage of commands as inputs allows the agent to perform more flexible actions. For a simple example, one could train an electric vehicle to reach a specific destination from some starting location. Standard RL agent could learn to do this in least amount of time or by achieving the goal with as much battery life left as possible (or both). While UDRL agent could be taught for example to arrive to destination in under t minutes, while having battery life over x%. Although that is not all UDRL is suitable for. Pilot versions of UDRL successfully outperformed standard RL models on some challenging issues.

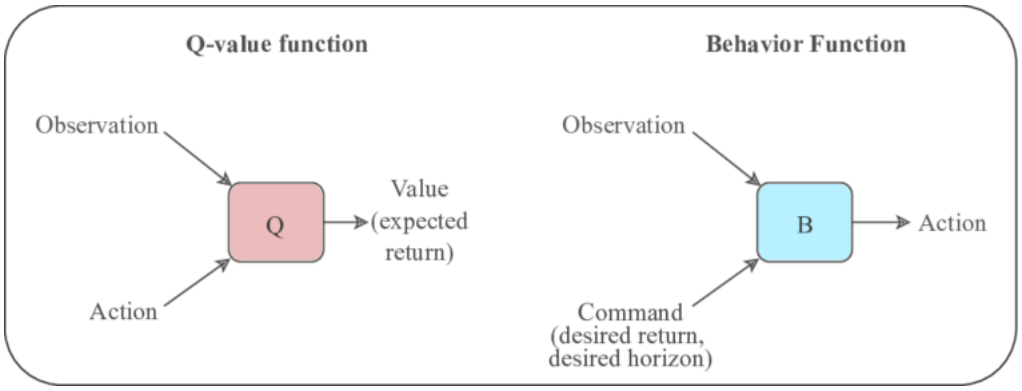


Figure 2. [5]

3.2 Training the agent

Schmidhuber [5] split training the agents based on the environment to: deterministic, probabilistic and partially observable environments.

They consider deterministic environments as such that there exists a Markovian interface between the controller and the world. In other words, this means that the controller knows everything about the current state, and can compute probabilities of next actions and their rewards [5]. This also means that a simple feedforward neural network (FNN) is enough for the controller.

Schmidhuber [5], introduces a high level algorithm for UDRL in a deterministic environment. It contains two algorithms running parallel exchanging information. Algorithm 1's object is to "Generalizing through a copy of C (with occasional exploration)", and algorithm 2's "Learning lots of time cumulative reward-related commands" [5]. Algorithm 1's function is to increase algorithm 2's training set by find paths in horizon(t) "the remaining time", that complete desire(t) "the desired cumulative reward" in time t. Controller C then outputs probability distribution out(t) of next possible actions. Algorithm 1 then chooses the most probable actions from out(t) as out'(t) if not set into exploration mode. If algorithm 1 is set to exploration mode it can choose random action to create diversity in the training set. Occasionally algorithm 1 transfer latest observations to algorithm 2 to increase A2s training set [5]. Algorithm 2 then uses this training set with Replay-based training on previous behaviors and commands compatible with observed time horizons and costs [5]. Replay-based learning in this context means that the algorithm stores and repeatedly presents the previous behaviors and commands to the gradient descent-based backpropagation, to improve the data-efficincy of the algorithm [5] [1]. Before and after this, the algorithm 2 synchronizes with algorithm 1.

In probabilistic environments the algorithm remains mostly the same. The desired cumulative reward desire(t) instead becomes expected immediate rewards, and the reward is independent of the history of previous actions [5]. Unless the randomness is affecting not only the next immediate rewards, but also affecting the next state, then it is possible to estimate cumulative expected rewards [5].

Partially observable environments are non Markovian environments, where the controller has incomplete information of the current state of the world [5]. This also means that the simple feed forward neural networks may no longer be sufficient, and recurrent neural networks (RNN) or similar general purpose computer is needed [6]. With RNN the it is possible generate meaningful representation of the current enviroment by translating

entire history of actions and observations [5]. By utilizing RNN, the algorithm 2 can and must be modified to take into account the entire history to certain time step.

3.3 Applications

While no papers of UDRL being used in real world application were found, it has been applied to common learning settings such as LunarLander problem. LunarLander problem is a task to control lander to land in the landing pad. Landing pad stays in some fixed location, and the lander has possibilty to use 4 actions: do nothing, fire the left orientation engine, fire the main engine, fire the right orientation engine. By moving away from the landing pad the agent loses reward, and gains it by closing the distance or depending on how much of the lander in on the landing pad once the episode ends (the lander crashes or comes to rest). Srivastava et. al. [7] compared performance of URDL in this problem against Double Deep Q-Networks (DQN) and Advantage Actor-Critic (A2C). They note that UDRL performed similar to DQN but both algorithm were behind of A2C in sample complexity and final returns. But when running additional experiements with sparse delayed rewards (In this case delaying all rewards until last step of each episode and while rewards in all other time steps being zero), results showed that UDRL was unaffected by the change, while other algorithms became unstable and slow [7].

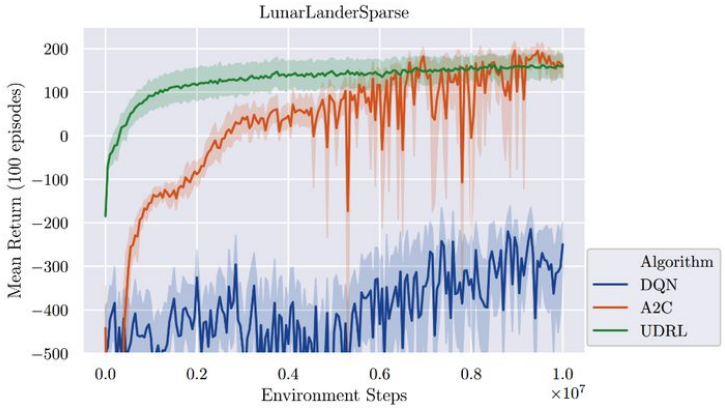


Figure 3. [7]

As can be seen in figure 3, DQN performed poorly and A2C had large variations in performance, while UDRL achieved good and stable performance.

Aside from experiements with machine learning problems, as an concept

example of real world application, Schmidhuber [5] presents use of UDRL in trainining a robot. This is done by making the robot imitate a human by recording humans movements. The recording would then be divided into separate frames. These frames act as sequential commands and are transformed into tuples of extra(), desire(), horizon(), and work as input to an RNN controller resulting in supervised learning which the robot must learn to imitate [5]. Controller then maps the tasks as rewards to corresponding actions. Once robot has learned to execute command, it can be set to refine learned behaviour, or test how well robot obeys by series of trials [5].

4 Conclusion

UDRL shortcuts traditional RL methods by replacing reward predicting with mapping rewards to actions. UDRL experiement results are promising, and they show that UDRL can outperform other RL algorithms in some situations. Also while similar behaviour can be taught with other RL methods, UDRL can provide simpler alternative to develop. Unfortunately UDRL still lacks any real world applications, and most of the papers about the subject are from the same author.

References

[1] Sander Adam, Lucian Busoniu, and Robert Babuska. Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):201–212, 2011.

[2] Peter Dayan and Yael Niv. Reinforcement learning: The good, the bad and the ugly. *Current Opinion in Neurobiology*, 18(2):185–196, 2008. Cognitive neuroscience.

[3] Yuxi Li. Reinforcement learning applications, 2019.

[4] Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning. *CoRR*, abs/1710.06574, 2017.

[5] Jürgen Schmidhuber. Reinforcement learning upside down: Don't predict rewards - just map them to actions. *CoRR*, abs/1912.02875, 2019.

[6] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[7] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning, 2019.

[8] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[9] Georgios Theocharous, Philip S. Thomas, and Mohammad Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. In *IJCAI*, 2015.

Recent advances in Reinforcement Learning

Félix Lepeltier
felix.lepeltier@aalto.fi

Tutor: Anton Debner

Abstract

This seminar paper aims to introduce Reinforcement Learning (RL), and more specifically Offline RL, such that the reader can familiarise themselves with the recent advances in this field. Research in Offline RL has seen exponential growth over the recent years, where dealing with the inherent issue of distributional shift has been, and remains one of the main challenges. Addressing distributional shift can be done in several fashions, such as policy constraint methods and uncertainty estimation, both of which are described in this paper alongside their corresponding state of the art methods. To conclude this paper, a potentially promising research direction is suggested to the reader.

KEYWORDS: Machine Learning, Reinforcement Learning, Offline Reinforcement Learning, Batch Reinforcement Learning

1 Introduction

Reinforcement Learning (RL) is a branch of Machine Learning (ML), where an agent aims to maximise a specified reward function, which defines what an agent is supposed to do, by trial and error [12]. Such an agent will therefore interact with the environment and observe the reward that

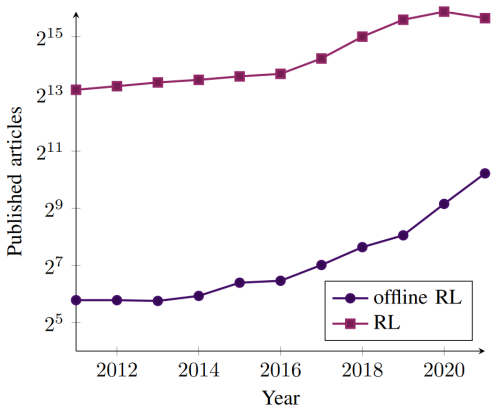


Figure 1. Figure from Prudencio et al.'s survey on Offline RL [12]. Visualisation of the number of publications in RL and Offline RL per annum, from Google Scholar data. The exponential growth in number of publications for Offline RL is clear.

his actions produce. This corresponds to an online learning paradigm, which makes applying RL methods to safety-critical domains such as the autonomous driving and healthcare industries impractical due to the high cost and safety implications of interacting with these environments [12].

Offline RL (equivalent to Batch RL [12]) on the other hand is a data-driven paradigm which learns a policy solely from static datasets of prior experiences, with no further interaction with the environment. This area of RL has experienced consistent exponential growth in the number of published papers over the past five years as Prudencio et al. [12] display in their survey on Offline RL (Fig. 1 visualises this publication growth).

However, Offline RL faces inherent issues. As described in Levine et al's. [8] tutorial article on Offline RL, distributional shift is the main challenge since the distributions under which the learning model is trained and evaluated might differ.

This paper will review the recent advances in Offline RL, and more specifically address the issue of distributional shift. This paper is organized as follows. Section 2 will cover the essential background knowledge required to understand the paper by defining concepts and notation. Section 2.1 will highlight some recent applications in which Offline RL has proven itself to be useful. Section 2.2 will point out the shortcomings and issues which Offline RL faces. Section 2.3 will briefly discuss the focus areas of research in RL over the recent years. Section 3 will cover the main methods used to deal with distributional shift. Section 3.1 will discuss policy constraint methods and current state of the art. Section 3.2 will explain briefly how uncertainty estimation works and highlight recent methods. Section 4 will summarise the seminar paper and discuss a

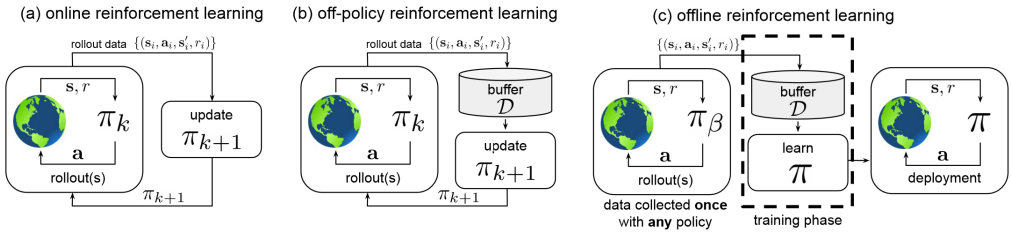


Figure 2. Figure from Levine et al.'s tutorial article on Offline RL [8]. The illustration depicts notable differences between online RL, off-policy RL and offline RL.

potentially promising research direction for Offline RL.

2 Concepts and overview of (Offline) RL

RL relies heavily on the concept of Markov Decision Processes (MDP), since the environment with whom an agent interacts is modelled as such. An MDP is defined as a tuple of five elements $(\mathcal{S}, \mathcal{A}, \rho, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\rho(s_{t+1}|s_t, a)$ is the transition function which defines the dynamics of the MDP [3], $r(s_t)$ is the reward function, and γ is the discount factor. In a RL setting, an agent will aim to maximize the discounted sum of rewards (Eq. 1) in order to learn an optimal policy $\pi^*(a_t, s_t)$. In this equation, L is the length of a trajectory $\tau(s_1, a_1, ..., s_L, a_L)$ corresponding to a sequence of states $s_t \in \mathcal{S}$ and actions $a_t \in \mathcal{A}$.

$$\max_{\pi} \mathbb{E}_{s,a\sim\rho,\pi}\left[\sum_{t=1}^{L} \gamma^t r(s_t)\right] \tag{1}$$

In an Offline RL setting this remains largely similar, the only difference being that the objective must be optimised without further data collection on the fixed set of transitions τ [14].

The differences between online and offline RL can be visualised clearly through fig. 2. Figure 2 a) illustrates classic online RL, whereas c) illustrates offline RL. In online RL the policy (π_k) "is updated with streaming data collected by π_k itself" [8].

Offline RL however, uses a dataset which is collected by some policy π_β which doesn't have to be known, and on which the policy will be trained and deployed only once the training is complete. In other words, training a policy in offline RL does not involve any interaction with the MDP. [8]

2.1 Applications of Offline RL

Applying standard RL methods can be impractical in many domains such as healthcare and robotics due to safety and cost issues when interacting with the environment.

Offline RL however, is a promising candidate in the healthcare industry [4], in robotics [16, 8] where some recent workflows for using it have been proposed [7], and in other domains such as advertising and recommender systems [8, 21].

Some concrete examples within the healthcare industry are the long term planning of treatment plans for schizophrenic patients [15], lung cancer treatment [17] and learning *When-to-Treat* policies [10]. There remains one major downside to using offline RL in healthcare, which is that most data that is collected pertains to patients with more severe outcomes since treatment for less severe cases might not always be needed [4]. An extreme bias such as this one in the data used to train policies means that an agent could jump to false conclusions as it only knows data about severe outcomes.

In robotics, Levine et al. mention in their tutorial article [8] that appealing tasks for offline RL include robotic grasping, robotic manipultion skills and robotic navigation. Oftentimes model-based methods in robotics are trained on real or simulated data (e.g. in the autonomous driving indusry, it would be simulated), where the policy which will be applied to the real system will be extracted from within those models [8].

2.2 Shortcomings of Offline RL

A central issue with Offline RL is distributional shift [8]. The shift between distributions of data a model learns on and those on which it is evaluated is "due both to the change in visited states for the new policy and, more subtly, by the act of maximizing the expected return" [8].

Fujimoto et al.'s [3] Batch-Constrained RL indicates that by inducing a data distribution that is entirely contained within the data on which the model learns (the batch), policies could eliminate extrapolation error entirely for deterministic MDPs. This policy constraint approach to distributional shift mitigation is simple in theory, but in practice such a policy will not be able to improve the behaviour policy (the policy used for data collection) without deviating from it [8].

2.3 Recent advances in Offline RL

The focus of research in Offline RL over the recent years has mostly been around the inherent issue of distributional shift, as discovering new and improved ways of dealing with it would open up more possibilities for applying Offline RL in many domains. Methods which deviate from Fujimoto et al.'s [3] simple idea of Batch-Constrained RL - in the sense that they allow for improvement over the behaviour policy - are being published frequently.

3 Addressing distributional shift

There are two main ways in which this can be done, either by constraining the distribution which a policy will be evaluated on to be close to the distribution which it has been trained on, or by estimating the epistemic uncertainty as a measure of distributional shift [8]. In a mathematical context such as the one presented in this paper, epistemic uncertainty refers to the uncertainty of which probability distribution is being dealt with [18]. The following sections (3.1, 3.2) will cover in more detail how the mentioned methods do indeed mitigate distributional shift, as well as review some state of the art methods.

3.1 Policy constraint and policy penalty methods

Two potentially useful methods for addressing distributional shift are policy constraints and penalty methods. Recently, new solutions have been proposed to address distributional shift by utilising policy constraint and penalty methods. Some of which are the focus of this section.

Policy constraint methods

Policy constraint methods aim to ensure that the evaluation policy distributions are close to the behaviour policy distributions. Achieving this would reduce the amount of out of distribution actions when computing the expected value of the Q-function. The Q-function refers to the function which assigns a value to a state-action pair.

Various policy constraint methods differ mostly by the probability metric they use to evaluate how close the distributions are [8].

Such methods can be described formally with a constrained objective function to be maximised (Eq. 2), once the aforementioned probability

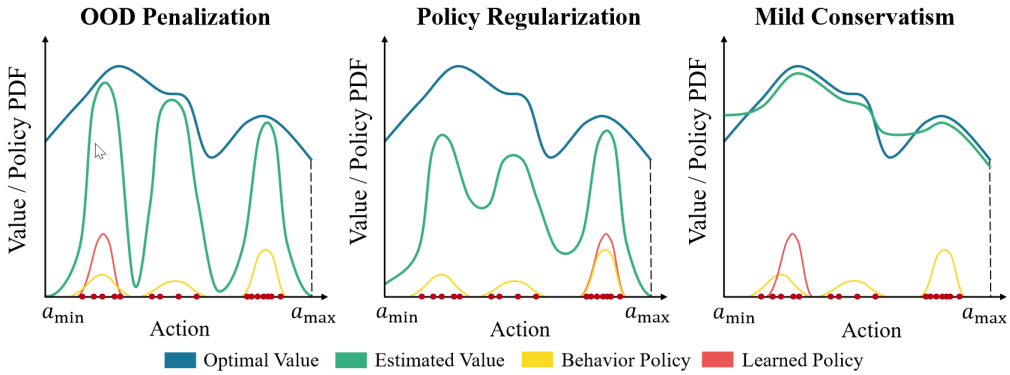


Figure 3. Figure from Lyu et al.'s paper on MCQL [9]. OOD Penalisation is clearly conservative as the estimated value dips very low when OOD. Policy Regularisation over-generalises as the estimated value is much below the optimal value. Mild Conservatism makes the most out of the conservatism and generalisation balance.

metric $P(\cdot, \cdot)$ has been defined [12],

$$J(\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}, \mathbf{a} \sim \pi_\theta(\cdot|\mathbf{s})}[Q^\pi(\mathbf{s}, \mathbf{a})] \tag{2}$$

such that the probability distributions are close enough with respect to $P(\cdot, \cdot)$.

$$P(\pi_\theta, \hat{\pi_\beta}) \leq \epsilon \tag{3}$$

Where $\hat{\pi_\beta}$ is an estimate of the behaviour policy π_β which is not necessarily known [8] and could be extremely hard to estimate due to its complex nature (data could be collected by humans, it could be made of multiple different policies, etc.). Several methods have been proposed to estimate this behaviour policy, such as training a parametric model with behaviour cloning [19], or using conditional variational autoencoders [6].

Policy penalty methods

Penalty and policy constraint methods differ because of their intentions. With penalty methods, the intent is not only to reduce the deviation from the behaviour policy at each state, but also to avoid actions who will lead to additional deviation from this policy at future time steps [8]. This can be done by adding a penalty term to the reward function, which is convenient when using constraints such as the KL-divergence constraint or other f-divergences [11, 8].

Generalisation and conservatism

Policy constraint and penalty methods mainly suffer from one thing, which is to find the adequate balance between generalisation and conservatism. There is a natural trade off between the two, as increasing generalisa-

tion is done by avoiding querying out of distribution samples, and conservatism is exacerbated by penalising out of distribution actions. Figure 3 from Lyu et al.'s recent paper on MCQL [9] illustrates this concept and proposes an innovative method to address this balance of conservatism and generalisation.

Mildly Conservative Q-Learning (MCQL) is designed such that it "induces a policy that behaves at least as well as the behavior policy and no erroneous overestimation will occur for OOD (out of distribution) actions" [9]. This model is arguably among one of the most advanced recent models in Offline RL due to its state-of-the-art performance in the D4RL benchmarks [2].

3.2 Uncertainty estimation

Uncertainty estimation to deal with distributional shift basically functions as such, "if we can estimate the epistemic uncertainty of the Q-function, we expect this uncertainty to be substantially larger for out-of-distribution actions, and can therefore utilise these uncertainty estimates to produce conservative target values in such cases" [8].

In recent years, many methods using uncertainty estimation have been proposed such as MOReL [5], MOPO [22], EDAC [1] and UWAC [20], all of which achieved state of the art performance on industry standard benchmarks when published.

Among these methods, UWAC has shown great promise by outperforming previous state of the arts (MOPO, MOReL, BEAR, etc.) [20] with its Monte Carlo dropout for uncertainty estimation. Additionally, Wu et al.'s Uncertainty Weighted Actor-Critic paper uncovered the fact that enforcing trajectory-wise consistency is incompatible with Offline RL, as the dataset on which the policy is trained does not have to contain full trajectories.

4 Conclusion

This paper has concisely introduced the field of (Offline) RL and has reviewed recent promising methods aiming to mitigate distributional shift in Offline RL, both by policy constraint and by uncertainty estimation. Seeing as most research has focused on how to limit distributional shift during offline training, switching directions and exploring how to recover

from out of distribution states as Reichlin et al. have begun doing [13] could be of substantial value to the field, as this area of RL has not been researched extensively yet.

References

[1] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7436–7447. Curran Associates, Inc., 2021.

[2] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[3] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration, 2018.

[4] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature Medicine*, 25(1):16–18, Jan 2019.

[5] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2020.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[7] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning, 2021.

[8] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[9] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning, 2022.

[10] Xinkun Nie, Emma Brunskill, and Stefan Wager. Learning when-to-treat policies, 2019.

[11] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization, 2016.

[12] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems, 2022.

[13] Alfredo Reichlin, Giovanni Luca Marchetti, Hang Yin, Ali Ghadirzadeh, and Danica Kragic. Back to the manifold: Recovering from out-of-distribution states, 2022.

[14] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning?, 2022.

[15] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Mach. Learn.*, 84(1-2):109–136, July 2011.

[16] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 907–917. PMLR, 08–11 Nov 2022.

[17] H. H. Tseng, Y. Luo, S. Cui, J. T. Chien, R. K. Ten Haken, and I. E. Naqa. Deep reinforcement learning for automated radiation adaptation in lung cancer. *Med Phys*, 44(12):6690–6705, Dec 2017.

[18] Wikipedia. Uncertainty quantification — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Uncertainty_quantification, 2022. [Online; accessed 20-November-2022].

[19] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019.

[20] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning, 2021.

[21] Teng Xiao and Donglin Wang. A general offline reinforcement learning framework for interactive recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4512–4520, May 2021.

[22] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization, 2020.

Cryptocurrency price analysis using old and new effective factors

Giancarlo Andriano
giancarlo.andriano@aalto.fi

Tutor: Hoseini Seied

Abstract

Since the invention of Bitcoin in 2008, the scientific community has tried to model the behaviour of cryptocurrencies price by studying relevant factors and their effects on the markets. The identification of the most important effective factors on price has been a dynamic process, continuously updated through collected historical data analysis and new discoveries. This paper aims to summarize and describe the most common factors shared by all cryptocurrency markets, as well as to introduce new others not considered by previous research, such as energy prices, stock market indexes and inflation.

KEYWORDS: Cryptocurrency Price, Effective Factors, System Dynamic Model

1 Introduction

Starting in 2008 with the invention of Bitcoin, cryptocurrencies have gained widespread interest from different communities ranging from private investors to financial institutions and regulators. They represent digital financial assets, whose ownership and management is supported by distributed cryptographic protocols called blockchains. Such technologies

build on top of a peer-to-peer overlay network architecture to enable untrusted participants to exchange tangible/intangible assets or a unit of value that represents the currency itself. Transactions or digital events taking place can be collectively verified by a majority agreement of the participating community members, who then save those records in a chronological order.

Although blockchains provide cryptocurrencies with significantly better results in terms of traceability, transparency and decentralization than traditional currencies , high price volatility bears many implications and concerns for investors and regulators as well [1]. Small investors willing to pursue an investment in crypto assets mostly follow either a buy-low sell-high strategy, realizing returns during bull market periods, or through "mining" specific currencies [2]. However, cryptoexchanges now make the investment process easier, and the market capitalization of the two main coins (i.e., BTC and ETH) was above $500 billion in September 2022. These factors attract large investment companies, mutual funds, edge funds and private investors who have increasing interest in modelling behaviours of such new technologies.

The research literature has therefore focused on defining price prediction models and effective factors, i.e, factors usually related to such price changes. For digital assets, high volatility is mainly responsible for uncertainty regarding future expected returns [3], making it difficult to define predictive models. Furthermore, by considering the investor's risk aversion, portfolio management policies need to determine whether and in what measure invested capital should be allocated to them.

The purpose of this paper is to clarify some key factors involved in cryptocurrency price change by using System Dynamic Modelling on Vensim®, a software that provides user-friendly graphical modeling interface. This work also aims to help constructing more accurate models for crypto price prediction, by highlighting the properties of price changes, as well as to suggest new factors to consider in future research.

The structure of this paper is the following: the next chapter summarizes some common effective factors along with their description and composition. Chapter 3 provides then a graphical representation of a model using these effective factors, by means of System Dynamic Modelling on Vensim® software. Chapter 4 analyses past research literature and explains the reason why some new factors could be included in the model to

improve its accuracy. It the end, chapter 5 draws some conclusions and future research challenges.

2 Common Factors

This chapter analyses some common factors usually related to price changes in cryptocurrencies. In order to understand the internal dynamics shared by all blockchain technologies, supply and demand are presented first, together with competition, which provides inter-blockchain relationship analysis . Then, Investor Sentiment and policy-related components follow, providing an overview of the external forces shaping market price.

2.1 Supply and Demand

Contrary to many other products or commodities, the total supply of cryptocurrencies is usually fixed and directly specified in the protocol of each specific coin [4]. In this way, price dynamics do not exhibit cyclical fluctuations as per common livestock markets, and more reliable model parameters that accurately track price changes are easier to define. Moreover, the upper limit on supply together with a fixed hashrate increases the "mining" marginal cost (i.e., the cost of "mining" one more unit of cryptocurrency) and reduces revenue streams. On the other hand, demand is influenced by other factors, including transaction costs (i.e, fee to pay when exchanging currency between wallets) and forks (i.e, when a blockchain splits into two separate branches) [5].
In addition, lottery-like behaviour has been found in many cryptocoins market [6], suggesting that speculation may also be an important factor affecting demand.

2.2 Competing Cryptocurrencies

Cryptocurrencies attractiveness also depends on competition. On one side, newborn coins usually introduce new features to solve problems of earlier blockchain architectures and provide better quality in their implementations. On the other, older cryptocurrencies, which have been longer on the market, are more familiar to investors and benefit from time advantage: a key factor supporting their reliability expectations [7]. Since both these aspects affect market price, it is difficult to establish a general rule to retrieve either a positive or negative effect, even though competi-

tion created by newer succeeding cryptos usually redistributes capital allocation on the market, therefore generating negative effects on the older ones.

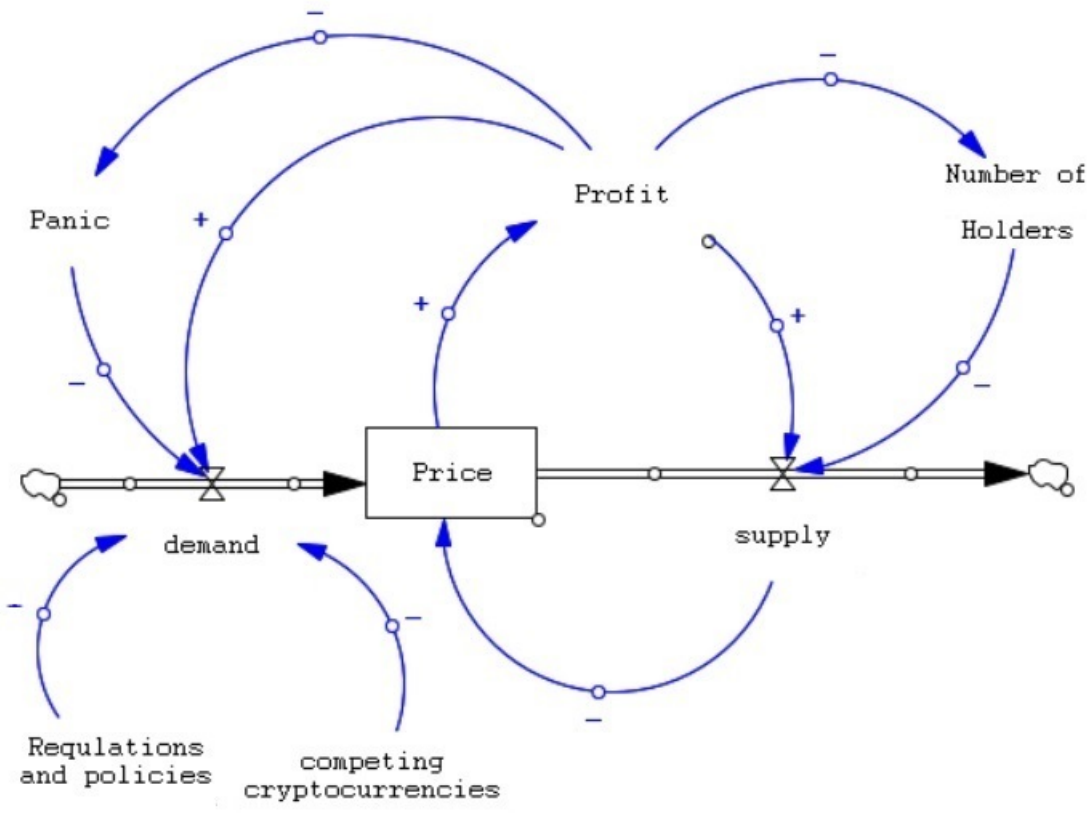
2.3 Investor Sentiment

Investor Sentiment probably represents the most difficult factor to analyze, because of its many different factors involved. Although the number of holders for each specific coin is directly measurable, their sentiment is related to different effects, including profit and panic. The former is generated when the coin is traded in the cryptomarket and exchanged for real money. The latter refers to the phenomena of large-scale selling of the digital assets, causing a sharp decline in prices. It has been widely analyzed by current research which usually relates it to social media coverage. In particular, English tweets on the online social media Twitter [8] and users replies/comments posted on relevant online cryptocommunities [9] are effective price predictors for most cryptocurrencies, when enough data is available for collection.

2.4 Policy and Regulation

Attempts or implementations of regulatory measures on blockchain technologies by governments and financial institutions has been demonstrated to impact cryptocurrencies prices [10]. In particular, tightening monetary policies, such as anti-money laundering, exchange or issuance regulation, have "economically significant negative effect on the market" [10], whereas relaxation of policy measures brings appreciation in value. This explains why considering cryptocurrencies as a financial asset is still not a straightforward topic, especially when regulation generates depreciation and market forces are towards unregulation .

3 System Dynamic Model



The image above shows a graphical representation of the System Dynamic Model by means of Vensim®. The representation is similar to a Stock and Flow diagram, where the Price stock is determined by its inflows and outflows. In this case, according to the classical definition [11], market price is directly coupled with demand and inversely with supply, which correspond to respectively an inflow and an outflow. Each one of these two flows is correlated to different factors, either negatively (-) or positively (+), and such correlations are represented by signed arrow connections. Moreover, as mentioned in Chapter 2.3, investor sentiment is hereby deconstructed into its three main factors, namely panic, profit and number of holders or investors.

Considering all the effective factors, Demand is negatively correlated with almost all of them, excluding profit. Regulation and policies, together with new competition in the market, contribute to decrease the demand of the existing cryptocurrencies, especially in case of investor panic. Profit, instead, is the only positively coupled factor, hence incrementing demand when profit opportunities rise for investors.

On the supply side of the model, the discussion is mainly confined to the

profit, which has both direct and indirect effects on supply. The direct effect is positively correlated, since high profit opportunities imply that many investors are selling their assets and generating revenue streams, therefore increasing cryptocoins availability on the market. The indirect effect, instead, has negative correlation through the factor "number of holders". Indeed, low supply phenomena are caused by the increased number of holders who buy the digital assets following their increased future profit expectations.

4 Literature overview

The latest research in cryptocurrencies has highlighted interesting considerations not analyzed by earlier studies. During the last three years, starting from 2020 with the Covid-19 pandemic, until the higher inflation period characterizing the year 2022, these events have shaped at different levels and in different ways cryptocurrencies behaviours on financial markets. Unprecedented events have led to alternating periods of prosperity and distress, thus giving stage to abnormal events which could not have been observed and studied otherwise.

The first consideration to be mentioned is the alleged hedging role that cryptos should play during economic turndowns [12]. Hedge assets are normally used by investors to protect against uncertainty or restrictions in traditional assets markets, but this was not the case for many coins, including Bitcoin [13].

Secondly, cryptos are increasingly integrating themselves into the global financial market, and becoming subject to the same macroeconomic factors influencing stock markets and market indexes [14]. This trend was surprisingly not true during the pre-Covid era, when cryptocurrencies were unrelated to traditional commodities and assets [3]. Furthermore, the recent increase in global energy prices is also proving to be an important factor leading such new transformations.

5 New factors

The idea of relating cryptocurrencies models to normal stock markets is already emerging in latest research [15]. These changes are mainly brought either by changes in the world global economy as well as by the

development of new blockhain technologies, and the private financial sector is already offering to clients crypto-backed securities as financial instruments. Stablecoins represent an example: they are pegged digital currencies linked to a reference asset, which may be cryptocurrencies, fiat money or exchange-traded commodities [16]. Therefore, gold-pegged or USD-pegged stablecoins, which have already existed in the market for a few years at the moment of writing, represent a direct connection between the two markets. In addition, new hedging methods against volatility are implemented in form of derivatives, such as crypto options and futures [17], hence raising many concerns about the urgency for regulation of such digital assets [18].
For these reasons, this chapter suggests new factors that will be important for future research to elaborate more accurate systems and models for cryptocurrencies price analysis, since, at the moment of writing, no substantial work has been done in this direction.

5.1 Energy prices

Energy consumption cost figures as the main source of expense for cryptocurrency miners [19] who should buy, build and maintain an infrastructure of highly efficient and costly hardware to perform hard calculations. Together with carbon cost (i.e., the cost of produced CO2 dismantlement from such infrastructres), it drives the profitability of the entire business, especially during times of globally high energy prices, as for year 2022. A more accurate model for cryptocoins price could implement an analysis of energy prices using publicly available data on energy indexes, such as the European Union Consumer Price Index (EUCPI) or the Global Price of Energy index (PNRGINDEXM). Considering that high energy prices have not been common after the invention of first cryptocurrency Bitcoin in 2008, evaluating or builing a new model based on previous historical data is a difficult process, but research could head in this direction and use year 2022 data for the purpose.

5.2 Stock market indexes

As mentioned at the beginning of chapter 5, as a consequence of the increasing number of investors and the rise of new financial instruments in the business, the cryptomarket is becoming much more correlated and tied to the stock market than it was before [15][13]. In fact, the assump-

tion that this new market is unaffected by global financial cycles is no longer accurate [14], and better models should be further investigated. For example, global financial flows and effects could be included in forms of technological stock markets indexes, such as S&P500 or NDXT, influencing directly or indirectly cryptocurrencies price.

5.3 Inflation

A last important factor which has been influencing cryptos price during the whole year 2022 is inflation [14]. Altough the effect of inflation is not relegated to the cryptomarket alone, it represents an important macroeconomic factor influencing stock markets and energy prices as well. Therefore, classical approaches to estimate inflation, such as future interest rates expectations or energy futures/options price, could also be used to improve cryptomarkets price model accuracy by future research.

6 Conclusion

Overall, although other effective factors could have been considered in the discussion, the purpose of this paper was to provide a simple model for cryptocurrency price analysis on the market, as well as to introduce new potentially interesting ones for future research. The latest developments in these blockchain technologies and their implementation into regular financial instruments are signs of a new changing condition, where claiming isolation between normal stock markets and the crypto counterpart is no longer accurate, and therefore new reliable models should be able to pictures such new changes.

References

[1] Giudici Giancarlo, Milne Alistair, and Vinogradov Dmitri. Cryptocurrencies: market analysis and perspectives. *Journal of Industrial and Business Economics*, January 2020.

[2] J. O'Dwyer Karl and Malone David. Bitcoin Mining and its Energy Footprint. *IET*, September 2014.

[3] Liu Yukun and Tsyvinski Aleh. Risks and Returns of Cryptocurrency, August 2018.

[4] Gandal Neil and Halaburda Hanna. The microeconomics of cryptocurrencies. *Journal of Economic Literature*, June 2016.

[5] Sovbetov Yhlas. Factors Influencing Cryptocurrency Prices: Evidence from Bitcoin, Ethereum, Dash, Litcoin, and Monero, February 2018.

[6] Grobys Klaus and Junttil Juha. Speculation and lottery-like demand in cryptocurrency markets. *Journal of International Financial Markets, Institutions & Money*, March 2021.

[7] Liu Yukun and Tsyvinski Aleh. Can we predict the winner in a market with network effects? competition in cryptocurrency market, August 2018.

[8] Valencia Franco, Gómez-Espinosa Alfonso, and Valdés-Aguirre Benjamín. Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. *MDPI*, June 2019.

[9] Bin Kim Young, Gi Kim Jun, Kim Wook, Ho Im Jae, Hyeong Kim Tae, Jin Kang Shin, and Hun Kim Chang. Predicting fluctuations in cryptocurrency transactions based on user comments and replies. *PLOS ONE*, August 2016.

[10] Shanaev Savva, Sharma Satish, Ghimire Binam, and Shuraeva Arina. Taming the blockchain beast? regulatory implications for the cryptocurrency market. *Research in International Business and Finance*, January 2020.

[11] Supply and demand. *https://en.wikipedia.org/wiki/Supply_and_demand Wikipedia*.

[12] Demir Ender, Huseyin Bilgin Mehmet, Karabulut Gokhan, and Cansin Doker Asli. The relationship between cryptocurrencies and covid-19 pandemic. *Eurasian Economic Review*, March 2021.

[13] Conlon Thomas, Corbet Shaen, and McGee Richard J. Are cryptocurrencies a safe haven for equity markets? an international perspective from the covid-19 pandemic. *Research in International Business and Finance*, December 2020.

[14] Le TN-Lan, Aikins Abakah Emmanuel Joel, and Tiwari Aviral Kumar. Time and frequency domain connectedness and spill-over among fintech, green bonds and cryptocurrencies in the age of the fourth industrial revolution. *Technological Forecasting and Social Change*, January 2021.

[15] Sharma Rakesh, Stapleton Chip, and Kirsten Rohrs Schmitt. Is there a cryptocurrency price correlation to the stock market?, October 2022.

[16] Stablecoin. *Wikipedia: https://en.wikipedia.org/wiki/Stablecoin*.

[17] Nekhili Ramzi and Sultan Jahangir. Hedging bitcoin with conventional assets. *Borsa Istanbul Review*, July 2022.

[18] Treasury official says the need for stablecoin legislation is 'urgent'. *www.bloomberg.com*.

[19] Krause Max J. and Tolaymat Thabet. Quantification of energy and carbon costs for mining cryptocurrencies. *Nature Sustainability*, November 2018.

An interesting use case of model checking: Storm surge barrier at Rotterdam, the Netherlands

Hung Nguyen
hung.g.nguyen@aalto.fi

Tutor: Chris Brzuska

Abstract

This seminar paper focuses on the useful application of model checking on the Maeslantkering storm surge barrier system, especially targeting the crucial component of Beslis en Ondersteunend - BOS, which is the safety-critical mechanism ensuring that the barrier is closed or open in time. The investigation mainly targets the study of a Radboud University Nijmegen team, who conducted a formal verification on the determine excessive water level (DEW) component of BOS and, eventually, detected mismatches between the component's specification and code implementation. Also, there were deeper errors that occurred during the model checking process. As a result, the example clarifies the usefulness the idea of using formal verification in system development as its usage can help find potential flaws in the early stage of development.

KEYWORDS: model-checking, formal verification, storm surge barrier

1 Introduction

Formal verification is a practice of verifying the consistency between a system design and its formal specifications, or properties, by utilizing mathematics or formal methods. For the last twenty years, progress in

formal methods has enabled many promising verification techniques that can detect implementation flaws in complex systems. Furthermore, with the help of potential and powerful software instruments, these verification techniques are even more useful when the possibility for automation is even clearer. Catastrophic cases such as Ariane-5 missile [6], Therac-25 radiation therapy [9], etc., would have been prevented if there had been better formal verification at that time [2]. Among the exisiting verification techniques, model-checking emerges as a reliable tool for automatically verifying the specifications, pointing out several subtle defects and unexpected behaviors in the early stage of system design, thus, acting as a safety net against disastrous events [4].

For a long time, coastal areas have been a target of natural hazards such as land erosion, seawater intrusion, hurricanes, floods, etc. Moreover, climate change also adds to the severity of these disasters one key effect: sea level rise [12]. Therefore, there is a critical request to have a better solution to increase safety in coastal zones. One interesting solution to this problem is the Maeslantkering storm surge barrier in Rotterdam, the Netherlands. However, when it comes to a sophisticated and complex system, which is responsible for the safety of a city of more than half a million of people, the correctness and reliability of that system needs to be thoroughly checked. With the help of model-checking verification, many defects and inconsistencies have been found, thus, increasing the robustness and trustworthiness of the system [8] [11] [15]. In the end, the paper points out the importance of using formal verification during the early stage of every system development, which can help detecting possible bugs.

2 Model checking

2.1 Software and hardware formal verification

Because of the growing complexity of software and hardware, a large percentage of time and effort are spent on verifying the correctness of the system. To reduce the verification time as well as ease the stressful effort of the checking process, formal verification is introduced as a method to evaluate the correctness of a program by applying mathematics in modeling [2]. After a long period of development and research, formal ver-

ification has come up with many approaches to increase automation in verification and detect early-stage defects, such as deductive verification, program derivation, or model checking, which is the main approach for verifying the use case of storm surge barrier at Rotterdam.

2.2 Model checking

A finite state model is a model that can only contain one of a finite number of states at any given time. Model checking (also known as property checking) is an approach to clarify if a finite state model of a system respects, or matches the system specifications. Usually, the specifications of the system are defined in temporal logic, which can be simply known as logic of time, and the model of the system is determined as a state-transition diagram. Despite its reliability and practicality in formal verification history, model checking is perceived to be less popular than software validation techniques such as integration testing, system testing, etc.

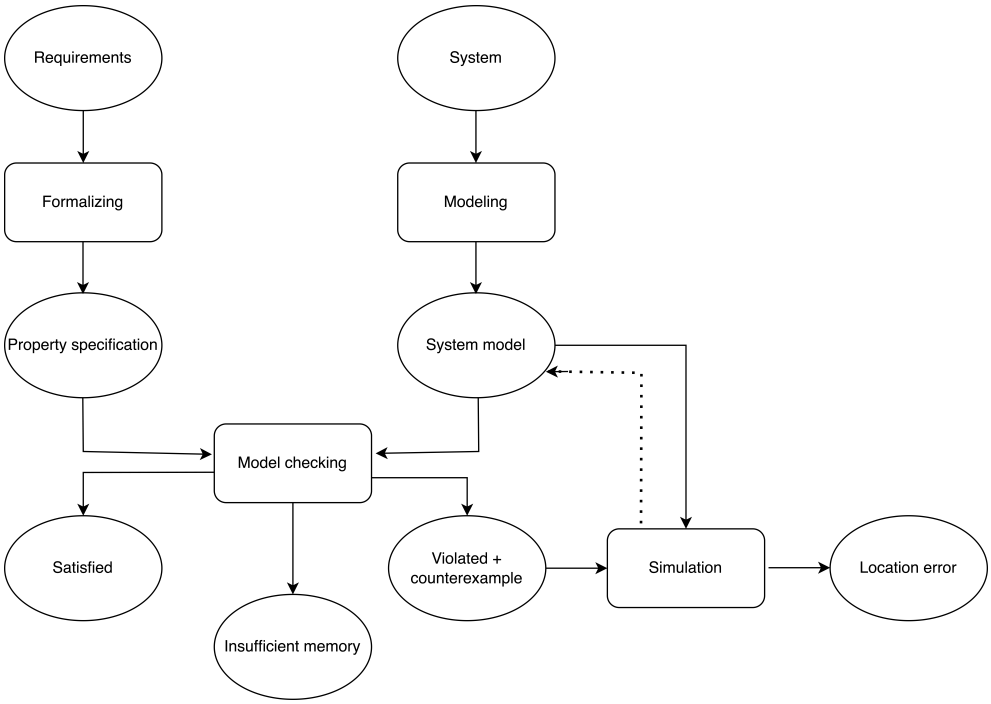


Figure 1. Overview of model checking technique.

Figure 1 above depicts an overview of how model-checking approach is performed. Initially, requirements need to be formalized to determine an unambiguous specification. After that, the system is carefully modeled as a system model then both property specification and system model are fed into the model checker for verification. There would be three possible outcomes after the model-checking phase. First, a satisfying result means

the system model accepts the property specification. However, this does not mean the system can satisfy the property, because the input of the system model when doing the modeling phase can be bad, thus resulting in a bad system model – also known as the garbage in, garbage out concept - a poor quality data input will produce a poor quality data output. Therefore, there is an assumption that the model-based verification technique is only as good as the model of the system [2].

Second, the result could be when the property we are checking is violated, then the model-checking technique will produce useful and diagnostic information as a counterexample for debugging. With the support of a simulator, the system can be rewound to the violating case and then, the property or model can be fixed accordingly.

Third, a possible outcome would be when there is insufficient memory, which means the model we are checking is too big to be handled, for example, the state explosion problem [5]. In this scenario, it is recommended to reduce the size of the model and repeat the entire procedure again.

Model checking has been widely applied to multiple fields where it can not only reduce the market release time but also act as a guarantee for avoiding catastrophic failures [10], provided that the modeling process is correctly carried out. For example, the Therac-25 Radiation overdose case which made at least 6 cases of overdoses and resulted in three cancer patients dying, or the Ariane 5 missile crash in 1996 due to a data conversion flaw are clear examples of how defects in software and hardware can produce such disasters. By using model-based verification technique, many design flaws in the control system of the storm surge barrier at Maeslantkering have been revealed, therefore, reduced the likelihood of these disasters from happening.

3 Storm surge barrier control system

3.1 Maeslantkering storm surge barrier

The Netherlands have long been famous for their surface below sea level. Almost one-third of its area has this low elevation characteristic. As a result, this country has encountered many flooding disasters. One example is the North Sea flood in 1953, which caused the death of 1836 people and widespread destruction [1]. Therefore, there are multiple solutions

to mitigate these natural disasters such as storm surge barriers, closure dams, or tidal barrages, but the first one is considered to be a better solution towards tidal intervention for big cities [13] because of its mobility and advanced technology.

Figure 2. Maeslantkering - Storm surge barrier at Rotterdam, the Netherlands

Figure 2 above showcases the impression of the Maeslantkering storm surge barrier in Rotterdam, one of the biggest harbor cities in the world. It has two hollow floating arms (or walls) which are controlled to close and open when there is a threat of flooding. This operation is closely related to the safety of the whole city, and consists of tasks and maneuvers with complexity, therefore, it is safer to let computer systems take control of the barrier operation [3].

3.2 The BOS system - Beslis en Ondersteunend

BOS is a control system for making decision on when to close or open the barrier developed by RWS - a division of the Dutch Ministry of Transport, Public Works and Water Management. When the expected water level from the sea is so high that it could cause flood in the city, there will be an automated mechanism to close the barrier. However, Rotterdam is a crowded port, so every hour of closed entry would mean tremendous economic loss, so it is critical that the system only closes the barrier when actually needed.

3.3 Radboud University Nijmegen study - a useful case of model-checking

Z notation is a formal specification language which uses simple mathematics for describing computer systems [14]. Z has a type checker tool - ZTC, which is used for detecting syntax and typing errors of the Z specification [7]. Although the BOS system is complied with the use of formal methods with Z formal language associating with the ZTC type checking tool, and also some submodules of the system were also verified by using SPIN model checker - a tool for verifying the correctness of asynchronous system models, the BOS system can be even more rigorously checked with better approaches from formal verification.

Therefore, three years after the first time BOS automatically closed the barrier in 2007, a research group (Ken Madlener, Sjaak Smetsers, and Marko van Eekelen) from Radboud University Nijmegen, the Netherlands, conducted a project to increase confidence in the safety of the software under the request from NRG (the Nuclear Research and consultancy Group) and RWS. An important module - DEW - of BOS system with 800 lines of C++ code was selected and verified by using model checking technique. When developing the project, the authors manually sketched a Z specification of the component and a lightweight model from the C++ codebase. Then the model is used to prove its consistency with the specification.

3.3.1 The selected component – DEW

Closing barrier procedure of the BOS system is decided by the excessive water level predictions calculated by meteorological and hydrological information. When these levels are too high and can cause damage to the inner land area, the BOS system will send commands to the barrier system to start the closing procedure, and related departments will also be informed about the incident. During the operation, BOS will execute a code script that constantly performs calls to native functions. These functions are in charge of the process of sending a command, requesting for status, or making final decisions, whether to close or open the barrier. Determine excessive water level (DEW) is one of the components that are responsible for the crucial part of making decisions. The water level predictions are computed based on locations in the cities: Rotterdam, Dordrecht, and Spijkenisse, and are forecasted in one day advance with 10 minutes intervals. Predictions are saved in the database and queried by

DEW at a certain time through the hydraulic-model evaluator to compare with the maximum water level among the three locations. If the water level extends over the prediction, DEW will make it a flag to determine that there is an excess. Besides, the excess must be classified as critical to be considered for closing the barrier, and it is defined when one of the maximum water levels is surpassed, and within 20 minutes since the level is surpassed, the predicted water level is still as that high or more.

3.3.2 The Z specification

In the project, the authors formalized the requirements of DEW component into Z specification. Z specification can be seen as a literate specification whereas the Z specification source code contains constructions such as schemas and constraints [15]. The main schema of the component specification is defined below:

$$\begin{aligned} DEW == \quad & SetEvaluationParams \gg DetModelEvaluation \gg \\ & (EvaluationFailed \vee CoreDEW) \end{aligned} \tag{1}$$

In the main schema, the parameters such as the location's maximum water level or the forecast evaluation interval are formalized correctly through SetEvaluationParams. Then, the output is passed on to DetModelEvaluation schema. After that, if DetModelEvaluation produces an unsuccessful result, EvaluationFailed schema is selected, and if successful, the CoreDEW schema is chosen instead [11].

During checking the origin formal Z specification of the DEW component, the authors found two inconsistent points between the Z specification and the C++ codes:

- First, the specification does not take into consideration the definition of evaluation interval but on the other hand, specified in the code implementation. This could demonstrate a possibility of unawareness of the implementer, and the specification was not updated correspondingly. - Second, the prediction database is specified to be able to store many predictions for each location (or run-id), but on the implementation side, there is only one prediction for every location/run-id. This is another mismatch between the specification and code implementation.

3.3.3 Modeling the DEW component and verification

This section explains how the project team crafted the model for the DEW component and applied this model to verify the formalized specification. The idea of the modeling process is to convert the current C++ of into a

lightweight PVS model. To add more clarification, PVS is a specification language that is associated with an autonomous theorem prover as well as many supporting tools such as checkers for types, and built-in theory. Consequently, many applications have applied PVS to provide formal verification support for their system properties. Back to the project, due to the non-existence of object orientation or pointer arithmetic, it is safe to structure a simple PVS model from the C++ codebase.

During the conversion from C++ to PVS, datatypes are modeled as closed as the genuine code with lifted types are prefixed with L as a naming rule. For example:

```
LInt : type = LType  WITH [# iVal : int #]
```

For functions, the C++ functions are translated directly into PVS functions with the same naming convention and input parameters. For example:

```
static flag CoreDEWC (
          const LInt [] cLocList, // in
          const Interval& cInterval, // in
          flag& Excess, // out
          LTime& ExpTime // out
)
```

is converted into:

```
CoreDEWC_pvs (
          cLocList: [Loc → LInt],
          cInterval: Interval
) : [flag, flag, LTime]
```

After modeling the codebase, both specification and implementation (expressed in functions) are compared through SPIN model checker. Both functions return an array of value [flag, flag, LTime] (flag, Excess, ExpTime respectively) and after the verification, a flaw was found in the code when the last two elements in the arrays (Excess and ExpTime) are not put into effect. Therefore, the team fixed the code, re-run the model checking again, and there was no more error found, as well as the theorem, is verified perfectly. This could prove the usefulness of using model checking in detecting defects and verifying a crucial component of a system, DEW in this case.

3.4 Other related studies

Before the study at Radboud University, there had been multiple works that aimed to verify the correctness of the BOS system and its dependencies.

The communication between BOS and BESW (the barrier hardware operator) is defined through an interface, which is called BOS-BESW-Interface (BBI). In Ruys's project, from 1997-1998, model checking was used for proving whether the BBI was implemented precisely. The project owner modeled the BBI protocol in PROMELA, which is a process-modeling language, and then utilized SPIN model checker, especially with the SPIN simulator, to validate the model implementation. As a result, one critical timing flaw was discovered as well as several mismatches in the specification of BBI [15]. Thus, proving that formal approach actually has its own value when system design can expose many defects, vulnerabilities, and inconsistencies.

Before Ruys, there had been another project conducted by Pim Kars in 1996, who used the same technique with PROMELA and SPIN to validate the BOS-BESW interface. As a surprise, most of the errors were spotted during the analysis step, while a few of them were found in the verification phase. Therefore, the system design was also proved to be erroneous and had to be redesigned [8].

4 Conclusion

Defects in software or hardware system can be hazardous to human safety and cause fatal losses. And it would be very difficult to guarantee the correctness of a system by just human effort without the aid of computer checking. By using formal verification, many flaws in hardware and software can be exposed in the early stage, which increases confidence in system consistency and correctness. The storm surge barrier at Rotterdam, the Netherlands, is an interesting use case of model-based formal verification. Although it had been verified by many studies such as Kars' (1996) and Ruys' (1997), which already spotted multiple mismatches and errors in the system, there were still places for improvement as during the study of Radboud University team, when many inconsistencies between the system design and predefined specifications were found, and also even a flaw was discovered.

As a result, formal verification in general, or model checking in specific, can add great value to big projects . And since the checking process does not need human interaction, model checking can be fully automated in testing workflow in collaboration with other validation methods such as unit test, system tests, thus, reduces the testing team workload.

References

[1] Alexander Hall. The north sea flood of 1953, 2013. Accessed: 2022-10-24.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[3] Michel Chaudron, Jan Tretmans, and Klaas Wijbrans. Lessons from the application of formal methods to the design of a storm surge barrier control system. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 — Formal Methods*, pages 1511–1526, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[4] A. Cimatti, P. L. Pieraccini, R. Sebastiani, P. Traverso, and A. Villafiorita. Formal specification and validation of a vital communication protocol. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 — Formal Methods*, pages 1584–1604, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[5] Edmund M. Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. *Model Checking and the State Explosion Problem*, pages 1–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[6] Mark Dowson. The ariane 5 software failure. *ACM SIGSOFT Software Engineering Notes*, 22(2):84, 1997.

[7] Xiaoping Jia. Ztc: A type checker for z notation–user's guide. *DePaul University, Institute for Software Engineering, Department of Computer Science and Information Systems, Chicago, Illinois, USA, version*, 2, 1998.

[8] Pim Kars. The application of promela and spin in the bos project. In *The Spin Verification System*, 1996.

[9] N.G. Leveson and C.S. Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.

[10] John Lång. Three examples on the real-life applications of model checking, 2019.

[11] Ken Madlener, Sjaak Smetsers, and Marko van Eekelen. A formal verification study on the rotterdam storm surge barrier. In Jin Song Dong and Huibiao Zhu, editors, *Formal Methods and Software Engineering*, pages 287–302, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[12] Leslie Mooyaart and Sebastiaan N. Jonkman. Overview and design considerations of storm surge barriers. *Journal of Waterway Port Coastal and Ocean Engineering-asce*, 143:06017001, 2017.

[13] Leslie Mooyaart, S.N. Jonkman, Peter de Vries, Ad Toorn, and Mathijs Ledden. Storm surge barrier: Overview and design considerations. *Coastal Engineering Proceedings*, 1:45, 01 2014.

[14] V Ruhela. Z formal specification language-an overview. *International Journal of Engineering Research & Technology (IJERT)*, 1(6):14–27, 2012.

[15] Theo C. Ruys. Towards effective model checking. 2001.

Attacks on Passwords

Iiris Joutsi
iiris.joutsi@aalto.fi

Tutor: Sanna Suoranta

Abstract

Passwords are the most widely used digital authentication method [11]. Conklin et al. [1] note that authenticating subjects based on a personalised id along with a password was convenient when less digital systems were implemented and their user bases were smaller. Since then, people use more internet accounts, thus needing more credentials. The amount of passwords and password-based systems combined with the lacking security they provide makes passwords susceptible to different kinds of attacks.

The brute-force attack is the most time-consuming, but it is also bound to crack a short password. The dictionary attack is more efficient in terms of time but does not perform as well as the brute-force attack when cracking random passwords. The rainbow table attack is the most efficient of the three attacks in terms of time and success rate. It performs faster than the brute-force or the dictionary attack because the attack is based on a precomputed lookup table, thus no hashes are computed during the attack.

We find increasing the time used to calculate hashes is a good defence against brute-force and dictionary attacks. Time can be increased by using longer passwords, hash functions that take longer to compute or limitations to how often a password can be attempted.

KEYWORDS: password, hash function, brute-force attack, dictionary attack, rainbow tables, password spraying, password managers, salts

1 Introduction

Passwords are the most widely used digital authentication method [11]. Conklin et al. [1] note that authenticating subjects based on a personalised id along with a password was convenient when less digital systems were implemented and their user bases were smaller. The situation has changed since. Today, people use more internet accounts, thus needing more credentials. Remembering all of them becomes a chore, leading users to choose easy to guess passwords. The amount of passwords and password-based systems combined with the lacking security they provide makes passwords susceptible to different kinds of attacks. The attacks can range from sophisticated and complex attacks on networks to social engineering, in which the user is tricked to disclose their password.

In this paper, we present three different attacks on passwords. All the attacks target systems with password-based authentication, so no phishing or network attacks are discussed. The first attack is the brute-force attack. Alongside with the strengths and weaknesses of the brute-force attack we will briefly discuss password spraying. The second attack presented is the dictionary attack. The third and final attack is the rainbow table attack. The attacks will be discussed in terms of efficiency in time and attack target.

I also introduce common practices with passwords chosen by humans and machines as well as the concepts of hash functions along with password complexity to help investigate the attacks and their efficiency. After the attacks, we introduce time and randomness as ways for the user as well as the systems to complicate and prevent such attacks.

This paper is organised as follows. Section 2 presents both human and machine aspects of passwords as well as password complexity. Section 3 presents the different password attacks, while Section 4 discusses ways to prevent such attacks. Finally, concluding remarks are presented in Section 5.

2 Passwords Usage

This section discusses why and how passwords are used as well as some technical aspects concerning the implementation of password-based systems. Implementations will lead to defining hash functions and exploring their usage with passwords. Additionally, password complexity is dis-

cussed in terms of information entropy. Finally, the potential targets of attacks are discussed.

2.1 Passwords

According to NIST [10], the official definition of a *password* is "a string of characters (letters, numbers, and other symbols) used to authenticate an identity or to verify access authorization". Commonly, passwords are used by people to authenticate themselves to different digital systems. Despite passwords not being a very efficient or secure way of authentication, their usage is still very prevalent. This is because implementing systems with password authentication is easier and less expensive than authentication with biometrics or physical security tokens. Biometrics and physical tokens require certain hardware solutions so incorporating them would require the development of a large feature for a smaller amount of devices. For the rest of the users to use new authentication systems would require acquiring a new device. However, using a password works on all devices and only requires the system's ability to recognise what is typed.

As with most security related practices, passwords are not immune to the dilemma of security vs. usability. Many issues with passwords stem from the need for users to remember their passwords. This leads to users writing down their passwords to physical or digital notes, reusing their passwords in multiple different services and selecting simple passwords. All these practices can be exploited by password-cracking systems. For example, if an adversary manages to get a user's password to some service, they might also try to use the same password to gain unauthorised access in other services that the user has credentials to. This is called *credential stuffing* [9].

To make passwords easier to remember, people also tend to make their passwords more vulnerable by following some patterns. It is shown by Miessler and Haddix [3] that the most common passwords are lacking in complexity. Within the ten most used passwords, six are a string of consecutive numbers starting from "1". None of the ten most common passwords are combinations of letters, symbols, and numbers. According to Tatlı [14], the most common human-chosen passwords only consist of letters, the second most common pattern consists of only numbers and the third most common pattern is some letters followed by numbers. The above implies that many machine-chosen or default passwords consist of numbers. This significantly reduces the search space the adversary has

to operate in. Thus, a random password of a certain length is more secure than a password of the same length following a pattern.

2.2 Hash functions

Passwords should never be stored as plaintext. This is because in case of a leak, all sets of credentials could be used as is. Thus, passwords are stored as *hashes*, created with *hash functions*. According to Menezes et al. [8], a hash function h has at least the following two properties:

1. *Compression*. Given an input x of arbitrary length, h will output to $h(x)$ of fixed size.

2. *Computational simplicity.* A hash value $h(x)$ is easy to compute, given function h and input x.

Hash functions can be divided into two categories: *keyed hash functions* and *unkeyed hash functions*. The difference between the two is that while unkeyed hash functions only take one parameter, the message, keyed hash functions also take a secret key as parameter. [8]

In the context of password security, there are three more properties expected from hash functions. These properties are

1. *Pre-image resistance.* Given a hash function h and an output y, it is hard to find any x to which applies that $h(x) = y$.

2. *Second pre-image resistance.* Given a hash function h and an input x, it is hard to find an x' to which applies that $h(x) = h(x')$.

3. *Collision resistance.* Given a hash function h, it is hard to find any to inputs x and x' to which apply that $h(x) = h(x')$.

Some commonly used hash functions include Message Digest (MD) and Secure Hashing Algorithm (SHA) [2], [12].

2.3 Password complexity

Many password-based services have password policies that demand the user to choose a password fulfilling certain criteria. The password might

need to be of a certain length, include upper and lower case letters, numbers or symbols. Making a password more complex increases the time the adversary must use to crack a password, making complex passwords more secure than simple ones. One metric to measure password complexity is *information entropy*. Password entropy H is calculated as follows:

$$H = L\frac{\log 2}{\log N},$$

in which L is the length of the password and N is the number of possible symbols. When calculating entropy $\log$ can be in any base.

Although password entropy is a good starting point for assessing complexity, it is often lacking. As Ma et al. [7] show, a more intricate model called Password Quality Indicator (PQI) considers how similar the password is to a dictionary word as well as its effective length. Effective length is the standardised length of the password, and it considers the different character sets used to form the password. For example, a password with letters, symbols, and numbers would have a longer effective length compared to an equal sized string of only numbers.

2.4 Passwords as attack targets

Attacks on passwords have different success rates depending on their targets. That is, attacks perform differently if they attempt to find a certain user's password, any password, or all passwords. Attacking a certain password is necessary if the adversary requires an admin level authorisation in some service. Cracking any password is useful in cases when all users have some sensitive information saved on their accounts. For example, if every user of a service has a payment method saved, the adversary could crack any password to gain access to some payment method. Finally, attacks aimed to crack all passwords could be executed if the adversary wants to sell credentials on the dark web, for example.

3 Attacks

This section presents three different attacks on passwords. First, we will present the brute-force attack along with mentions of password spraying. The second attack is the dictionary attack and finally, the rainbow table attack will be presented. Attacks are presented with strengths and weaknesses as well as suitability for different targets.

3.1 Brute-force Attack

In a *brute-force attack*, the adversary tries all possible passwords in order to eventually guess the correct one. Because brute-force attacks are comprehensive and systematic, they can be efficient for short passwords and PINs. With enough time and resources, the brute-force attack is bound to crack a password eventually. However, brute-force becomes increasingly time-consuming with longer passwords. In fact, the average time needed for a successful brute-force attack grows exponentially as the length of the password grows linearly.

The brute-force attack is limited because of its time-consumption. It even has a physical restriction against really long passwords. For example, the attack is useless against a random 128-bit password as it takes too much energy. According to the Landauer principle [6], the theoretical lower limit for energy needed to erase a bit is $\ln 2 * kT$, in which T is the machine temperature expressed in kelvins and k is the Boltzmann constant. Assuming a computer would operate in around room temperature, just flipping through the 128-bits would take

$$(2^{128} - 1) * \ln 2 * 1.380649\frac{J}{K} * 298K \approx 9.7 * 10^{17}J.$$

This corresponds to around 0.17% of the worlds annual energy consumption.

There are a few aspects to consider still. This hypothetical scenario only considers flipping through the bits without any consideration on whether it is optimal or even required. Furthermore, the Landauer limit is theoretical and it is uncertain whether the energy level could be reached in a real life setting. In reality, a random 128-bit password would be close to unusable as it is practically impossible to remember.

Another form of the brute-force attack is *password spraying*, in which the adversary attempts a common or default password against all usernames. This way the adversary does not need to compute as many hashes. Additionally, the adversary is not limited by safety mechanisms set for accounts. For example, some services only allow a limited number of log in attempts. Password spraying is common because manufacturers choose easy passwords for initial setups. According to Knieriem et al. [5], common default passwords include "password", "admin", and "dba". Some applications had an empty string as password, meaning they were not protected at all.

3.2 Dictionary Attack

A *dictionary attack* is a guessing attack in which the adversary uses a list of words instead of all possible strings. Usually these lists comprise of common passwords but they can also include personalised information such as names or phone numbers. Other potential lists include celebrities, fictional characters, or films. [4]. Finally, lists can also include variations of the original words. One common way to derive variations is using leet. In leet, certain letters are replaced with a similar looking number. For example "Hello" could become "H3ll0".

If it is enough for the attacker to gain any credentials to the system, a dictionary attack can be highly efficient. Many users choose easy to remember passwords for systems and all of them can be considered points of failure for the system. Moreover, the search space is only a fraction of that of the brute-force attack. For example, there are a bit over a million words in the English language and even if there were ten variations for each word, the adversary would have to iterate through a bit over ten million potential passwords. However, there are $26^8 = 208827064576$ different eight letter combinations.

3.3 Rainbow table attack

A *rainbow table* is a precomputed lookup table from which plaintext passwords can be retrieved during an attack [4]. Compare this to an adversary trying to crack a password using the brute-force attack. Most of their time would go into computing the hashes. Now, what if they stored all the plaintext strings they had tried into a table alongside with their hash? The next time the adversary tried to crack password hashes, they only need to see if their rainbow table has a match with the list of password hashes. This lookup, even from a very large table, is significantly more efficient in terms of computational time than computing the hashes. So, rainbow tables remove the most time-consuming part of password cracking, making it even more efficient than a dictionary attack. This is not to say that an attack would be successful within an instant. Lookups can be incredibly time-consuming, but they are generally more efficient than calculating an equal amount of hashes.

The used hash function restricts the rainbow table attack. Each hash function requires its own table and it is difficult to tell from the hashes which hash function the system uses. However, some of the hash func-

tions are more common than others. For example, MD5 is commonly used although it has been proven to not be collision resistant [13].

4 Preventing attacks on passwords

This section provides two defence mechanisms against password attacks: time and randomness. Both mechanisms are presented from different parties of the authentication, such as the user and the system. Applications of both mechanisms are also mentioned.

4.1 Prolonging the attack

A simple way to make password cracking more difficult for the adversary is to increase the required computational time. There are multiple ways to do this. First, having the system use a hash function that is more time-consuming can massively slow down the attack. This can be done by using a hash function that natively uses more time. The computing of a hash can also be artificially slowed down. For example, the system under attack can be configured to wait for some time before accepting a login.

Another way to increase the time required for the attack is to limit the times an adversary can attempt different credentials. This is common in mobile phones. The SIM card PIN code can only be attempted a limited number of times until a more secure PUK is needed. Passcodes used to unlock smartphones do not have as extreme measures, but they can lock the user out for a shorter period of time. For example, an iPhone will allow five incorrect attempts. After the sixth incorrect attempt, the phone will be disabled for one minute. Seven incorrect attempts will result in the phone being disabled for 5 minutes and the disable time will increase fast after more wrong attempts. Finally, after ten incorrect attempts restoring the phone will require another authentication method.

As the brute-force attack was the most time-consuming, prolonging the attack time will be the most effective against those attacks. Online attacks in general are less successful when time-consumption is increased.

4.2 Randomness

Randomness is something a user can use to their advantage regardless of the system used. As discussed in Section 2.3, using random passwords will dramatically increase the search space the adversary must operate

in. This makes the password more difficult to crack as well as defends it from personalised attacks using family names or hobbies. Randomness comes with a downside, remembering. Remembering a single long, random password can be difficult but the task becomes near impossible when having to remember different passwords for all systems.

Password managers are storages for a user's own passwords. The passwords are accessed using one master password, meaning all passwords can be used when remembering one. Many password managers come with browser extensions making it easier for the user to log in. Selecting a correct set of credentials instead of typing them removes the risk of typos. Of course, using a password manager creates a single point of failure for the user. This is why it is important to use a very secure master password.

There is also a way for the system to add randomness: *salts*. Salts are random strings that are concatenated with the password to create a more random input for the hash function [4]. When using salts, the system stores the hash value as well as the salt. Because the salt is stored as plaintext, salting does not increase the password security if the adversary is attacking a certain user's password with the help of a leaked database table. They can simply copy the salt and use it while conducting a classic dictionary attack. However, if the goal of the adversary is to find any or all passwords, their workload is significantly increased. With such targets, salts do well when defending passwords against dictionary and rainbow table attacks.

5 Conclusion

In this paper, we studied attacks on passwords and ways to prevent these attacks. We presented the brute-force attack, the dictionary attack and the rainbow table attack and investigated their efficiency with different attack targets. Afterwards, we discussed ways to prevent such attacks.

The brute-force attack is systematic and bound to have results when attacking short enough passwords. The dictionary attack is less time-consuming but will not work against random passwords as it only targets passwords that are common or follow certain patterns. The rainbow table attack is more efficient than a simple brute-force attack because of the precomputed lookup table. It is also more comprehensive than the dictionary attack because random strings are also included.

Two ways to prevent or complicate such attacks were presented. These

mechanisms included time and randomness. The time an adversary uses to crack a password could be increased by the system by prolonging the time it takes to compute a certain hash or limiting the allowed attempts with time. In addition, the user can extend the required time by selecting a longer password. Another way to prevent these attacks was to add randomness. By avoiding simple, low entropy passwords, the users can improve their password security significantly. Using a password manager removes the usability issue of remembering long and complex passwords as well as password repetition.

References

[1] A. Conklin, G. Dietrich, and D. Walz. Password-based authentication: a system perspective. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 10 pp.–, 2004.

[2] Quynh Dang. Secure hash standard (shs). Technical report, NIST, 2012-03-06 2012.

[3] Daniel Miessler and Jason Haddix. Seclists, 2022. A collection of multiple types of lists used during security assessments. URL: `https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-10000.txt`. (Accessed 21.11.2022).

[4] Henk C. A. van Tilborg and Sushil Jajodia. *Encyclopedia of Cryptography and Security*. Springer New York, NY, 2nd edition, 2011.

[5] Brandon Knieriem, Xiaolu Zhang, Philip Levine, Frank Breitinger, and Ibrahim Baggili. An overview of the usage of default passwords. In Petr Matoušek and Martin Schmiedecker, editors, *Digital Forensics and Cyber Crime*, pages 195–203, Cham, 2018. Springer International Publishing.

[6] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[7] Wanli Ma, John Campbell, Dat Tran, and Dale Kleeman. Password entropy and password quality. In *2010 fourth international conference on network and system security*, pages 583–587. IEEE, 2010.

[8] Menezes, A.J., van Oorschot, P.C. and Vanstone, S.A. *Handbook of Applied Cryptography*. CRC Press, 1st edition, 1997.

[9] Neal Mueller. Credential stuffing. *OWASP*, 2022. URL: `https://owasp.org/www-community/attacks/Credential_stuffing`. (Accessed 21.11.2022).

[10] National Institute of Standards and Technology. Security requirements for cryptographic modules. FIPS 140-2, NIST, May 2001.

[11] Maria Papathanasaki, Leandros Maglaras, and Nick Ayres. Modern authentication methods: A comprehensive survey. *AI, Computer Science and Robotics Technology*, Jun 2022.

[12] Ronald L. Rivest. The MD5 Message-Digest Algorithm. Technical Report 1321, IETF, 1992.

[13] Marc Stevens. Fast collision attack on md5. Cryptology ePrint Archive, Paper 2006/104, 2006. URL: https://eprint.iacr.org/2006/104. (Accessed 21.11.2022).

[14] Emin Islam Tatlı. Cracking more password hashes with patterns. *IEEE Transactions on Information Forensics and Security*, 10(8):1656–1665, 2015.

User Modelling with Reinforcement Learning

Joel Rämö
joel.j.ramo@aalto.fi

Tutor: Alex Hämäläinen

Abstract

The need for fast and accurate user modelling methods is in high demand across various domains. A traditional method used for modelling human behaviour has been reinforcement learning (RL). This paper reviews the current status of RL methods for the task of user modelling. Based on recent literature, the benefits of RL in modelling human behaviour agents are the natural similarities between motivation-based decision making and the task of optimizing a reward function in reinforcement learning framework. The challenges in implementing RL in user modelling are often the lack of initial data to initialize the reinforcement learning algorithm. Three recent case studies show that reinforcement learning can be effectively used for the user modelling requirements of modern software systems and applications, but there are still problematic areas that the framework can not yet cover without future research.

KEYWORDS: reinforcement learning, user modelling, inverse reinforcement learning, offline reinforcement learning

1 Introduction

Systems that run applications of artificial intelligence (AI) are increasingly engaging with the vast majority of human population. Simultaneously, modern AI breakthroughs are still accomplished in applications where social co-operation is not required: modern AI applications have had immense success in applications such as 1-to-1 games, image recognition and self-driving pilot. These applications are attractive targets for AI applications, as plenty of data is available and generating more data is not a problem. However, as AI applications starts increasing their presence in the natural world, it creates threats and opportunities that require extending the collaborative capabilities of an AI. For example, a self-driving AI may excel at navigating across natural world, but it still has very limited possibilities for interaction with other agents in the environment, such as pedestrians or other cars [3].

A crucial part of social interactions is the ability to infer the internal state of other agents (*theory of mind*) in order to predict and understand their actions correctly [12]. Simulating the internal state of a human is known as cognitive modelling, which is already widely used in artificial applications such as expert systems, natural language processing and robotics. Some aspects of human nature are still difficult for a cognitive model to predict, such as the impact of fatigue, stress, distraction, or emotion in decision-making [4]. Cognitive modelling faces more difficulties as the individual differences need to be taken into consideration [6]. Such example could be user modelling, where variety of behavioural strategies in information processing as well as the approximation of individual interests and capabilities need to be embedded into the predictive model, often with very limited amount of useful data [13].

One particular challenge in the progress of human-AI cooperation is the lack of data that could be used to generate accurate user models. Collecting data of humans is costly and includes issues related to privacy that restrict the quality of data that can be used in the user model. For these reasons, most cognitive models are based on parameters drawn from existing research and offer very little adjustability to individual differences in users. De Peuter et al. [4] describe the current situation of AI-based design tools as such where useful predictions can only be inferred if there is either great amount of data available or excessive contribution from the user to train the model. De Peuter et al.[4] suggest that modern machine

learning methods and tools can be used to improve user modelling within the current context.

This paper is a literature survey on contemporary approaches to user modelling with RL. The paper focuses on finding an answer whether or not viable methods exist to the problem of inferring a useful user model without a need to collect excessive amounts of data of the user. In specific, the paper considers applications of RL methods to the task of user modelling. The paper first presents background to what makes RL interesting method for the use of user-modelling and which examples of modern RL methods are proposed for the task, and then analyses studies where such methods have been implemented for user modelling.

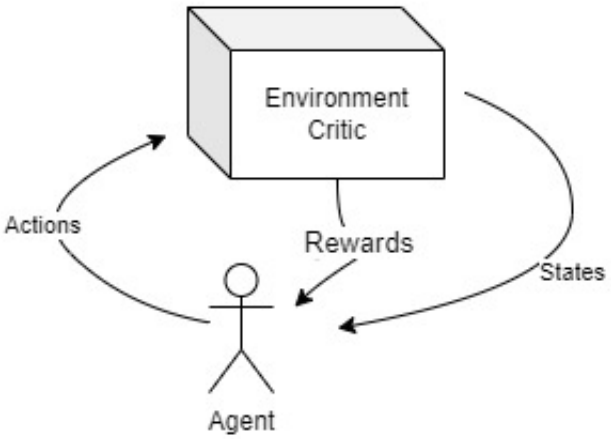
2 Background

When modelling a user without previous knowledge of their behaviour, it is necessary to design the methods used based on certain properties. Albrecht et al. [1] proposed that such properties include the level of determinism in the users' action choices, the model's behaviours variability, the availability of the knowledge used in the decision-making process, the independence of the decision-maker as well as different goals that agents may have. Some use-cases may also require modelling of the environment of the agent, which increases the demands of the method further. The environment sets requirements for the order of actions, simultaneous or alternating, representation of the actions and the observability of the actions. These form the initial requirements for the methods that shall be reviewed.

2.1 Reinforcement learning

A commonly used approach to user-modelling is reinforcement learning (RL). In RL, the AI agent seeks to form a reward-maximizing policy through trial-and-error interactions in its environment as shown in figure 1. The policy can then be used to model near-optimal behavioural skills. However, applying RL methods to train the capability to reason human behaviour requires a large amount of learning activity, which is difficult, costly, or otherwise non-feasible to obtain [7]. In order to overcome this issue, the demand has increased for methods that may somehow avoid the costly data-collection stage [7]. Examples of such methods could be

Figure 1. Reinforcement learning framework



data-driven reinforcement learning methods that provide techniques of pre-teaching the AI before it interacts with its environment [7], as well as inverse reinforcement learning (IRL), which can use very minimal and generic data records for learning.

The suitability of RL algorithms for the task of user modelling lies in their suitability for solving problems of bounded optimality. Determining optimal actions is similar to the concept of human rationality, where the assumption is that the actions of a rational agent are determined by the perceived maximal utility in its operating environment. The goals of computational simulation of rationality thus line up with the basic elements of RL: utility (or reward) function, environment and a bounded learning agent.[9] Usage of RL in the context of modelling human behaviour is common practice in computational rationality framework, which has recently emerged as a new direction for understanding interaction between human and computer [10].

2.2 Inverse reinforcement learning

One of the most well-studied approaches to user-modelling with AI is inverse reinforcement learning [4]. Compared to regular RL, the inverse method attempts to infer the reward function which resembles the behaviour of the agent under inspection, and then proceeds to behave according to it [5]. Whereas in regular RL the AI learns by observing the consequences of different actions, the algorithm in IRL makes an assumption that the agent under observation is already following an optimal reward function, thus it only needs to deduce the policy that is followed.

The inherent drawback of such method is the reward model of the tar-

get may not be the reward model that the AI should use, and thus some conversion needs to be defined [5]. IRL in user modelling is not as simple as it may sound, since the reward models in human behaviour can be complex or intractable, which is proven to be often impossible to infer a reward function from undirected demonstrations [7]. Some variations of IRL, such as collaborative IRL (CIRL), attempt to solve this issue by introducing more feedback from the user in the modelling process [4].

2.3 Offline reinforcement learning

The previous description of traditional reinforcement learning requires the agent to interact directly with the target user and environment, making it essentially an online learning method. Compared to online RL, offline RL refers to reinforcement learning that can be taught before it is deployed into the target environment. Essentially, this reduces the risk and cost involved in the process of learning policies, which makes offline RL particularly interesting in domains such as robotics and healthcare [8]. However, the limitations of current machine learning tools hinder the use of pre-existing data difficult for reinforcement learning. Also, the lack of successful actual offline RL methods has been bypassed with alternative offline methods such as using a simulator to obtain policies. [8].

The immediate limitation of offline learning is the variety of available data, since a data set may not cover some important aspects of the agent's behaviour, or it may not perform well in new environments or agents with different behaviour. Second important issue is the nature of offline learning, which requires forming hypothesis that performs better than what the behaviour presented in the dataset is. In order to achieve such performance, it is required to mix orders of actions, which is not possible with the currently available machine learning tools due to the resulting shift in distribution. [8]

3 Examples of using reinforcement learning in user modelling

This section discusses some case studies where reinforcement learning has been applied in user-modelling. The studies were picked based on their recency, application of reinforcement learning methods and the focus on the problem of creating an accurate user model to provide the user better value.

3.1 Case study 1: Personalized task difficulty in MOOC

The first case study by Zhang et al. [14] reviews the application of reinforcement learning methods to personalize the task difficulty in Massive Open Online Course (MOOC) system. Goal was to implement a dynamic machine learning algorithm, which could capture the individual differences between the users' skill levels and adjust the difficulty of the tasks to suit individual user's capabilities. The study showed that a machine learning based method would significantly help slower users in completing their tasks, but it provided no benefits in slowing down faster users with increased difficulty. The results are that with machine learning methods, the automatic difficulty adaptation worked better than the traditional methods. This can provide better engagement of users into the learning platform, which the study shows to hold true especially on players of lower competency level.

The RL algorithm was based on one-step Markov Decision Process where the actions are completed tasks, difficulty of which correlates negatively with the grade the user receives. In order to make the algorithm responsive, the study used policy learning method called *Bootstrapped Policy Learning*, which provides a guaranteed unbiased convergence to an optimal difficulty level in a noisy environment. The responsiveness was also increased by predicting an initial ranking with an offline clustering method, which would take place before applying the online reinforcement learning.

3.2 Case study 2: Reducing user fatigue in Virtual Reality applications

The paper from Cheema et al.[2] studies the use of RL methods for user modelling in virtual reality application. Goal was to find out how to reduce unnecessary "mid-air movement", which refers to the VR-users arm movement when using the controllers. The motivation is that having too much mid-air movement causes fatigue, which could be minimized with an accurate user model. The results provide more efficient and relaxed posture policies for the users.

The study was conducted by using offline reinforcement learning method called *Proximal Policy Optimi*, which uses neural network to optimize the policy. The training of the algorithm was done without having any real-world data available. Instead, the training data was created by creat-

ing randomized movement patterns and predicting the movements that would require the least effort. User models were then created by gauging the users in a virtual environment based on estimates from literature.

3.3 Case study 3: Learning user models in energy sharing systems

In a case study by Timilsina et al. [11] RL method was implemented to model user preferences in energy sharing systems (ESS). Goal was to optimize the performance of an ESS with realistic user behaviour model in terms of preferences, engagement, and bounded rationality. In the current state-of-the-art implementation, the system matches and recommend users who are willing to share or requesting energy based on the amounts provided or requested with the potential transmission losses included. The study attempted to solve the problem by implementing a recommendation system based on the users' behaviour models, which could provide more shared energy by increasing the engagement from the users into the system. Results showed that including user modelling and RL provides significant performance improvements compared to state-of-the-art approaches with 25% higher efficiency and 27% more transferred energy.

The study applied two algorithms, *User Preference Learning* (UPL) and *BiParTite-K* (BPT-K), both of which rely on RL to form the user model. The study addressed an issue with initialization time that slowed down both RL algorithms and proceeded to calculate the initial weights with *Faster Initialization Algorithm*. The initialization phase was required to work based on preferences collected from the users, so there seemed to be no possibility to train the model with pre-collected data.

4 Discussion

All three analysed case studies showed success in implementing a RL method to the task of user-modelling, as can be seen on table 1. The first case study showed an approach to the issue of slow responsiveness with reinforcement learning and offline clustering. The study by Cheema et al. [2] showed that taking advantage of deep RL with neural networks can turn offline RL into a very powerful tool to generate valid approximations of ground truth human data. Most importantly, each of the studies succeeded in applying RL methods to user models without excessive pre-

Table 1. Results of reviewed case studies

Application	**RL method**	**Reported result**
Personalized task difficulty in MOOC [14]	*Bootstrapped Policy Learning*	Significant increase in helping students with difficulties
Reducing user fatigue in Virtual Reality applications [2]	*Proximal Policy Optimi (Unity toolbox)*	The fatigue measurements decreased as the user's input efficiency increased
Learning user models in energy sharing systems [11]	*User Preference Learning, BiParTite-K*	Significant performance improvements over the state-of-the-art methods

liminary data requirements, which could encourage more research on the subject.

Compared to the initial expectations, the state of user modelling with reinforcement learning has not yet reached a point where it could be easily applied to create realistic user models with ease. There seems to still remain a blind spot in the research where psychological user behaviour model could be applied for potentially greater results of the methods [11]. It has also been noted that the implementation of RL methods to predict human behaviour still requires efforts to understand how to model human motivations, contexts, learning, and social interactions[10]. This could provide a starting point for further research on the subject. For modelling certain narrow snapshots of human behaviour, reinforcement learning seems to be a feasible method that can provide good results without excessive costs.

5 Conclusion

This paper showed that applying RL to modern demanding user modelling tasks can be done successfully. Some of the drawbacks of conventional RL algorithms, such as need for costly training data, can be avoided by using offline RL, inverse RL or other assisting algorithms as shown in the case studies. User modelling can. The research could be continued here to cover how a more embedded application of behavioural psychology can improve user modelling with reinforcement learning algorithms

References

[1] Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[2] Noshaba Cheema, Laura A. Frey-Law, Kourosh Naderi, Jaakko Lehtinen,

Philipp Slusallek, and Perttu Hämäläinen. Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground, 2021.

[4] Sebastiaan De Peuter, Antti Oulasvirta, and Samuel Kaski. Toward ai assistants that let designers design. *arXiv preprint arXiv:2107.13074*, 2021.

[5] Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative inverse reinforcement learning. *arXiv preprint arXiv:1606.03137*, 2016.

[6] Antti Kangasrääsiö, Kumaripaba Athukorala, Andrew Howes, Jukka Corander, Samuel Kaski, and Antti Oulasvirta. Inferring cognitive models from data using approximate bayesian computation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1295–1306, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Antti Keurulainen, Isak Westerlund, Samuel Kaski, and Alexander Ilin. Learning to assist agents by observing them. In *International Conference on Artificial Neural Networks*, pages 519–530. Springer, 2021.

[8] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[9] Richard L. Lewis, Andrew Howes, and Satinder Singh. Computational rationality: Linking mechanism and behavior through bounded utility maximization. *Topics in Cognitive Science*, 6(2):279–311, 2014.

[10] Antti Oulasvirta, Jussi P. P. Jokinen, and Andrew Howes. Computational rationality as a theory of interaction. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[11] Ashutosh Timilsina, Atieh R. Khamesi, Vincenzo Agate, and Simone Silvestri. A reinforcement learning approach for user preference-aware energy sharing systems. *IEEE Transactions on Green Communications and Networking*, 5(3):1138–1153, 2021.

[12] Zhengwei Wu, Paul Schrater, and Xaq Pitkow. Inverse rational control: Inferring what you think from how you forage, 2018.

[13] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1469–1478, New York, NY, USA, 2020. Association for Computing Machinery.

[14] Yaqian Zhang and Wooi-Boon Goh. Personalized task difficulty adaptation based on reinforcement learning. *User Modeling and User-Adapted Interaction*, 31(4):753–784, Sep 2021.

Habituation and self-control: What SETA can learn from NeuroIS

Johanna Sänger
johanna.sanger@aalto.fi

Tutor: Sanna Suoranta

Abstract

In the fifteen years since the emergence of Neuroscience applied to Information Systems (NeuroIS) as a field, research has been conducted on several topics relating to security. These topics include habituation to security warnings and the role of self-control in security policy violations. However, the results of these studies are not used within the security education, training and awareness (SETA) field. This paper explores the NeuroIS research on habituation and self-control and current SETA research. The paper discusses how the NeuroIS research can be used to improve SETA and indicates several research gaps where the two fields can be combined.

KEYWORDS: NeuroIS, habituation, self-control, SETA, security education

1 Introduction

Users are often the weakest link in a security system. User behaviour plays a part in many security failures [11]. Users forget their passwords, share them with colleagues, or leave their computers unlocked [8]. Educating users can improve their security behaviour. Security education, training and awareness (SETA) is commonly employed by organisations

for security governance. The implementations and effectiveness of SETA vary widely between organisations.

The field of neuroscience applied to information systems (NeuroIS) was proposed in 2007 to study information systems using methodologies and theories from cognitive neuroscience [3]. NeuroIS research has been conducted within the context of security as well, with studies on the effect of habituation on human responses to security warnings [1, 12] and on how self-control affects intention to break security policy [7]. The studies on habituation have shown that as users are exposed to the same warning multiple times, their response to the warning decreases [12]. When the user sees a warning for the first time, a mental model is created about the warning. Upon repeated exposure to the same or similar warnings, the user automatically and unconsciously compares the warning to the mental model that the user has of the warning. The study on self-control found lesser neural engagement in the decision making process for low self-control individuals compared to high self-control.

The results from NeuroIS research have not yet being implemented in SETA. This paper explores the current research within the fields of SETA and NeuroIS, and proposes several possibilities for bridging the gap between the two fields.

Section 2 summarises current literature on SETA. Section 3 summarises three NeuroIS studies that focus on security, in particular on habituation and self-control. Section 4 combines these two fields and makes several suggestions for future works. Finally, Section 5 provides concluding remarks.

2 Security education, training and awareness

The weakest link of security changes with the development of the computer and internet. Acre [2] states that at the main security concern of mainframes was flaws within the operating system and its security controls. Physical access to mainframes was limited due to physical access control and security clearance requirements. As such, opportunistic attacks were rare. Physical access to computers became more straightforward with the rise of personal computers due to the lack of physical access control. With physical access control no longer being a deterrent, the predominant threat was computer viruses. Virus spread was minimal due to limited bandwidth. Virus spread increased with the emergence

of hard-disk technology, as now viruses could spread between otherwise isolated computers by the use of floppy disks.

With the emergence of the internet, computers were open to the world [2]. Different technologies emerged to withstand the new threats, including firewalls, encrypted connections, authentication systems, and virtual private networks (VPN). As technology emerged and posed new threats, new technologies were developed to protect against those threats.

The aforementioned technologies make up the technological infrastructure of an organisation. This infrastructure is typically implemented and managed by trained IT specialists [2]. However, individual workstations are usually the responsibility of end users, who are often the least trained individuals within an organisation. As such, it is important to train, educate, and make end users aware of security measures that they can take.

End users make security mistakes for many reasons. Sasse et al. [11] identified several issues that lead to unsafe password behaviour, including password sharing among colleagues being seen as a sign of trust in them, users thinking they will not be targeted, users thinking they will not be held accountable, or users thinking that someone getting into their account will not be able to cause serious harm.

Despite users often behaving unsafely with passwords, passwords are commonly used for authentication. Passwords typically have to adhere to rules that restrict the characteristics of a password. Complex password rules that require unique passwords for each system and need to be regularly updated cause users to create weaker passwords, repeat passwords, create passwords with patterns in them, or write down passwords on paper or in unprotected electronic files [8, 11]. Users are only willing to spend a reasonable amount of effort on security-related tasks [9]. Reducing the effort needed to perform security-related tasks by reducing the cognitive load is effective at getting users to adopt more secure habits.

One method of user compliance is to increase awareness, education, and training related to security. McCrohan et al. [8] investigated whether giving users extensive information influenced the strength of the passwords chosen by the users in the study. The group of users that was given extensive information on why password strength is important selected stronger passwords two weeks after the lecture than the group who received a general background on passwords. This research highlights the importance of educating users on the threats of internet use and the steps the users can take to mitigate some of the risk.

Increasing awareness alone might not be effective. Guo and Yuan [6] studied the effect of the organisational level of sanctions on their effects on the intention of security policy breaking by users. The researchers found that sanctions that came from the immediate peers of users and from users themselves were more effective at dissuading security policy breaking intentions than sanctions that were made on an organisational level. Guo and Yuan attribute the effectiveness personal self-sanctions to the individuals feeling accountable for their actions, and the effectiveness of peer sanctions to the disapproval of peers when individuals engage in non-compliant behaviour. Additionally, the authors conclude that organisational level sanctions do not affect the actions of individuals significantly. Next to that, Guo and Yuan found that the more senior an employee is, the more likely the employee is to violate security policies. Guo and Yuan suggest to focus on peer-sanctions and self-sanctions as an alternative to deterrence-based research.

Furnell and Vasileiou [4] plead that security education and training should shift from what users should know to how to get the users to know what they should know. Rather than tailoring security training to the sector of work, the authors argue that security training should be tailored to the individual based on five parameters: role, prior knowledge, barriers, learning style and security perception. These parameters determine the attitude of an individual towards security awareness. The individual attitude should shape how the personal security training plan is put together, e.g., whether push- or pull-based influence should be used.

3 NeuroIS research

NeuroIS is a relatively young field, with the term coined in 2007 by Dimoka et al. [3]. They proposed using methods from cognitive neuroscience when studying information systems, allowing an "objective, reliable and unbiased measurement of thoughts, beliefs, and feelings and link them to specific human processes" [3].

A 2020 systematic review [10] of 200 papers in 2008-2017 NeuroIS research highlights several trends. First, the increase in completed empirical studies indicates the maturity of the field. Second, the main topics studied within the first decade of NeuroIS are emotion, stress, attention, trust, and technology acceptance. Third, more NeuroIS research has been conducted on the autonomic nervous system than on the actual brain

activity. The autonomic nervous system can be measured using pupil dilation, heart rate, skin conductance, fEMG, and blood pressure. Measuring brain activity requires the use of functional magnetic resonance imaging (fMRI) or electroencephalography (EEG). Riedl et al. [10] argue that the higher cost of MRI and EEG machines could play a role in this. They suggest that intrusiveness of fMRI and EEG is a more likely reason, due to its restriction of freedom of movement, natural position or invasiveness compared to eyetracking, heart rate monitoring or skin conductance measurements.

Using EEG, Hu et al. [7] examined difference in self-control affected event-related potentials (ERPs) whilst participants deliberated over violating security policies. Hu et al. [7] define event-related potentials as "an index of the activity of populations of cortical neurons measured at the scalp to sensory, cognitive, or motor stimuli".

Hu et al. [7] investigated whether the self-control theory of criminology is also applicable in information security. The theory of self-control states that whilst all humans have the same potential to commit crimes, it is the difference in self-control that determines whether someone becomes a criminal. The participants of the study were screened on self-control using the scale by Grasmick et al. [5]. Twenty participants per group were chosen from the 25% highest and lowest scoring participants, respectively.

Hu et al. [7] proposed and validated a "novel ERP paradigm for studying individual behavior in the context of information security". The paradigm is a combination of a scenario that permits security violations to occur, whilst at the same time being able to be used in a laboratory setting, and motivating participants to truthfully respond to the stimuli. The stimuli were one of three types: control, minor, and major. Control stimuli presented routine decisions with no relation to information security that were usually inconsequential. The minor and major violation stimuli were both related to information security, with minor and major possible consequences, respectively.

Whilst Hu et al. [7] found no significant difference in the reported answer to the stimuli between the high and low self-control groups, the EEG data showed lesser neural recruitment of the right and left prefrontal cortex in the low self-control group. The authors suggest that this may indicate that both high and low self-control engage similar processes for deliberating security violations, but that the low self-control group engages these processes less or engages them more superficially. These results

lend "some support to the idea that self-control is a stable characteristic of an individual, formed early in life, and remains relatively stable throughout the life span" [7]. This finding is significant for information security and will be further discussed in Section 4.

A second NeuroIS study that focused on information security is the study by Anderson et al. [1] on the habituation to security warnings, as well as the follow-up study by Vance et al. [12]. In the initial study, Anderson et al. [1] analysed habituation to security warnings using eye tracking and defined habituation as "decreased response to repeated stimulation". To decrease the effects of habituation, the researchers proposed polymorphic security warnings. The security warnings have nine variations, visible in Fig. 1: color of text, highlight of text, signal word, pictorial signals, ordering of options, background color, size of warning pop-up, contrast, and border. The polymorphic security warnings caused lesser habituation compared to the static warnings. However, due to the experiment set-up where the participants got to see 200 warnings in a row in a laboratory setting, the authors remark that "results may differ from habituation in actual practice" [1].

Vance et al. [12] addressed the cross-sectional aspect of the previous study. The researchers conducted a three-week longitudinal field study in which the participants were asked to install three applications per day and rate them. The applications were installed through a third-party app store controlled by the researchers. One group of participants was presented with static security warnings and another with polymorphic security warnings. The participants were asked to install only the apps without risky permissions. The results showed that the participants habituated more to the static warnings than to the polymorphic ones.

Vance et al. [12] also conducted a longitudinal eye tracking and fMRI study. Whilst lying in an MRI machine, participants were presented with 80 static warnings, 80 polymorphic warnings, 80 general software images, and 20 unique general software images for that day. The 260 images were displayed in a random order over two blocks. For each image, the participants rated the severity of the content on a four-point scale. This process was repeated for five consecutive days at the same time-of-day. The fMRI data showed that the left and right ventral visual processing streams displayed habituation effects.

(a) Original warning screenshot

(b) Color of text variation

(c) Highlight of text variation

(d) Signal word variation

(e) Pictorial signals variation

(f) Ordering of options variation

(g) Color variation

(h) Size variation (3X larger)

(i) Contrast variation

(j) Border variation

Figure 1. The static security warning (a) and the nine polymorphic variations (b-j) as proposed by Anderson et al. [1].

4 NeuroIS applied to SETA

Furnell and Vasileiou [4] state that "there is potentially a lot to be learned from broader fields such as education, marketing and communications" which can be applied to how security education, training and awareness is implemented. Next to these fields, NeuroIS, and perhaps the wider neuroscience discipline, can provide novel insights into teaching security.

The field of NeuroIS studies the subconscious ways in which humans interact with technology with the use of neuroscience research methods. Studying the subconscious decision-making processes provides a better understanding for how the human mind works in the face of security. A better understanding of the decision-making processes allows for better security education and training.

This paper focused on three NeuroIS studies, by Hu et al. [7], Anderson et al. [1] and Vance et al. [12].

Hu et al. [7] concluded that self-control affects the decision-making process, as the group with low self-control participants showed lesser neural recruitment of the right and left prefrontal cortex than the high self-control participants. Whilst both groups engage similar processes for evaluating whether to commit security violations, the low self-control participants engage these processes less or more superficial. Security education and training should be tailored to the self-control aspects by providing clear consequences of the security risks taken by the users. These consequences should highlight both the security-implication of their actions and the company policies for dealing with such infractions.

McCrohan et al. [8] showed that the level of information users have influence the strength of passwords chosen by them. The study did not investigate the role of self-control in the selection of passwords. As such, there is a research gap on how self-control and information levels interact when it comes to user behaviour. Possible interplay between self-control and information ought to be researched, as this could open up new ways to tailor SETA to the individual.

Anderson et al. [1] and Vance et al. [12] determined that users habituate to security warnings. The habituation effect is stronger for static warnings than for polymorphic warnings. When not presented with security warnings, the habituation recovers to some extent though never to the original level.

The habituation effect is important to SETA in three ways. First, the

way security warnings are designed matter. Users habituate faster to the static warnings. Implementing the polymorphic warning artefact of Anderson et al. [1] and Vance et al. [12] is one method to lower the habituation to the warnings. Another possibility is to vary the security warnings between different systems or different types of warnings within a single system. This option combines the polymorphic approach for the overall set of security warnings seen with static designs for individual ones. However, this approach does not work if a user predominantly sees one type of warning. The polymorphic warning artefact has drawbacks as well. One drawback is that users are taught that interfaces look one particular way and to look for specific indicators of authenticity. The polymorphic warning design increases the cognitive load of this task, as the users have to remember the indicators for several variants of the interface. Another drawback is that polymorphic security warnings are easier to mimic, as a non-matching interface can be interpreted by users as a polymorphic instance rather than an actual security risk. As such, the feasibility of polymorphic security warnings in practice should be researched.

Secondly, the habituation effect should be taken into account for security education and training as well. If users have to repeat the same security education and training material regularly, with the same material presented in the same manner, users will habituate to this training material by relying on their prior mental model, rather than fully paying attention to the material at hand.

Thirdly, habituation can be useful. As humans get more used to something, the cognitive load of that task lowers. The decrease in cognitive load means that the task takes less out of the compliance budget. The lesser impact of a task on the compliance budget, the more likely the user is to comply with the safety regulation. As such, habituation is useful in certain situations.

5 Conclusion

Studying how humans interact on a subconscious level with computers and information systems allows insight into the underlying decision-making processes. Visualising these decision-making processes gives researchers a better understanding in why users interact with security systems in a particular way. Understanding how users interact with systems is fundamental to developing good teaching material and methodologies on how

to get users to behave more securely.

This paper explored how three studies from the field of NeuroIS can and should influence security education, training and awareness. Within the field of SETA, research has been conducted on the social aspect of security, such as at what company level sanctions are the most effective [6], and that informed users make better security decisions [8]. As Furnell and Vasileiou [4] indicated, the 'what' of SETA is often known, and the focus should shift to the 'how'. SETA research should focus more on the methods of teaching users to behave more securely and for that understanding how users act on a subconscious level is crucial. However, the results from the NeuroIS studies discussed are not yet being implemented in SETA. This is a research gap that ought to be addressed. This paper proposed two particular instances that should be investigated: a feasibility study of polymorphic security warnings in real systems, and the interaction between levels of self-control and information on security behaviour.

Bibliography

[1] 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. A Comprehensive Perspective on Information Systems Security - Technical Advances and Behavioral Issues.

[2] I. Arce. The weakest link revisited [information security]. *IEEE Security Privacy*, 1(2):72–76, 2003.

[3] Angelika Dimoka, Paul A Pavlou, and Fred D Davis. Neuro-is: The potential of cognitive neuroscience for information systems research. In *Proceedings of the 28th international conference on information systems*, pages 1–20, 2007.

[4] Steven Furnell and Ismini Vasileiou. Security education and awareness: just let them burn? *Network Security*, 2017(12):5–9, 2017.

[5] Harold G. Grasmick, Charles R. Tittle, Jr. Robert J. Bursik, and Bruce J. Arneklev. Testing the core empirical implications of Gottfredson and Hirschi's general theory of crime. *Journal of Research in Crime and Delinquency*, 30(1):5–29, 1993.

[6] Ken H. Guo and Yufei Yuan. The effects of multilevel sanctions on information security violations: A mediating model. *Information Management*, 49(6):320–326, 2012.

[7] Qing Hu, Robert West, and Laura Smarandescu. The role of self-control in information security violations: Insights from a cognitive neuroscience perspective. *Journal of Management Information Systems*, 31(4):6–48, 2015.

[8] Kevin F. McCrohan, Kathryn Engel, and James W. Harvey. Influence of awareness and training on cyber security. *Journal of Internet Commerce*, 9(1):23–41, 2010.

[9] Shari Lawrence Pfleeger, M Angela Sasse, and Adrian Furnham. From weakest link to security hero: Transforming staff security behavior. *Journal of Homeland Security and Emergency Management*, 11(4):489–510, 2014.

[10] René Riedl, Thomas Fischer, Pierre-Majorique Léger, and Fred D. Davis. A decade of neurois research: Progress, challenges, and future directions. *SIGMIS Database*, 51(3):13–54, jul 2020.

[11] M A Sasse, S Brostoff, and D Weirich. Transforming the 'weakest link' – a human/computer interaction approach to usable and effective security. *BT technology journal*, 19(3):122–, 2001.

[12] 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. *MIS Quarterly*, 42(2):355–380, 2018.

Comparing different tools for computer-aided proofs in cryptography

Joona Ahonen
joona.ahonen@aalto.fi

Tutor: Christopher Brzuska

Abstract

This paper reviews and compares the different cryptographic proof assistants that have been developed and used. These proof assistants are grouped together based on what general proof assistant they rely on and whether they operate in the computational or the symbolic model. Coq is the most common base prover to be used by cryptographic proof assistants, but many do not utilize a base prover at all, implementing everything themselves. Computational model is more used than symbolic model, but not by a large factor, as computational model is generally more practical, but in contrast symbolic model is easier to incorporate into computer-aided cryptography. This paper concludes that cryptographic proof assistants have very varying approaches in implementing tools for computer-aided cryptographic proofs, each with their own capabilities.

KEYWORDS: cryptographic proof, computer-aided proof assistant

1 Introduction

In his paper, Halevi [17] introduces the reader to computer-aided cryptographic proofs. He states that over the past, cryptographic protocols have become increasingly complex and long. Showing the security and correctness of cryptographic implementations manually is difficult and prone to many errors, and furthermore verifying that the proof is valid is a cumbersome process [17] [4]. This may result in the possibility of vulnerable cryptographic implementations being used in practice. For example, the new TLS 1.3 standard is over a hundred pages long [24]. As a result of some proofs of correctness of cryptographic implementations or security assumptions being so long and complex, too few will bother to verify the proofs made by their peers. In the worst-case scenario, this might even encourage creators of cryptographic implementations to not write the proof at all.

To address this problem of increasing complexity of cryptographic proofs, computer-aided cryptography (CAC) has become a growing research area [4]. CAC utilizes the automation and functionality of computers and programming languages to programmatically verify the correctness of a cryptographic proof, possibly revealing flaws in the proof or finding attacks against the implementation. There are many tools created for these purposes, such as EasyCrypt or CryptoVerif, with varying approaches, such as the computational and symbolic models. A brief introduction is given to the most notable tools to find the main similarities and differences between them.

This paper reviews what kind of cryptographic proof assistants have been developed and what they aim to solve. These tools are categorized based on whether they work in the computational or symbolic model and on the general proof assistants behind them. The papers and tools are included in this paper based on their relevance and whether they have evidence of being useful. A good indicator of a cryptographic prover's usability is whether it has been useful in the proof of some cryptographic statement.

The second chapter introduces important general proof assistants Coq and Isabelle/HOL used by many computer-aided cryptographic proof assistants. In the third chapter, the most important cryptographic proof assistants are introduced. Finally, the discussion and conclusions based on the findings of the third chapter are in the fourth chapter.

2 Preliminaries

2.1 Game-based proofs

Game-based proofs are a method of proving the security of a cryptographic primitive, where a threshold is defined for the probability that an adversary gives a certain output after interacting with the game. The term game hopping means that the goal of game-based proofs is to modify the games slightly, so that the games are indistinguishable. The goal of this is to reach a game, for which the security is easier to prove than for the original one.

2.2 Protocol models

Security protocols can be modelled in the symbolic model or the computational model. The symbolic model, often called Dolev-Yao model, assumes perfect cryptography [10] meaning that an adversary should not be able to break the security property of a protocol with any probability. The symbolic model is a method of modelling cryptographic primitives which are represented as black-box function symbols, where the messages consist of terms based on these primitives [10]. The adversary can only utilize these primitives in its computation [10].

The computational model is a more realistic and widely used, but complicated alternative to the symbolic model for cryptographic proofs [10]. In the computational model the cryptographic primitives are described as functions from bitstrings to bitstrings, where the adversary is a probabilistic Turing machine [10]. The security parameter specifies the length of keys used in a protocol. To prove a security property in the computational model, one must prove that the probability for it not holding is negligible in the security parameter [10]. An important remark is that both of these models ignore physical attacks such as side-channel attacks [10].

2.3 Cryptographic system security

Information theoretic security and computational security are methods to separate cryptographic systems. A directly stronger assumption than perfect cryptography is information-theoretic security or unconditional security which holds for a cryptographic system if even with infinite comput-

ing power the possibility of breaking the system is zero [20].A more practical definition of computational security is weaker in the sense that it can be broken with unlimited computing power, i.e. it is enough that breaking the security of the system is computationally infeasible [20]. For example, to achieve perfect secrecy, a secret key equal in length to the plain text is needed [20], which further acts as evidence of the impracticality of information theoretic security.

3 General proof assistants

Proof assistants are tools which are used to help create and verify proofs of mathematical problems. By verifying proofs with machines, users gain considerable confidence in the correctness of the proofs in contrast to proofs generated and checked by humans [18]. This chapter briefly introduces Coq and Isabelle/HOL, the general proof assistants mostly used by the cryptographic proof assistants considered in this paper.

3.1 Coq

As is described by the Coq documentation [18], Coq is a tool for proving theorems where users can describe mathematical concepts, after which Coq is used to interactively create proofs for these theorems that are then checked by Coq. To generate a proof with Coq, a sequence of tactics is entered by the user, which construct the steps of the proof [18]. Given a theorem statement as a goal, tactics are used to transform the goal to subgoals, which can also be transformed to further subgoals and then all the subgoals are verified instead [18]. The correctness of definitions and proofs are automatically verified by the Coq compiler [22].

3.2 Isabelle/HOL

The most used instance of Isabelle is Isabelle/HOL which is a proof assistant with a higher-order logic theorem proving environment, which supports both automatic reasoning tools, and interactive operation modes [1]. Isabelle uses Isar as its formal proof language [1]. Isabelle provides tools, such as the *classical reasoner*, which can be used to prove formulas by performing long sequences of reasoning steps, and the *simplifier*, which reason about equations and utilize equations in its reasoning [1]. Executable specifications can be turned directly into code in SML, OCaml,

Haskell, and Scala [1].

4 Cryptographic proof assistants

4.1 CertiCrypt and EasyCrypt

In their paper, Barthe et al.[6] present CertiCrypt which is a framework built on top of the Coq proof assistant in the computational model, utilizing the game-based techniques for proving the security of cryptographic systems by creating and verifying code-based cryptographic proofs. As its formalism, CertiCrypt adapts pWhile, which is an imperative programming language often used in describing games, utilizing structured data types, procedure calls, and random assignments [6]. CertiCrypt provides exact security, which is used to produce exact bounds for the advantage and execution time of the adversary compared to the common approach of showing a negligible advantage for an adversary [6]. The proofs created with CertiCrypt are verified automatically and independently by a proof checking engine utilizing Coq [6]. Relational Hoare Logic (RHL) is formalized by CertiCrypt to utilize tactics to support the code-based reasoning of CertiCrypt [6]. As an example, CertiCrypt has been successfully used to develop the proof of existential unforgeability under adaptive chosen-message attacks for the Full Domain Hash signature scheme [25].

In their paper, Barthe et al. [5] present EasyCrypt which is a tool in the computational model [3] that automatically creates proofs of the security of cryptographic systems from proof sketches. Proof sketches are first verified by chosen satisfiability modulo theory (SMT) solvers and theorem provers, producing Coq files, which are then checked by CertiCrypt [5]. The properties of a security proof are captured as a sequence of games and hints [5]. With EasyCrypt, the games are represented as programs in an imperative language, and adversaries are described as abstract procedures, which have access to a defined list of oracles[5]. According to the comparison of CertiCrypt and EasyCrypt by Barthe et al.[5], EasyCrypt proofs generally require less than a third of the code compared to CertiCrypt, and EasyCrypt proofs run approximately twice as fast. As EasyCrypt utilizes CertiCrypt, and EasyCrypt is easier to use, EasyCrypt can be thought of as a successor for CertiCrypt. As an example given by

Barthe et al. [5], EasyCrypt has been used in developing proofs of security for Hashed ElGamal encryption and the Cramer-Shoup cryptosystem.

4.2 The Foundational Cryptography Framework

In their paper, Petcher and Morrisett [23] introduce The Foundational Cryptography Framework (FCF) which is a foundational framework in the computational model built on top of the Coq proof assistant. FCF is inspired by CertiCrypt, where the key difference between them is that instead of using deep embedding of a probabilistic programming language, FCF uses a shallow embedding with the functional programming language Gallina [23]. This allows the easy extension of the language and better utilizes Coq's tactic language and its current automated tactics [23]. The goal of this is to lighten the workload required when developing proofs [23]. As an example, FCF has been used to verify the cryptographic properties of an OpenSSL implementation of HMAC with SHA-256 such that the expected cryptographic properties are guaranteed by its functional specification [8].

4.3 CryptoVerif and ProVerif

In his paper, Blanchet [12] presents CryptoVerif which uses the computational model to analyse security protocols [12]. Utilizing game hopping, CryptoVerif converts the initial protocol with transformations into an ideal game, where the security property is easy to verify [12]. These game transformations are organized based on advice to prove protocols; if a transformation is unsuccessful, other transformations i.e., simplifications and expansions of assignments are suggested to reach the wanted transformation [12]. This property often allows protocols to be proved completely automatically [12]. The transformations can also be applied manually [12]. CryptoVerif can prove correspondences including authentication, and secrecy [13]. CryptoVerif has been used to verify the Full-Domain Hash (FDH) [15] signature scheme and Kerberos [9].

In turn ProVerif, an alternative tool developed by Blanchet [11] uses the Dolev-Yao model [14]. As its input ProVerif takes a model of the protocol, and the security properties that are to be proven [11]. ProVerif automatically converts the protocol from its input into a set of Horn clauses, and the properties to be proved are converted into derivability queries on

these clauses [11]. To find out if there may be an attack against the input protocol, ProVerif attempts to derive a fact from the clauses, where a successful derivation of a fact may mean there exists an attack against the security property [11]. But because some abstractions are made by the Horn clause representation, the derivation might correspond to a false attack [11]. As an example, ProVerif has been used to successfully analyse the protocol Just Fast Keying (JFK) [2].

4.4 CryptHOL

In their paper, Basin et al. [7] present CryptHOL which is a framework built on top of the Isabelle/HOL theorem prover by combining the accuracy of HOL with the structure of game-based proofs and utilizes mechanical theorem proving. CryptHOL operates in the computational model [3]. In CryptHOL, generative probabilistic values (GPVs), which are computations with input and probabilistic output, are combined with a functional programming language to model game-based proofs [7]. Isabelle is used to mechanically check all proof steps and definitions, to guarantee soundness and correctness [7]. CryptHOL attempts to keep the ideas of the main proof clear from low-level technical details in a declarative way with the help of the user, utilizing the structuring techniques and proof automation of Isabelle [7]. In addition, Isabelle's proof automation is extended with theory of relational parametricity, which is used to justify whether a game is equivalent to the game constructed by a game transformation in game-based proofs [7]. CryptHOL also gives users the ability to add new rules in which proofs are constructed to achieve extensibility [7]. As an example given by Basin et al. [7], CryptHOL has been used to prove the INS-CCA security of a symmetric encryption scheme built from an unpredictable function and a pseudo-random function.

4.5 VerifPal

In their paper, Kobeissi et al. [19] describe Verifpal which is an automated modelling framework and verifier that operates in the symbolic model, utilizing Coq. The goal of Verifpal is to be much easier to use than other tools with symbolic security analysis in order to alleviate the learning curve [19]. The protocols are modelled in a more intuitive manner with Verifpal, with the goal that the descriptions created are close to how they could be described in an informal conversation, but while

still being accurate and meaningful for formal modelling [19]. Because Verifpal only allows using built-in cryptographic functions, the modelling effectively avoids user errors [19]. Verifpal is compatible with Coq and the models can also be translated into ProVerif models [19]. As an example given in [19], VerifPal has been used in modelling and verifying unlinkability, freshness, confidentiality and message authentication in the Decentralized Privacy-Preserving Proximity Tracing (DP-3T) protocol.

4.6 Tamarin Prover

In their paper, Meier et al. [21] describe Tamarin, a prover that can execute symbolic analysis of security protocols automatically. However the automated option might not terminate [21]. If it terminates, it will return a proof of correctness or a counterexample, which can be for example an attack [21]. The tool can also be used in interactive mode, where the user can investigate the different states of the proof, view the graphs for attacks, and most importantly guide the proof manually alongside the automatic proof search [21]. As an example, Tamarin Prover has been successfully used to analyse the TLS 1.3 revision 10 specification [16].

5 Discussion

Based on the previous chapter, the proof assistants can be categorized based on whether they use Coq, Isabelle or something else as the base proof assistant, and in which model they work in. These results are summarized in Table 1.

Table 1. Summary of cryptographic provers

name	base prover	model
CertiCrypt	Coq	computational
EasyCrypt	Coq	computational
FCF	Coq	computational
CryptoVerif	Own	computational
ProVerif	Own	symbolic
CryptHOL	Isabelle/HOL	computational
VerifPal	Coq	symbolic
Tamarin Prover	Own	symbolic

Cryptographic provers are more commonly built on top of general proof assistants i.e., they translate their input to be used by the general proof assistant e.g., Coq or Isabelle/HOL. Having an established underlying proof assistant such as Coq will increase the trustworthiness and confidence of the results output by cryptographic proof assistants. Utilizing general proof assistants seems to be the most common approach; no point in implementing a totally new proof assistant just for cryptography when there are more general ones for math. But seems to not always be the case, as there are tools such as CryptoVerif, ProVerif and Tamarin Prover that seem to implement the underlying prover themselves. A possible explanation for this could be that the creators of these cryptographic provers sought after some feature that is not provided by the base provers, forcing them to implement more themselves.

As of the model in which the provers operate, the computational model is the favourite, but not by a large factor. Computational model's popularity may be explained by the practicality of it, but in contrast symbolic model's popularity may be explained by the fact that it is easier to incorporate into computer-aided cryptography.

In conclusion, the field of computer-aided cryptography is yet quite young and far from the final stages of development, as the tools in the field have very different approaches in implementing tools for computer-aided cryptographic proofs. There might even never be a single best approach to implementing these tools, as different cryptographic problems and implementations might always require different approaches when analysing them.

References

[1] What is isabelle?, 2021. https://isabelle.in.tum.de/overview.html.

[2] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. 10(3):9–es, Association for Computing Machinery, jul 2007.

[3] David Baelde, Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos, and Solène Moreau. An interactive prover for protocol verification in the computational model. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 537–554. IEEE, 2021.

[4] Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. Sok: Computer-aided cryptography. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 777–795. IEEE, 2021.

[5] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 71–90. Springer, 2011.

[6] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. *SIGPLAN Not.*, 44(1):90–101, jan 2009. https://doi.org/10.1145/1594834.1480894.

[7] David A Basin, Andreas Lochbihler, and S Reza Sefidgar. Crypthol: Game-based proofs in higher-order logic. *Journal of Cryptology*, 33(2):494–566, Springer, 2020.

[8] Lennart Beringer, Adam Petcher, Katherine Q. Ye, and Andrew W. Appel. Verified correctness and security of OpenSSL HMAC. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 207–221. USENIX Association, August 2015.

[9] B. Blanchet, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Computationally sound mechanized proofs for basic and public-key kerberos. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '08, page 87–99. Association for Computing Machinery, 2008.

[10] Bruno Blanchet. Security protocol verification: Symbolic and computational models. In Pierpaolo Degano and Joshua D. Guttman, editors, *Principles of Security and Trust*, pages 3–29. Springer, 2012.

[11] Bruno Blanchet. Automatic verification of security protocols in the symbolic model: the verifier ProVerif. In Alessandro Aldini, Javier Lopez, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII, FOSAD Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 54–87. Springer, 2014.

[12] Bruno Blanchet. Cryptoverif: A computationally-sound security protocol verifier. *Tech. Rep.*, 2017.

[13] Bruno Blanchet. Cryptoverif: Cryptographic protocol verifier in the computational model, 2022. https://bblanche.gitlabpages.inria.fr/CryptoVerif/.

[14] Bruno Blanchet. Proverif: Cryptographic protocol verifier in the formal model, 2022. https://bblanche.gitlabpages.inria.fr/proverif/. Accessed on 23.10.2022.

[15] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 537–554. Springer, 2006.

[16] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of tls 1.3: 0-rtt, resumption and delayed authentication. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 470–485. IEEE, 2016.

[17] Shai Halevi. A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Paper 2005/181, 2005. https://eprint.iacr.org/2005/181.

[18] CNRS Inria and contributors. Introduction and contents, 2021. https://coq.inria.fr/distrib/current/refman/.

[19] Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. Verifpal: Cryptographic protocol analysis for the real world. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *Progress in Cryptology – INDOCRYPT 2020*, pages 151–202. Springer, 2020.

[20] Ueli Maurer. Information-theoretic cryptography. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 8 1999.

[21] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 696–701. Springer, 2013.

[22] Christine Paulin-Mohring. *Introduction to the Coq Proof-Assistant for Practical Software Verification*, pages 45–95. Springer, 2012. https://doi.org/10.1007/978-3-642-35746-6_3.

[23] Adam Petcher and Greg Morrisett. The foundational cryptography framework. In Riccardo Focardi and Andrew Myers, editors, *Principles of Security and Trust*, pages 53–72. Springer, 2015.

[24] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[25] Santiago Zanella-Béguelin, Gilles Barthe, Benjamin Grégoire, and Federico Olmedo. Formally certifying the security of digital signature schemes. In *2009 30th IEEE Symposium on Security and Privacy*, pages 237–250. IEEE, 2009.

Intent-Based Networking: an overview

Joonas Harjumäki
joonas.harjumaki@aalto.fi

Tutor: Vesa Hirvisalo

Abstract

Traditionally network configuration has been a human-centric manual process. The constant need to improve operational costs and resource allocation has led to the emergence of Intent-Based Networking (IBN). IBN seeks to improve network management through automation. This paper provides an overview of IBN concepts and analyzes its strengths from a network management perspective. We find that the IBN addresses the significant issues of manual configuration.

KEYWORDS: autonomic networks, intent, networking

1 Introduction

In the IP networks routers and switches enable the flow of information by relaying packets of data. Even though IP networks have been adopted widely by the Internet, the management of traditional network equipment is hard. In order take into use high level network policies, operators need to configure individual network devices using low-level and vendor-specific methods. The static nature of configuration makes it hard to adapt networks to dynamically changing fault and load conditions. [7]

A network can be modeled to encompass two logical planes: the control

plane and the data plane. The control plane consists of rules that dictate what actions (e.g., drop, modify, forward, etc.) to apply to the traffic. The data plane applies the rules of the control plane to the traffic. Traditional networking devices are vertically integrated, meaning that the control plane and the data plane are bundled together in the devices. That integration reduces flexibility and innovation. [7]

Software-Defined Networking (SDN) aims to resolve the limitations in current network infrastructures. SDN breaks vertical integration by removing the control plane from switches and routers. That results in simple networking devices that only forward traffic. The control is performed in the logically centralized controller, simplifying enforcement of policies, (re)configuration, and evolution. It is essential to notice that centralized control doesn't imply physically centralized control. Production systems commonly deploy a physically distributed control plane to meet performance, scalability, and reliability requirements. [7]

A well-defined API between an SDN controller and a switch realizes the separation of control and data planes. Through the API, the controller has complete control over the operation of a data plane switch. One of the most notable SDN protocols is OpenFlow[11]. In OpenFlow, the API between the controller and the switch is called Southbound API, whereas the Northbound API provides an interface to the controller itself. An OpenFlow switch has one or many tables of rules (flow tables) that control the flow of packets through the switch. Rules, for example, drop, forward, and modify packets. Depending on the configuration of flow tables, an OpenFlow switch can act as a firewall, switch, router, or perform other operations like load balancing or traffic shaping. [7]

5G networks are expected to expand network services beyond mobile devices. Services like remote health care, autonomous vehicles and advanced robotics will benefit from ultra reliable, low latency and high-speed communications provided by 5G. [6] 5G supports a diverse set of services with very different communication requirements. Manual configuration of services is time-consuming, error-prone, and requires a great deal of expertise. Furthermore, the manual approach doesn't result in an optimal allocation of resources among services. Hence, it's necessary that network resources can be allocated automatically in an optimal configuration according to the demands of services. With current systems, the issue is that it's impossible to make significant configuration changes on-demand due to the complexity of traffic patterns. [1]

Intent-Based Networking (IBN) is a promising framework for automated network management [12, 1]. IBN builds on the programmability of the SDN controller. It enables the operator to define *what* policies are enforced in the network instead of defining *how* network equipment should be configured to meet those policies. The network intents provide a simple declarative approach to configuring complex networks. For example, the operator could declare that hosts A and B are allowed to communicate with a bandwidth capacity X without having detailed knowledge about network topology or switch configuration. [12] Ultimately IBN enables networks to organize, assure, heal and configure themselves. [1]

In this paper, we present the problems that arise in the management and operation of present networks. We present the core concepts of IBN and evaluate IBN as a solution to recognized problems. We also show how the concepts of IBN manifest themselves in a practical system.

The rest of this paper is structured as follows. In section 2, we discuss the challenges of present networks. The Section 3 presents the core concepts of the IBN. In Section 4 a practical system is presented. The findings of the paper are discussed in section 5. Finally, conclusion is in the section 6.

2 Challenges of present networks

The operation of networks (e.g. ISP, enterprise, data center, campus, home network) is ruled by high-level network policies that are derived from network-wide requirements. Policies consider connectivity, security and performance requirements. Policies can be of static or dynamic nature. Dynamic policies are the ones that are triggered on demand. Traditionally, network administrators have translated high-level network policies to low-level device configuration commands that are provisioned to routers, switches, and specialized middleboxes, like firewalls and proxies. The translation process is in a large part a manual activity performed by experienced network administrators. In large organizations, many parties, like server administrators, network engineers, and DNS administrators, define policies concerning their responsibility area to be applied organization's network components. It requires manual coordination between the parties to ensure that the combined set of policies is conflict-free and it implements high-level policies set by each party. [9]

In the current distributed policy management, it takes from days to

weeks to plan and implement new policies. The slow and partly manual process of cross-checking the policies of different parties is required to ensure the correctness and consistency of the new policy. Despite the careful checking of policies, problems still arise after the deployment of the policies. Users might lose connectivity, security breaches might result from vulnerable components, and the performance of applications might degrade. It would be ideal to automatically recognize conflicts between policies and to form a conflict-free policy set before provisioning it to the physical infrastructure. Furthermore, if there were a high-level abstraction layer for defining policies, that were decoupled from the infrastructure, it would ease the burden on administrators and users for implementing policies. [9]

3 Intent-Based Networking

RCF9315 recognizes that it is infeasible to manage modern networks by configuring individual network devices with low-level methods. Keeping device configurations consistent across a network is challenging, not to mention that configurations must remain consistent with respect to the needs of services provided by the network. Further challenges are imposed by the requirement for the network to adapt dynamically to the changing needs in a scalable way. Also, there is a constant need to automate network operations more in order to lower operational costs but also to enable fast reconfiguration of networks on the sub-second scale. Automation also helps to ensure that network is working as intended. [2]

To address these issues, autonomic networks have gained much attention by the discussions in the ANIMA Working Group of the IETF. The goal of autonomic networks is to lower operational costs and to simplify management of the networks in general. Even though autonomic networks have self-management properties they still need information about the purpose and goals of the network. That information is commonly referred as an intent and a network that accepts input from operators in form of intents is called an Intent-Based Network. To implement the functionality, there is a centralized and a distributed approach. In the centralized approach there is a control application running on a set of servers whereas in the distributed approach the functions are distributed to network nodes and they in a cooperation implement the functionality. [2]

Intents are not just a form of interaction in a higher-level of abstraction. They let operators to focus more on what they want to achieve instead of focusing on the implementation details. The ability to focus on desired outcomes results in a better operational efficiency and flexibility, and shorter time scales. It also reduces dependence on error-prone human activities. Combing Intent-Based Networking with artificial intelligence can eventually bring network automation to the next level. [2]

3.1 Intent and Intent-Based Management

RFC9315 defines intent as a declarative set of operational goals and outcomes. The declarative nature means that the intent doesn't specify how to achieve or implement it. The term "intent" appeared first in the context of autonomic networks, where it meant guidance information that a user provided to an autonomic network, which operated otherwise without human intervention. The goal of Intent-Based Management is to simplify network management and operation so that networks require only minimal user intervention. Even autonomic networks need information about what is required from them, e.g., what policies they must enforce or what services they must provide. That information constitutes an intent. [2]

The declarative nature of intents implicitly applies several important concepts. First of all, it provides data abstraction. The users don't need to be aware of low-level device configuration details. Secondly, the functional abstraction frees the user from figuring out how to achieve the given intent. The Intent-Based System (IBS) derives the required course of action by applying an algorithm or translation rules to the intent. [2]

Ideally, autonomic networks would translate the intent into a course of action and apply it by themselves. The centralized orchestrator that processes the intent would not be needed and the network devices would use distributed algorithms and local device abstractions to take the intent into the use. Because the intent applies to the network as a whole, ideally, the intent would be automatically distributed across network devices and the devices themselves would decide the required actions. [2]

In practice, complete decentralization is not desired. Users, for example, require a single logical point of interaction through which they make requests to the network in the form of intent and receive updates about the network status. Also, most network devices can be simple packet-forwarding devices without any intent processing capabilities, or their processing power could be too low for intent processing. In those cases,

there needs to be a separate system that performs the required actions to fulfill the intent. [2]

A logically centralized system would also be useful if particular intent requires a complete view of the network. Due to the sheer size of data describing the network or time lags related to the propagation of information across the network, it might be infeasible to maintain a complete view in each network device. [2]

Whether the implementation is decentralized or centralized an Intent-Based Network is a network that is managed using intent. The network takes input from the user in form of intent, translates it into course of action and achieves the desired outcome. The outcome is achieved without getting any technically detailed information concerning required steps from the user. [2]

The functionality of taking an intent as a input from a user and applying it to a network is called the intent fulfillment. This functionality is described in section 3.2.

3.2 Intent fulfillment

Intent fulfillment functionality consists of interfaces that users utilize to enter an intent into the system and functions that perform the required actions to achieve the intent. Algorithms that derive the required actions, functions that continuously learn to optimize network operation, and orchestration that coordinates the provisioning of configuration commands to the network are also part of this functionality. [2]

Functions of intent fulfilment can be divided into three categories: intent ingestion, intent translation, and intent orchestration.

Intent ingestion

The process of fulfilling an intent begins with intent ingestion. Here the system obtains the intent from the user and possibly lets the user refine the intent until it is actionable by the IBS. Usually, the user interface involves intuitive workflows that guide a user through the process of entering an intent, ensuring that all required information for intent modeling and subsequent translation has been gathered. Instead of only prompting the user for input, the user interface may provide user clarifications, explain ramifications and trade-offs, and facilitate refinements. [2]

The ultimate goal is that the IBS is as natural as possible for the user to use. The expectation is that IBS doesn't involve a steep learning curve,

for example, in form of requiring users to learn a new "language" of the system. Ideally, the system will learn from the user and not the other way around. [2]

Intent translation
The intent translation is concerned with translating the user intent into network configuration commands that can be provisioned to the network. The translation may result in multiple alternative configurations, which all achieve the intent. To choose the optimal alternative, translation may involve algorithms that learn about optimal configurations over time. [2]

Intent orchestration
Intent orchestration provides functions that orchestrate the provisioning of configuration commands across the network. Configuration commands are produced by intent translation. [2]

4 Lumi

Lumi[3] is a Intent-Based System that takes input from an operator as an intent in natural language. Lumi translates the intent into network configuration commands and deploys them to the network. Lumi consists of four modules which are executed in succession: information extraction, intent assembly, intent confirmation, and intent deployment. Each module can be replaced in a plugin-and-play fashion with an alternative solution. [3] In the subsections each module is described in detail.

4.1 Information Extraction

The entry point to Lumi is in the information extraction module. The entry point is based on a chatbot interface. The entities are extracted from the natural language intents using Named Entity Recognition (NER)[5]. Due to the popularity of personal assistants like Amazon's Alexa and Google Assistant, Lumi's goal is to serve users with little technical know how (e.g. home users) in addition to traditional network operators. Providing a natural language interface for the traditional user base of network operators is beneficial as the teams are often composed of operators with different levels of expertise and experience. [3]

Lumi solves the NER problem using Recurrent Neural Networks (RNN)[8]. Lumi's NER process is a supervised learning algorithm. The training data

is a set of input-output pairs where input is an intent expressed in natural language, and output is a set of entities and their labels. Although, NER is in general considered as a solved problem, practical challenges still remain. It requires careful *entity engineering*, which refers to selecting appropriate entities for the given problem, in order to achieve an acceptable accuracy. [3]

The output of the NER is a set of labeled entities that are parsed from the intent. In other words, entities are the set of operations supported by the system. In Lumi, entities are organized hierarchically. Raw textual values are at the bottom, forming the vocabulary Lumi understands and are referred to as common entities. For example, values for *@middlebox* are network functions like firewall, traffic shaping, and packet inspection. Intermediate level entities are composite entities that are formed by aggregating prepositions to common entities. For example, the *@origin* class contains composite values "from @location" and "from @service". The top of the hierarchy consists of immutable entities, and they form the core of Lumi. *@operations* class, for instance, expresses operations supported by Lumi. [3]

4.2 Intent Assembly

The input for the intent assembly module is a set of extracted entities from the information extraction module. For example, if the input to the chatbot were "Please add a firewall for the backend.", the intent assembly module would be given these entities: *{middleboxes: 'firewall'}, {target: 'backend'}*. Clearly, these entities alone don't enable checking the correctness of the intent. For that purpose, Lumi assembles extracted entities into a structured and well-defined intent. Lumi utilizes Network Intent Language (Nile)[4] as the intent definition language. The Nile grammar lets Lumi to check that the assembled intent is syntactically correct and contains all required information. If some information is missing, Lumi can prompt the operator to provide missing information through the chatbot interface. For example, if the operator had issued a "Please add a firewall." intent without specifying the target, the intent assembly module would not construct a Nile intent but would ask the operator to specify the target. [3]

The intent definition language also enables the operator to easily confirm the assembled intent. The operator could write the intent directly in the intent definition language, but it is less burden for operators to

use natural language and confirm the assembled intent afterward. It's well known that a person who understands a sentence is not necessarily capable of producing one. [3]

As an example, below is the Nile intent for input like "Add firewall and intrusion detection from the gateway to the backend for client B with at least 100mbps of bandwidth, and allow HTTPS only" as shown in [3]:

```
define intent qosIntent:
   from endpoint ('gateway')
   to endpoint ('database')
   for group ('B')
   add middlebox ('firewall'), middlebox ('ids')
   set bandwidth ('min', '100', 'mbps')
   allow traffic ('https')
```

4.3 Intent Confirmation

The challenge of using natural language as input is that it is inherently ambiguous. The same intent can be expressed in many different ways. Despite the recent advances in natural language processing, it is prone to produce false positives or false negatives. Lumi addresses this issue with intent confirmation. The intent is presented to the operator for confirmation before provisioning it to the network. [3]

4.4 Intent Deployment

The last stage of intent processing is the compilation of the Nile intent into a code that can be deployed to appropriate network devices. Lumi compiles the intent into a Merlin[10] program. Merlin was chosen over other network configuration languages, because it is a good match to Nile in terms of supported network features, it performs well and the source code is available. After the compilation to a Merlin program, there still exist unresolved logical handles, such as low-level IP addresses, VLAN IDs and IP prefixes, in the program. Lumi uses information provided during the bootstrap process to resolve the handles to actual values. Once handles are resolved, Merlin compiles the Merlin program into OpenFlow rules. [3]

5 Analysis

IBN enables network operators to work on a higher level of abstraction in the form of intents. This leads to better operational efficiency as operators can focus on desired outcomes instead of implementation details. The intents are automatically checked for conflicts and errors by the IBS, which means that the risk of facing issues after provisioning the intent is lowered. The autonomic operation of the IBN enables the network to optimize its operation over time. That also enables the automatic scaling of resources in response to the demand.

6 Conclusion

We reviewed the problems that arise with present networks from management and operation perspectives. We find that manual configuration is, in general, time-consuming and error-prone, and it doesn't allocate resources optimally. In addition, configuration errors can go unnoticed in the manual review of the configuration, resulting in connectivity issues, security breaches, and performance degradation after the deployment of the configuration. We showed that the autonomic nature of the Intent-Based Network and the interaction with the network in the form of intent addresses the significant issues of manual configuration.

We gave an overview of an Intent-Based Network and its functions. As a practical example of an IBN, we showed how the core functions, namely intent ingestion, intent translation, and intent orchestration, are implemented in Lumi[3].

References

[1] Khizar Abbas, Talha Ahmed Khan, Muhammad Afaq, and Wang-Cheol Song. Network slice lifecycle management for 5g mobile networks: An intent-based networking approach. *IEEE Access*, 9:80128–80146, 2021.

[2] Alexander Clemm, Laurent Ciavaglia, Lisandro Granville, and Jeff Tantsura. Intent-based networking-concepts and definitions. *RFC 9315*, 2022.

[3] Arthur S Jacobs, Ricardo J Pfitscher, Rafael H Ribeiro, Ronaldo A Ferreira, Lisandro Z Granville, Walter Willinger, and Sanjay G Rao. Hey, lumi! using natural language for {Intent-Based} network management. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 625–639, 2021.

[4] Arthur Selle Jacobs, Ricardo José Pfitscher, Ronaldo Alves Ferreira, and Lisandro Zambenedetti Granville. Refining network intents for self-driving networks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 15–21, 2018.

[5] Dan Jurafsky and James H Martin. Speech and language processing. vol. 3. *US: Prentice Hall*, 2014.

[6] Talha Ahmed Khan, Afaq Muhammad, Khizar Abbas, and Wang-Cheol Song. Intent-based networking platform: an automated approach for policy and configuration of next-generation networks. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1921–1930, 2021.

[7] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[8] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[9] Chaithan Prakash, Jeongkeun Lee, Yoshio Turner, Joon-Myung Kang, Aditya Akella, Sujata Banerjee, Charles Clark, Yadi Ma, Puneet Sharma, and Ying Zhang. Pga: Using graphs to express and automatically reconcile network policies. *ACM SIGCOMM Computer Communication Review*, 45(4):29–42, 2015.

[10] Robert Soulé, Shrutarshi Basu, Parisa Jalili Marandi, Fernando Pedone, Robert Kleinberg, Emin Gün Sirer, and Nate Foster. Merlin: A language for managing network resources. *IEEE/ACM Transactions on Networking*, 26(5):2188–2201, 2018.

[11] OpenFlow Switch Specification. Version 1.5. 0. *Open Networking Foundation*, 2014.

[12] Benjamin E Ujcich, Adam Bates, and William H Sanders. Provenance for intent-based networking. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 195–199. IEEE, 2020.

Microservices - when and how to use them

Juho Pousi
juho.pousi@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

A microservice architecture has become a popular choice over a traditional monolithic architecture. However, overall it is not typically clear when the microservice approach is better than the monolithic alternative. A literature survey was conducted to figure out which characteristics of microservices support the shift from the monolithic architecture to the microservice architecture.

We determined better scalability and maintainability in addition to suitability for agile development processes to be the most significant factors to promote the microservice architecture. On the other hand, microservices introduce additional complexity which defends monolithic approaches. We concluded that to gain benefits of the microservice architecture a large enough application is required, and the team's experience of the microservice architecture impacts the adoption complexity. Still the monolithic architecture is well-suited for example for small load applications.

KEYWORDS: microservice architecture, monolithic architecture

1 Introduction

A microservice architecture has become a popular approach to design services and has taken a foothold from traditional monolithic architecture. *Monolithic architecture* is a software architectural pattern where all server-side software components are packed to one program [11]. *Microservice architecture* is another architectural approach where the single service application is split to multiple modularized services which typically communicate via Representational State Transfer (REST) Application Programming Interface (API) [9]. The microservice architecture has evolved over the last decade, when the next generation of software development tools like containerization, monitoring and continuous delivery has emerged [9], finally leading to introduction of the term *microservice* in 2011 [11].

Traditional monolithic architecture has many problems concerning on scalability, technology lock-in, resilience, deployment and large codebase, for instance [12]. In addition, the shift to agile world in software development field even highlights the issues related to the monolithic architecture as it may not suit well for agile processes such as high deployment cycle. Microservice architecture aims to solve problems of the monolithic architecture for example by improving scalability and allowing faster deployments, in addition to being better architecture for today's cloud environments [9]. While the microservice architecture tackles some problems of the monolithic architecture, there is no clear winner that suits in every case.

This paper aims to evaluate when the microservice architecture is more suitable choice over the monolithic architecture. The paper performs this by collecting the main forces towards the microservice architecture over the monolithic architecture, and on the other hand the reasons to still use the monolithic architecture.

First this paper in Section 2 presents the relevant background of the microservice and the monolithic architectures. Section 3 discusses the benefits and drawbacks of adopting the microservice architecture over the monolithic architecture. In Section 4 we discuss observations concerning the topics presented in this paper. Finally, Section 5 summaries the main findings of this paper.

2 Microservice and monolithic architectures

Martin Fowler [6] raises the foggy definition of a software architecture noting that there exists different opinions in the field how the architecture should be defined. He emphasizes Ralph Johnson's idea that the architecture is "the shared understanding that the expert developers have of the system design" [6]. This definition allows us to recognize different patterns to implement the software architecture such as monolithic and microservice architectures, which this paper focuses on. Next Sections 2.1 and 2.2 present the main ideas of the monolithic architecture and the microservice architecture respectively in order to perform comparison in Section 3.

2.1 Monolithic architecture

A software application relying on the monolith architecture consists of "tightly coupled" components, which together implement the functionality and logic of the application [2]. All of these components are required at build or execution time. The literature uses the term *monolith* both for the application, which is compiled to the single executable and for the application that consists of modules [5]. In case of a modularized monolith, modules depend on the shared underlying resources such as database and memory, indicating that an individual module can not be executed independently.

To tie the term *monolith* to the application stack we can consider the typical enterprise application [11]. There exists three main units that form the enterprise application: a client-side user interface, server-side application and database. Here the server-side application is the monolith that handles database and HTTP connections, performs the business logic and serves the frontend files to the client. This kind of monolith runs all logics related to a request in a single process [11]. An example of the monolith architecture is given in [17, Fig. 1]. Note the modular structure of the monolith, which is the server-side component described by a gray rectangle boundary.

2.2 Microservice architecture

Microservice architecture can be seen as a subtype of service-oriented architecture (SOA) [9], and thus it is relevant to briefly define SOA before

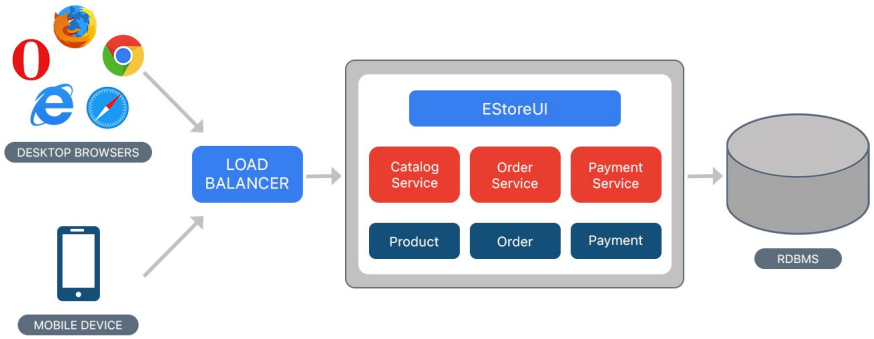


Figure 1. An example of monolithic architecture of E-Commerce Application [17].

diving into microservices. SOA can be defined as an architecture that aims to be a loosely coupled system consisting of self-contained units or on the other words services which communicate with each other forming an application [13]. A self-contained unit has responsibility of one functionality of the application and these services can be written in different languages. However, Fowler [11] for example says that SOA as a term is ambiguous and has different meanings for different experts. That is the reason why it is best to consider microservice architecture as an individual term also in this paper.

Defining microservice architecture is also difficult as it includes multiple concepts and different implementations. Thus, instead of defining it, Lewis and Fowler [11] try to express the microservice architecture using generally recognized characteristics. They outline that microservices consist of independent service components, which communicate for example via web service requests, and the service is responsible for one business capability. Another characteristic relates to the communication of services. Where SOA approaches rely on Enterprise Service Bus (ESB) communication mechanism, microservices on the other hand handle the communication inside the service using REST or some lightweight message bus such as RabbitMQ. Namely, the service itself receives a request or consumes a message, performs the requested logic and produces the response. One common and concrete characteristic they mention is decentralizing the data management, which in simply implies that each service uses separate service specific databases. In addition, a few other characteristics related to broader context of the microservice architecture were discussed, but we will return to those in Section 3 when performing the comparison.

Figure 2 [17] presents the E-Commerce application utilizing the mi-

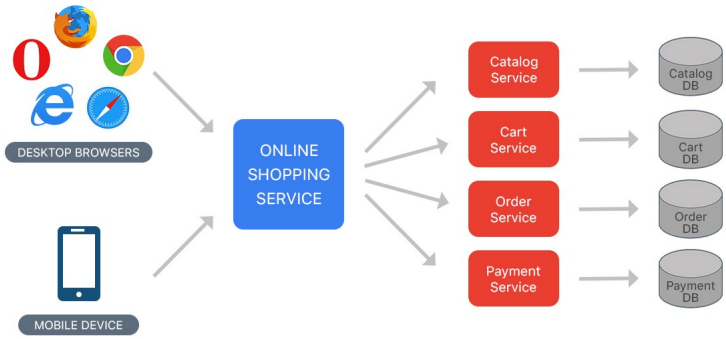


Figure 2. An example of microservice architecture of E-Commerce Application [17].

croservice architecture. Compared to the monolithic version of the application in [17, Fig. 1], here we can note the decomposition of the application to independent services and databases.

3 Microservice architecture compared to monolithic architecture

Monolithic and microservice architectures differ in multiple ways, both having numerous benefits and drawbacks. Since it would not be possible to discuss all of them in this short paper, we will focus on specific topics which were highlighted in multiple literature sources. Namely, we review performance, scalability, complexity, development process, reliability and security aspects. Naturally many topics interleave and multiple factors are considered under following subsections.

3.1 Performance

A choice between the monolithic and the microservice architecture may affect on the performance and efficiency of the application [8]. Performances of these two architectures were examined by performing HTTP request tests against two applications, which had similar functionalities. The first application relied on the monolithic architecture while another implemented the microservice architecture. The monolithic architecture clearly overcame the microservice architecture when 30 000 requests were sent at once. On the other hand, when the same test was performed with 300 000 requests, the microservice architecture handled more requests per second compared to the monolith.

In the previous study researchers concluded the monolithic architecture to be the better choice for businesses where the application load is rela-

tively small [8]. Correspondingly, the microservice architecture was reported to be more suitable for a software that is required to serve larger number of customers. While another research [1] concluded a similar result that monolith architecture overcame the microservice architecture with insignificant load such as less than 100 users, it discovered that microservice architecture outperformed the monolith only slightly when increasing the load. After all, evaluating the architectures simply by performance is difficult as performance depends on e.g., chosen scaling approach. In addition, performance may be measured by different metrics depending on the situation.

3.2 Scalability

In general, runtime scalability is frequently interpreted to be one of the main forces towards microservice architecture over monolithic architecture [16, 11]. Typically, vertical scaling is discussed in context of the monolithic architecture and horizontal scaling correspondingly with the microservice architecture. In vertical scaling more resources are allocated for the provided single server. However, physical hardware restricts vertical scaling. On the other hand, in horizontal scaling the service or application is replicated to multiple servers and e.g., load balancer is utilized to route the traffic. Horizontal scaling may be implemented for both architectures as Figure 3 [11] illustrates. Nonetheless, the problem is that horizontal scaling of monolith requires scaling of the entire application while microservice architecture allows scaling of the most demanding services [11].

Improved scalability was reported to be one of the highest motivations for transforming to the microservice architecture from the monolith architecture [16]. In addition, required infrastructure and scaling affect the running costs of the application. Researchers conducted that microservice architecture is more affordable approach in cloud environments such as Amazon Web Services (AWS) compared to monolithic architecture [18]. They noted that microservices implemented with serverless functions are the most cost-efficient solution when considering the infrastructure costs. Nonetheless, for a low demand application monolith architecture with a single virtual machine in cloud may be more affordable solution, as the research focused on large applications.

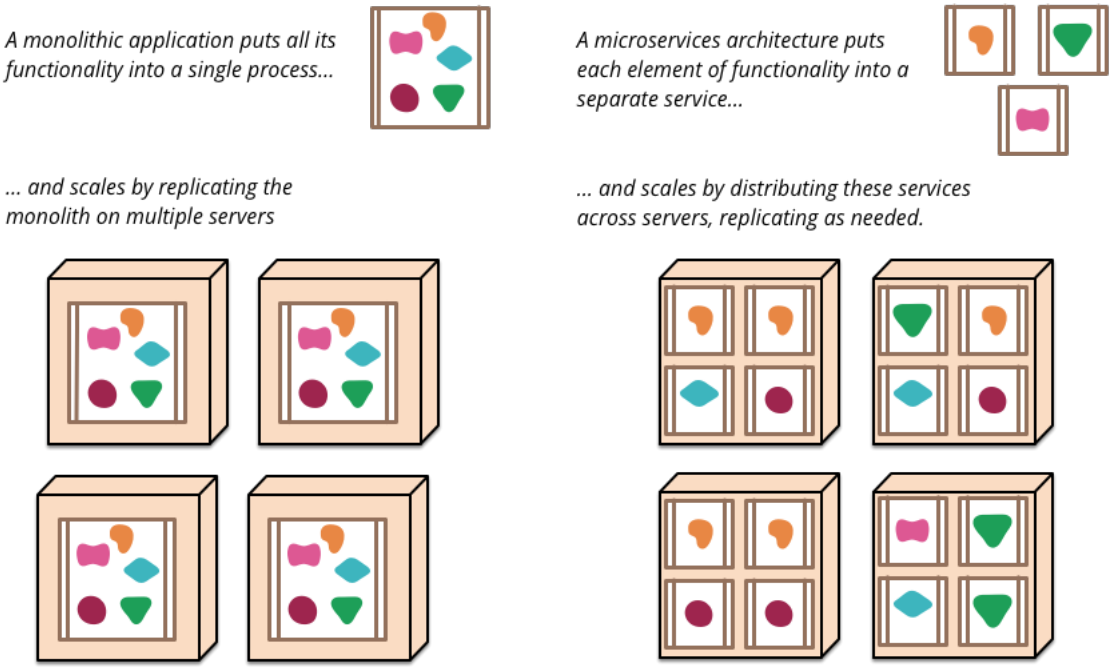


Figure 3. Illustration of horizontal scaling of monolithic and microservice architectures [11].

3.3 Development process and complexity

Semantically monolithic refers to something that is too large and unchangeable [4]. Monolithic architecture, at the first glance, is simple to develop, deploy and scale [15]. However, problems arise when the size of the monolith or the team grows. When the codebase of the monolith grows, it is more laborious to understand. Possible module boundaries tend to fade away, modifying the application becomes difficult and development process decelerates [15]. As a result the quality decreases and involving new developers becomes challenging. On the other hand, microservice architecture tries to tackle these problems by relying on multiple small services [14]. Smaller services are easier to understand and modify than monolith, also for new team members. Though, Martin Fowler [7] resembles that a large and well modularized monolith may be a proper choice instead of microservices as theoretically well modularized monolith should be easy to handle.

Both Richardson [14] and Fowler [7] discuss complexity. According to Fowler the microservice architecture suits for services with high complexity, otherwise a monolith is the optimal choice. Both of them however remind that while microservices reduce the complexity of an application, microservices also introduce the development complexity of dis-

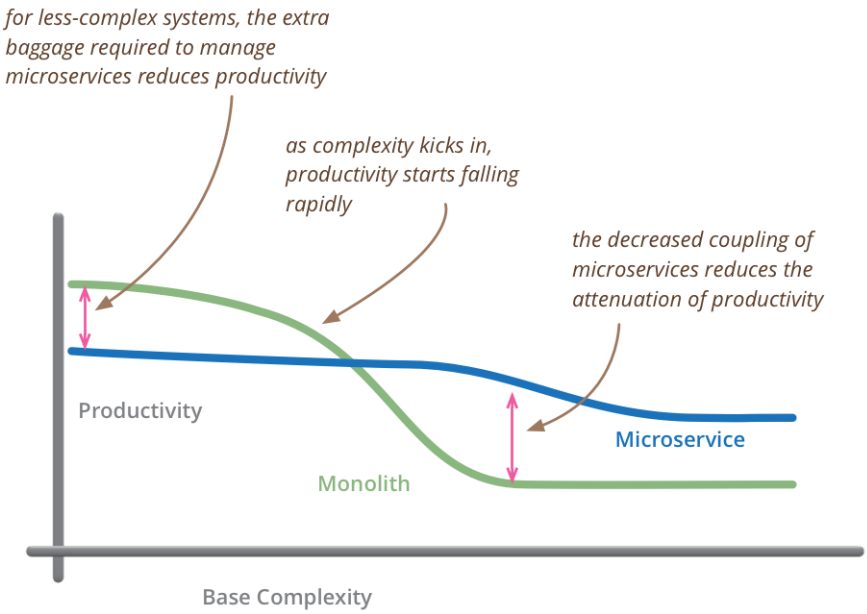


Figure 4. Development process productivity comparison of monolithic and microservices in function of complexity [7].

tributed systems. For example, Richardson [14] mentions increased implementation complexity related to communication between services, requests which depend on multiple services and testing service interactions. Fowler [7], in addition, mentions for instance automated deployment, failure handling and monitoring, as a source of complexity that microservice architecture introduces.

Figure 4 [7] illustrates how development team is required to address the additional complexity of microservice architecture in the beginning of the project, which decrease the development speed. On the other hand, when such an initial investment is completed and complexity begins to increase, the advantages of the microservice architecture help to maintain productivity better than the monolithic architecture. Although new businesses, like startups seeking to optimize time-to-market, may benefit high productivity of the monolithic architecture in the beginning [14]. However, when considering the mature project, microservice architecture allows faster time-to-market for new features due to continuous deployment [3]

Presently, continuous delivery and deployment are crucial parts of the development process. Monolithic architecture allows easy deployment as only a single executable or a package is deployed [15]. A drawback of the microservice architecture is that multiple services require deployments increasing complexity [14]. On the contrary, continuously deploying a

monolith can be difficult as the whole application deployment is required despite the scope of the implemented change [15, 11]. Richardson [15] mentions e.g., the risk that deployment of an entire application may produce bugs in components which were not modified, and as a result the courage to perform future deployments may decrease reducing the delivery cycle. Though, Lewis and Fowler [11] mention that automated pipeline for testing and deployment of monolith may be established quite easily and thus subsequent deployments are safer. However, microservice architecture enables deployment of services independently [14] and thus support faster deployment cycle. For example, different teams can perform deployments separately. Thus, selecting the best option in the context of the continuous delivery depends on the situation; for example, is it a problem to deploy an entire application at time, or is the team ready to handle the increased complexity that microservices produce.

There exist also remarks related to a technology stack. Monolithic architecture tends to force the team to use the technology stack selected at the beginning of the project, and updating to newer stack can require substantial amount of work [15]. Contrastingly, small loosely coupled microservices allow usage of different technology stacks in different services, and also renewing an old stack is easier in context of small units [14]. Lewis and Fowler [11] characterize this as a decentralized governance where the team has more freedom to choose the tools they use for building the software. Although they mention that monoliths allow this in some extent, it is more restricted.

3.4 Reliability and Security

According to literature, microservices achieve better reliability compared to a monolith [8, 14, 3]. While a single fault in the monolith may stop the entire application, fault in one microservice breaks only the corresponding microservice, but other services continue to serve customers [3]. To achieve that, microservices ought to be well-designed to handle failures and be able to isolate those. Lewis and Fowler [11] also list "design for failure" as one characteristic of the microservice architecture.

Another factor to consider is testability, and both architectures have their own pitfalls related to it. Monoliths are easier to test due to a single codebase and the fact that the entire application runs in a single process [3]. However, running tests takes a long time for a large monolith. Microservice architecture allows faster tests as only modified services can

be tested [14]. On the other hand, as mentioned before there exists some complexity e.g., related to service interaction testing [14, 3].

Ensuring application security requires more considerations in microservice architecture compared to the monolithic architecture [10]. Former has multiple smaller attack surfaces while the latter one has one large attack surface to secure. In case of microservices, more work e.g., related to encryption is required to secure the REST API connections between different services [5]. Monoliths, on the other hand, may implement the request validation at API level and after that pass messages between functions within the monolith [10]. Choosing the certain architecture does not secure the application. That is why extra attention is required to implement security in both cases but especially with microservices.

4 Discussion

While both microservice and monolithic architectures have multiple advantages and drawbacks, it is typically difficult to argue that for specific reason one option would overcome another. One reason may be that defining architectures itself is complicated, and many implementations may be labeled under a specific architecture. Literature also uses metrics which may be difficult to quantify, and evaluating the difference between architectures may be challenging with these metrics. For example increased complexity was frequently used to determine whether switch to microservices will emerge benefits. However, complexity itself is ambiguous and depends on the situation.

After all, main reasons why especially large businesses are adopting microservice architecture are improved scalability, maintainability and suitability of microservices for agile development process. In addition, an interesting motivation for adopting microservice architecture was "because everybody does it" [16]. Companies may not actually understand the benefits of microservices prior to adopting, but they follow the mainstream and possibly notice benefits after the adoptation. Nonetheless, replacing a monolith with microservices requires a substantial effort as splitting a monolith to microservices is a non-trivial task. It is interesting that this motivation gathered multiple mentions because transform investment takes time and money and could be supposed that the decision to transform would have been made after a careful evaluation.

While benefits and issues of the microservices are recognized and suc-

cessfully compared to the monolith, frequently it is fuzzy when to start using microservices over a monolith. Monoliths have drawbacks and while microservices try to address these they meanwhile introduce new issues and complexities. Rather than focusing only on the possible benefits, the company or the team should consider which drawbacks they are ready to live with. For example the team experience may be the most significant criteria when choosing the architecture. Like presented in Section 3 monolithic architecture may still be the best choice, especially if the team does not have experience of microservices.

5 Conclusions

We have evaluated differences between the microservice architecture and the monolith architecture. The purpose was to collect the reasons when the currently popular microservice architecture produces more value compared to traditional monolith architecture. We performed the literature survey to be able to present the most reported benefits and drawbacks of both architectures to provide a general picture of the topic.

We found e.g., better scalability, maintenance and modern DevOps practices like continuous deployment to be main forces to adopt the microservice architecture over the monolithic. On the other hand, we identified that monoliths perform better on small scale as microservices introduce overhead and reduce pure performance. In addition, more effort is required to establish microservice infrastructure compared to the monolithic option.

Frequently there exists a counterargument to both directions and optimal architectural choice depends on the specific business and its scale. Research in the field of microservices is quite an empirical due to historical emergence of microservices in practical business. We suggest more research to determine better metrics to quantitative determine the optimal architectural choice for different use cases.

After all, we have showed that as hard as it is to define the microservice architecture as it is to compare it to the monolithic architecture, not even speaking of selecting the winner. The microservice architecture and software architectures in general evolve continuously. Thus, we may soon notice many leaves of the microservice architecture which do not fit to generally agreed definition of microservices.

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, 2018.

[2] Rahul Awati and Ivy Wigmore. Monolithic architecture. https://www.techtarget.com/whatis/definition/monolithic-architecture, May 2022. Accessed: Sept. 26, 2022.

[3] Hiren Dhaduk. Monoliths vs microservices: Which is right for your application? https://www.simform.com/blog/monoliths-vs-microservices/, Apr 2022. Accessed: Oct. 6, 2022.

[4] Cambridge Dictionary. Meaning of monolithic in english. https://dictionary.cambridge.org/us/dictionary/english/monolithic, 2022. Accessed: Oct. 6, 2022.

[5] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. *Microservices: Yesterday, Today, and Tomorrow*, pages 195–216. Springer International Publishing, Cham, 2017.

[6] M. Fowler. Software architecture guide. https://martinfowler.com/architecture/, Aug 2019. Accessed: Sept. 26, 2022.

[7] Martin Fowler. Microservicepremium. https://martinfowler.com/bliki/MicroservicePremium.html, May 2015. Accessed: Oct. 10, 2022.

[8] 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, 2020.

[9] 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.

[10] George Lawton. 6 new requirements for securing microservices vs. monolithic apps. https://www.traceable.ai/blog-post/6-new-requirements-for-securing-microservices-versus-monolithic-applications. Accessed: Oct. 12, 2022.

[11] J. Lewis and M. Fowler. Microservices. https://martinfowler.com/articles/microservices.html, Mar 2014. Accessed: Sept. 25, 2022.

[12] Sam Newman. *Building Microservices*. O'Reilly Media, 2015. Accessed: Sept. 25, 2022. [Online]. Available: https://learning.oreilly.com/library/view/-/9781491950340/.

[13] Tom Nolle. Service-oriented architecture (soa). https://www.techtarget.com/searchapparchitecture/definition/service-oriented-architecture-SOA, Feb 2020. Accessed: Sept. 26, 2022.

[14] Chris Richardson. Pattern: Microservice architecture. https://microservices.io/patterns/microservices.html, 2021. Accessed: Oct. 6, 2022.

[15] Chris Richardson. Pattern: Monolithic architecture. https://microservices.io/patterns/monolithic.html, 2021. Accessed: Oct. 6, 2022.

[16] Davide Taibi, Valentina Lenarduzzi, and Claus Pahl. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32, 2017.

[17] Siraj ul Haq. Introduction to monolithic architecture and microservices architecture. https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63, May 2018. Accessed: Sept. 26, 2022.

[18] Mario Villamizar, Oscar Garcés, Lina Ochoa, Harold Castro, Lorena Salamanca, Mauricio Verano, Rubby Casallas, Santiago Gil, Carlos Valencia, Angee Zambrano, and Mery Lang. Infrastructure cost comparison of running web applications in the cloud using aws lambda and monolithic and microservice architectures. In *Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, CCGRID '16, page 179–182. IEEE Press, 2016.

Adversarial attacks and defenses on deep neural network classifiers

Kerkko Karttunen
kerkko.karttunen@aalto.fi

Tutor: Blerta Lindqvist

Abstract

Adversarial examples are maliciously crafted images that can deceive deep neural network (DNN) classifier systems into misclassifying objects. Research on adversarial examples seems to suggest that the existence of adversarial examples in the input space is an inherent part of DNN classifiers. This paper introduces some of the most prominent attack methods for generating adversarial examples, and defenses that can be utilized to mitigate the effects of those attacks. The paper also illustrates how adversarial examples are a threat in different threat models, such as in the physical world.

KEYWORDS: adversarial attack, adversarial defense, deep neural network, classifier

1 Introduction

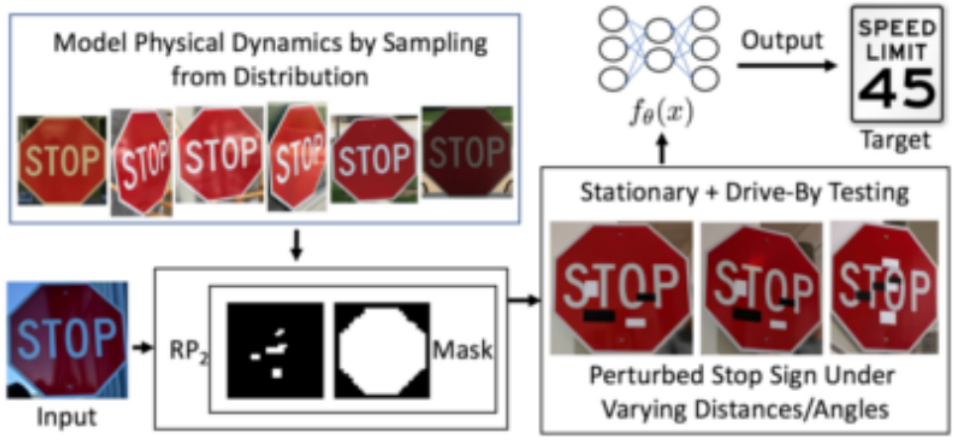
Deep neural networks (DNNs), such as image classifiers, are widely used in real-world applications. Image classifiers take images as inputs and output class probabilities, which can be used to classify an object in the image. Neural network image classifiers have been used early on for handwritten zip code detection with a size of just 1256 nodes [13], whereas

current classifier DNNs can comprise 650 000 nodes and be able to classify the ImageNet database [4] of a thousand classes [10]. Modern classifier systems are seemingly very capable of image recognition tasks with high accuracy.

However, the input image for DNN classifiers can be altered by introducing small perturbations, making the classifier misclassify the adversarial example image [23]. The perturbations can be so small that the adversarial examples are indistinguishable from the original images [23]. These perturbed images create an attack vector for attacking classifier systems and a challenging problem for defenders. Ilyas et al. [9] propose that the misclassification of adversarial examples is an inherent feature of DNN models, rather than a bug. Similarly, Goodfellow et al. [6] suggest that the excellent results from DNN classifiers are a facade due to the fact that outside of naturally occurring data (such as adversarial examples) the systems perform misclassifications.

When machine learning systems are introduced to real-world tasks, such as autonomous vehicles, healthcare, and other safety-critical systems [1], the significance of adversarial attacks and defenses becomes apparent. For example, street signs can be physically modified with stickers such that they are classified as another street sign by image classifiers [5]. Figure 1 shows the pipeline for creating adversarial examples in the physical world. Whereas street signs perturbed by other physical obstacles, such as snow or vandalism, could be misclassified, the possibility of adversarial attacks in the physical world in safety-critical systems presents a clear threat.

Figure 1. Figure depicting the pipeline of crafting an adversarial example in the physical world to change the classification of a stop-sign into a speed limit sign [5].



This paper gives an overview of adversarial attacks and defenses against them. The paper also aims to present the practical consequences of these attacks and motivate the need for robust defenses.

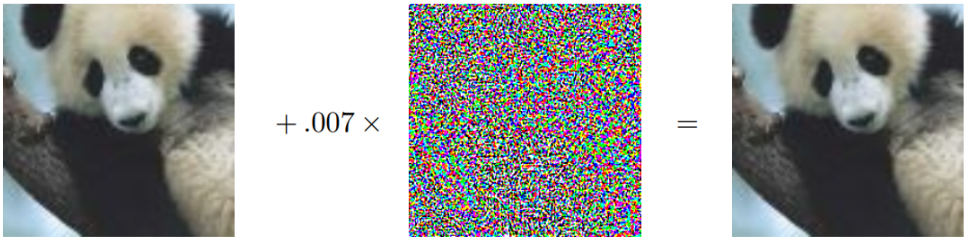
This paper is structured as follows. Section 2 illustrates the different kinds of adversarial attacks and the theory of various attack methods. Section 3 presents the standing defenses that have emerged to defend against the known attacks. Section 4 discusses the implications of adversarial attacks on machine learning systems, the continuing race between attacks and defenses, and known theories for the causes of this flaw. Section 5 presents the conclusions of the paper.

2 Attacks

A neural network classifier is a highly non-linear function from the input to the output, and when trained with a softmax activation function, the output layer represents a probability distribution of a label being assigned to the input [23]. These networks work on the idea of generalization, where inputs similar to the training data on one label result in the same label such that, for example, images of dogs from different perspectives are still labeled as dogs. The theory for creating adversarial examples was first formulated by Szegedy et al. [23]. They remark that there are areas in the input space that would result in an incorrect label, but they are hard to find with random sampling [23]. Yet these areas could be found by solving an optimization problem. This optimization is essentially gradient descent with the loss function of the DNN in order to find the positions of the input space that produce the adversarial label. The loss function essentially tells the difference between the predicted output and the actual output of the DNN. While the loss function of a DNN is minimized in training to have the best output from the system, the adversarial loss function is maximizing the error. Despite the different available attacks, many of them are based on this same idea.

Many prominent attacks are so-called white-box attacks, meaning that the attacker has access to the entire trained network. Alternatively, black-box attacks include attacks that have only partial or no information about the network.

Figure 2. Example of the FGSM-attack on ImageNet-based GoogLeNet network, where an imperceptible perturbation is applied to an image of a panda to generate an adversarial example labeled as a gibbon [6].



$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

2.1 Types of attacks

L-BFGS

Szegedy et al. [23] formalized the adversarial examples in the following way as minimization of two terms. This is an approximation with box-constrained L-BFGS.

- Minimize $c|r| + loss_f(x + r, l)$ where $x + r \in [0, 1]^m$

Here $f : \mathbb{R}^m \to \{1...k\}$ is the classifier that maps the input pixel values to the label set. This classifier function f has a continuous loss function $loss_f : \mathbb{R}^m \times \{1...k\} \to \mathbb{R}^+$. For an adversarial example, the term $x + r$ is the nearest input to x that maps to the adversarial label l. The first term minimizes the perturbation to keep it visually equivalent to the input image, whereas the second term minimizes the loss of the adversarial label. The constant $c > 0$ controls the balance between the terms and is used for performing the line search until the adversarial label is reached. The L_2 norm was used by Szegedy et al. [23] however any norm, such as L_∞ or L_0, can be used that can measure the distance between the original image and the adversarial example.

The minimization can also be formalized with x' being the adversarial example input:

- Minimize $c|x - x'| + loss_f(x', l)$ where $x' \in [0, 1]^m$

Fast Gradient Sign

The fast gradient sign method (FGSM) aims for the simple and fast generation of adversarial examples [6]. The method doesn't necessarily aim at making the perturbed samples minimally different from the original images [6]. Kurakin et al. [11] have also suggested an iterative version of the algorithm, which uses the same idea but iterates several times with small step sizes and clipping of intermediate results. The regular fast gradient sign method is as follows

- $x' = x + \epsilon sign(\nabla_x J(\theta, x, y)$, where x is the original image and the second term is the perturbation with θ being model parameters [6]

The required gradient can be computed with backpropagation as long as the model parameters, input, and output are known [6]. The gradient gives the direction and the epsilon gives the magnitude of descent along the direction. Figure 2 shows how FGSM can be applied to change the classification of a panda to a gibbon with $\epsilon = 0.007$.

PGD

The Projected Gradient Descent (PGD) -attack is an iterative attack, which resembles the FGSM-attack in that the method for finding x' with FGSM is similar to a single step in PGD [16]. The multi-step iterative variant of FGSM [11] is thus effectively PGD with a negative loss function [16]. The main difference to normal gradient descent using the loss function is that in PGD, the maximization of the loss happens subject to a selected constraint. PGD has been argued to be the best attack, which uses first-order information about a network [16].

JSMA

The Jacobian-based Saliency Map Attack (JSMA) introduced by Papernot et al. [20] is an iterative method based on the forward derivative and an adversarial saliency map of the DNN. The adversarial saliency maps indicate the features that should be perturbed in order to get adversarial examples efficiently [20]. The forward derivate is evaluated over the network directly, allowing the use of the forward propagation [20]. The resulting algorithms use these components to identify the most easily perturbed features and then modify them.

DeepFool

DeepFool is an iterative method, which models the boundary between the original class and other classes as a set of hyperplanes [19]. The algorithm is used to find the closest boundary and thus find the minimum perturbation required to change the classification.

CW

The CW-attack continues upon the work of Szegedy et al. [23] by changing the constant c to be a balancing factor in the second term [2]. They also insert a confidence parameter κ into the minimization. There are several versions of the CW attack based on which distance metric is used, for example, the attack with L_2 is the CW_2-attack. The different distance metrics may not be differentiable fully, such as with the CW_∞-attack, in which case the attack is performed iteratively [2].

2.2 White-box and black-box attacks

The introduced attacks have assumed full access to the target model. This is not necessarily unrealistic, since for example Szegedy et al. [23] noticed that adversarial examples could be transferred to other models. However, in a different attack model, the attacker can have a variable level of access to the original network and its outputs. When considering attacks on commercial models, each query usually costs money, which leads to a monetary limit on queries that can be made in the generation of adversarial examples [8].

Attackers can also have limited visibility into the model's outputs, such as just seeing probabilities for the top k classes [8]. In the label-only setting, only the output labels in order of probability are shown to the attacker [8]. Ilyas et al. [8] produced reliable adversarial examples with just the top label visible to the attacker. Thus even in an extremely limited information threat model, the adversary could be able to generate adversarial examples if the network is not adequately defending against the attacks.

2.3 Transferability

Szegedy et al. [23] created two different models, which were trained using the same training data, and found that an adversarial example generated on one model transferred to the other model as well, causing misclassification. When they partitioned the training data to create two differ-

ent models, the adversarial examples generated on one transferred to the other [23]. Therefore the adversarial examples can be transferrable between different models even when the training data is different. Thus the attacker may not need the target model but can create adversarial examples on one accessible model, and transfer the inputs to the target model. Resistance against transferability can be improved by increasing network capacity and using adversarial examples in the training data [16].

3 Defenses

Adversarial defenses are mainly based on modifying the network during training to be more robust to adversarial examples during training. Many defenses have been broken soon after publication due to new or modified current attacks being released, or due to failures to evaluate the proposed defenses extensively enough. Intuitively if DNNs are trained on images to divide into classes, but adversarial examples result in the wrong label, the examples can be fed to the network during training to increase robustness against an attack. This is essentially the idea behind Adversarial training.

3.1 Adversarial training

Adversarial training [16, 12] is the process of feeding generated adversarial examples to the DNN during the training phase. This process was noticed to reduce the effect of adversarial examples by Szegedy et al. [23] when they first formulated adversarial examples. Further research [6, 16, 12, 7] has shown how this process can be additionally honed to reach lower error levels on different kinds of networks and attacks.

Goodfellow et al. [6] were able to reduce the error rate from 89.4% to 17.9% on a DNN using adversarial training and FGSM to generate the examples. However, Madry et al. [16] proposed that using single-step methods such as FGSM left the networks vulnerable to multi-step iterative methods such as PGD, which indicates that the strongest attacks should be used in the training instead. Additionally, Kurakin et al. [12] proposed that larger high-capacity networks are by default more robust to adversarial attacks.

Huang et al. [7] built on top of the work by Goodfellow et al. [6] by focusing on maximizing the classification error with the adversarial examples

and minimizing it with the classifier in training. This is essentially giving the worst examples to the network to prepare it for the worst-case scenario [7].

3.2 Randomized Smoothing

Randomized smoothing for DNNs is a defense technique, where the inputs are smoothened using Gaussian noise [3]. Cohen et al. [3] propose that using randomized smoothing is advantageous for large models (such as ImageNet) and proves a robustness guarantee for its results. Using smoothness and noise for robustness has also been extensively studied by others, such as [14, 15]. Whereas randomized smoothing may prove to be theoretically efficient, its practicality against black-box attacks has been questioned [18].

3.3 Barrage of Random Transforms

Transforms on input images have been used in the training of neural networks to fortify them against data not completely covered by the training images. The barrage of Random Transforms (BaRT) method [21] combines several of these weaker transforms into a randomized large set of transforms applied on the input images. By introducing randomness, the attack attempts to make it harder for the attackers to predict the specific transforms used on the images [21]. Thus the attacker can not predict directly which gradient to attack and is forced to use, for example, averages of transformed gradients. Raff et al. [21] used the ImageNet dataset and beat the accuracy provided by Adversarial training when the model is under attack, resulting in 24 times the accuracy of previous defenses.

The robustness of random transformation defenses has been contested by Sitawarin et al.1 [22], who claim that it has not been rigorously tested. They propose that random transformation defenses have been evaluated with attacks that are not adequate to beat the defense [22]. Thus it remains unclear whether BaRT is a viable defense after all. BaRT has also been evaluated in the black-box setting, [17] in which it did not achieve notable defense accuracy.

4 Discussion

The plethora of available adversarial example generation methods, which stand undefeated against scrutinous review obviously indicates a need for robust defenses against them. Many defenses have been produced, but not many of them have been able to adequately defend against common attacks. Some attacks are also transferable between models, which makes the white-box attack threat model very viable, as attackers can use other models in a white-box setting to attack their target model.

DNN classifiers (and other DNNs) are vulnerable to attacks, which can be transferred between models and used in real-life threat models. Current safety-critical DNN classifiers can be fooled into making false predictions, which alters the perceived reliability and safety of these systems. Thus it is questionable whether these DNN classifiers can be used in safety-critical domains without reasonable doubt over their vulnerabilities.

While adversarial training and other defenses may increase the robustness of a network, new attacks could find adversarial examples, which the network has not been trained against, and thus bypass the defense. By using the strongest known attacks, this problem could be mitigated.

The cause for adversarial examples existing for classifier networks remains still somewhat unclear. Some suggest that the adversarial examples are a feature [9]. There have been several suggestions for the reasons for adversarial examples existing. It seems like the existence of adversarial examples is an inherent part of a DNN classifier, but the specific reason for this remains an intriguing topic to explore in future research.

5 Conclusion

This paper has given an overview of adversarial attacks, and defenses. The paper also considered the practical consequences of adversarial attacks. Adversarial attacks present a clear threat to DNN classifiers both in the digital and the physical world. While training a DNN classifier, one must consider the possibility of adversaries being able to fool the system, and thus use strong defenses, such as Adversarial training.

It may be that adversarial examples are an inherent part of DNN classifiers. Strong attacks such as PGD and the CW attack can be utilized in the training phase to achieve better robustness against the attacks. As

new attacks are introduced, new defenses must be considered, and old defenses must be updated to match the new threat profile.

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.

[2] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society.

[3] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing, 2019.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models, 2017.

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[7] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvari. Learning with a strong adversary, 2015.

[8] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information, 2018.

[9] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[11] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[12] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2016.

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[14] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy, 2018.

[15] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise, 2018.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[17] Kaleel Mahmood, Deniz Gurevin, Marten van Dijk, and Phuoung Ha Nguyen. Beware the black-box: On the robustness of recent defenses to adversarial examples. *Entropy*, 23(10):1359, oct 2021.

[18] Thibault Maho, Teddy Furon, and Erwan Le Merrer. Randomized Smoothing under Attack: How Good is it in Pratice? In *ICASSP 2022 - IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5, Singapore, Singapore, May 2022. IEEE.

[19] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2015.

[20] 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, 2016.

[21] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially robust defense. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6521–6530, 2019.

[22] Chawin Sitawarin, Zachary Golan-Strieb, and David Wagner. Demystifying the adversarial robustness of random transformation defenses, 2022.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

EEG biometrics for Authentication

Kiran Joy Kulangara
kiran.joykulangara@aalto.fi

Tutor: Sanna Suoranta

Abstract

With fingerprints and face recognition being widely used for authentication in phones and laptops, the use of biometric based identity verification has seen an exponential increase. Hence, there is a significant demand for development of biometric systems which can further enhance security and privacy. Recent research has demonstrated that EEG is a suitable psychometric signal for biometric verification, and possesses key characteristics, including resilience to spoofing attempts and impossibility to use under stress. This paper provides a brief introduction to the field of EEG based biometric authentication and discusses some of its benefits, drawbacks, constraints and related privacy concerns.

KEYWORDS: EEG, Authentication, User identification

1 Introduction

User identification and authentication has always been an important topic in the world of information security. In recent years, there has been an increase in the use of biometrics for authentication. This method of authentication involves using user's distinct biological traits as passwords. The uniqueness of these traits make biometric authentication highly ef-

fective.

Traditional methods of biometric authentication include the use of fingerprints, iris or voice as the authentication factor. However, simple spoof attacks are able to break these methods [1]. To overcome these challenges, several researches have focused on the possibilities of using biometric signals, such as EEG, ECG or EMG for authentication.

This paper reviews the use of *electroencephalogram* (EEG) biometric for authentication and its benefits and challenges. The remainder of this document is structured as follows. Section 2 discusses the genetic traits of brain signals and authentication based on them. Section 3 enumerates the benefits of using EEG signals as an authentication factor. Section 4 presents the challenges involved in EEG based authentication. Section 5 reviews the current state of the system in a real world setup and the scope for future research. Finally, Section 6 discusses the drawn conclusions.

2 EEG

EEG is the electrical recording of brain activity, represented as voltage fluctuations resulting from ionic current flows within the neurons of the brain [2]. Based on their frequency and voltage, these signals are classified into five separate frequency bands: theta, alpha, beta, gamma and delta waves respectively. These waveforms from the different parts of brain give some indication about a person's mental or physical states [3].

The delta waves fall in the frequency band between 0.5-4 Hz and are the slowest EEG waves with amplitudes ranging between 75-200 μV [4]. They reflect the brain of an unconscious person and are observed during deep sleep in adults. Theta waves have a frequency of 4-8 Hz with amplitude less than 100 μV. They are associated with memory recalling. The most dominant frequency band is the alpha band with its frequency ranging between 8-14 Hz and amplitude less than 50 μV. They are observed during the state of relaxed awareness. Beta waves are associated with increased alertness and active concentration. These waves are observed while executing body movements. The frequencies of beta waves normally range from 14-30 Hz, and their amplitude is normally less than 30 μV. Brain waves with frequency over 30 Hz are classified as Gamma waves. They are observed during multiple sensory processing. They have the lowest amplitude among all, i.e., less than 2 μV. Thus, in general, it can be assumed that the low frequency waves are associated with inactive

state of the brain whereas, the high frequency waves are associated with active information processing.

2.1 EEG traits

The synaptic activation of the brain's neuron creates electric fields, which generates EEG signals. If these signals are obtained in response to visual or emotional stimuli, then they can be classified as behavioural biometrics [1]. To be suitable for authentication purpose, every biometric factor needs to meet a certain set of requirements: universality, distinctiveness, permanence, and collectability [5].

Studies have shown that EEG is compliant with these quality measures [6]. Universality means that every person should posses the trait, and EEG satisfies this requirement as the absence of EEG is a clinical sign of brain death [7]. Distinctiveness requires the biometric to be unique for each person. Permanence requires the trait to stay sufficiently invariant over a period of time. Berkhout and Walter [8] demonstrate the stability and individuality of the EEG signals. The collectability criteria refers to the requirement that it should be easy and comfortable to collect and measure the trait. La Rocca et al. [9] argues that the major limiting factor in the collectablility is the number of electrodes used, as a large array of electrodes is needed to achieve an identification accuracy of more than 90%. However, Armstrong et al. [6] performed biometric identification with only three electrodes and has showed that it is possible to maximize the collectability with a minimum number of electrodes.

2.2 Biometric authentication based on EEG

The authentication factors used in the identity verification process are often some previously known, specific information about the user. There are typically three different kind of authentication factors, which include something that a person knows, something that a person has or something that a person is. EEG based authentication belongs to the third category [10].

The first step in constructing an EEG based authentication is the EEG acquisition. Abo-Zahhad et al. [1] discusses the four categories of EEG acquisition protocols. The first category is recording EEG signals during relaxation with eyes open or closed. The next is visual simulation where the brain activity is recorded during reaction to a visual stimuli. The

Figure 1. Commercial EEG headsets [1]

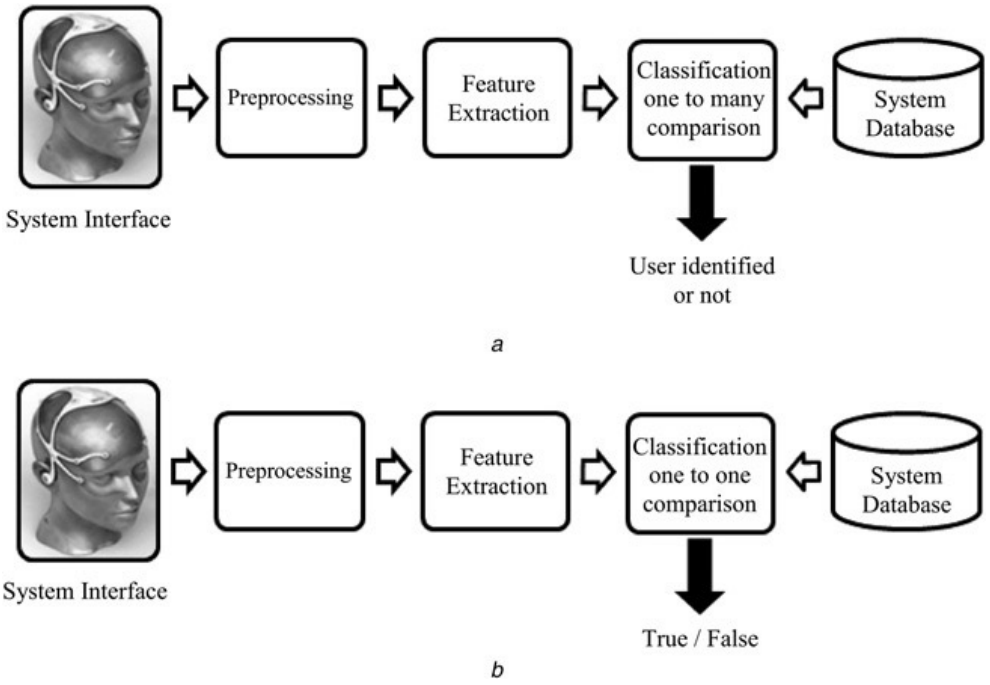


Figure 2. EEG biometric authentication system [1]

third group of acquisition protocol involves performing mental tasks, like, imagination of a body part movement or performing mental mathematical calculations. The last group includes EEG signals based on emotional stimuli where the subjects are asked to focus on the emotions linked to a personal circumstance.

The authentication system records these with the help of EEG headsets. Examples of some commercially available devices for this are shown in Figure 1. These signals are then preprocessed to remove noise and enhance the quality. Features extraction is done from these preprocessed signals, which is then used to train a classifier model. There are two separate modes in EEG based authentication. The first one is identifica-

Biometric	Study	Entropy (bits)
Fingerprint	Li et al. [12]	48
Face	Feng and Yuen [13]	75
Iris	Kanade et al. [14]	94
Retina	Arakala et al. [15]	17
EEG	Bajwa and Dantu [16]	82

Table 1. Entropy of different biometrics

tion mode, which answers the question, who is the user, and the model tries to identify the user's class by receiving their EEG records. The second mode is verification mode where the user claims an identity and the model should grant or reject access based on it. Both modes contains an initial recording phase called *registration phase* during which the model is trained. After this, the users will be able to verify or identify using the system by performing the same task which they performed during the initial recording phase. This signal, which is captured by the EEG headset is then processed and classified by the model for identification or verification. An overview of the processes involved in EEG-based authentication is shown in Figure 2, for both modes.

3 Benefits of using EEG

Brain biometrics potentially have many advantages over current conventional biometrics, such as fingerprints and retinal scans [11]. Studies show that the entropy of EEG is approximated to be around 82 bits, which is far superior than the other commonly used biometrics, such as fingerprint, face, iris and retinal scan. Furthermore, it is important to note that the entropy of human chosen passwords is usually only around 20-22 bits, which makes EEG a better candidate for authentication. Table 1 summarizes the entropy of some of the most commonly used biometrics.

EEG signals are inherently more privacy complaint than other popular biometrics, like, face, iris and fingerprints, since they occur as a result of cerebral activity and therefore are not exposed and cannot be captured from a distance. Biometrics, like, fringerprints are very common these days in devices, such as mobile phones and laptops. These can be easily forged as they can be left on these devices itself and can be recreated from those using various techniques [17]. However, EEG signals are immune

to these kind of spoofing attacks as they are recordings of brain activity in response to a particular task during a certain mental state and are not left on the devices or objects. Moreover, with other biometrics an attacker can use a dead body to authenticate their access to a system, but with EEG biometrics this is not possible as a dead brain would not generate EEG singals. Thus, the liveness detection, which is a major challenge in conventional biometrics, is naturally overcome using EEG.

Another benefit of using EEG for authentication is that it prevents the hackers from forcing the users to authenticate without their consent. With other common biometrics, like, fingerprint or face recognition it is easier for an intruder to make the user authenticate by using force, but with brain biometrics this is not case. If forced, the measured EEG signals would show signs of stress and would not match with the previously recorded EEG signals during relaxed state and thus it will prevent access [4]. All these benefits show that EEG is a viable candidate for biometric authentication as it guarantees that the users are alive and are authenticating by their own will.

4 Open challenges

The above discussions show that EEG is a leading biometric authentication with many advantages over other existing systems. However there are still several open challenges in the field. All the studies were conducted in controlled environments and this is not the case in real life. For example, if users are running to catch a bus then their stress level would be high and in between if they try to unlock their phone which uses EEG based biometric then it would result in denial of access. This is not an ideal or desired behaviour for an authentication system. In this section we will examine some of the main challenges in employing an EEG based biometric system for personal authentication in real life.

4.1 *Universality*

Current studies have mostly been done on limited set of subjects who are healthy and young and this is not sufficient to guarantee universality in real life scenarios for users belonging to different age groups or having other medical conditions [11]. Moreover, almost all the researches have focused on a classifier learning from the recorded EEG signals of a set

of subjects, and then this model is used to identify the users. The major drawback for with this kind of a system is that to add a new user to the system, it will require at least one EEG sample of the new user and the whole model will have to be re-trained with it from scratch [11].

4.2 *Permanence*

Another issue with EEG based biometric is that there is a language dependency on the part of the brain, which is activated with response of linguistic stimuli. Reiterer et al. [18] show that the native language processing in adults mostly involves left hemisphere, whereas foreign language processing is more distributed over both left and right hemispheres. Further, with increasing proficiency in the secondary language, the linguistic processing shifts more towards the left hemisphere. This will result in an inconsistency in the recorded EEG pattern over the scalp over time. Thus, the authors believe that if pass thoughts going to be used are words or songs then it should be in the user's native language.

4.3 *Acceptability*

The success of a practical biometric system depends mainly on the acceptability of the system, i.e., are users willing to use the system. This puts forward certain challenges for EEG based authentication. As EEG signals represent the mental state of a person, it reveals several private information about the individual. A solution for this is to use either encryption or hashing. In case of encryption the user's information will be compromised if the attacker gains access to the encryption key. Furthermore, the conventional hashing methods used for storing passwords cannot be employed in case of EEG based authentication, as the EEG signals recorded over different sessions are never exactly the same for a user. These small changes in input to the hash functions will result in different outputs thus the system would not be able to verify the user's claimed identity [11]. These privacy concerns poses a major barrier towards acceptability of EEG based authentication and further research is needed in this direction to make EEG based biometric systems practical in real life.

5 Discussion

Using a commercial dry-electrode EEG headset, Yang et al. [19] demonstrated the viability of EEG-based authentication in a real-world, outside-of-lab environment. The accuracy levels of the demonstration were lower than those attained in previous studies performed in controlled environments using clinical-grade EEG equipment. This shows that the EEG biometrics requires further improvements on its accuracy before it can be used for practical system implementation.

Despite the many benefits of EEG biometrics, the main barrier to the adoption of these systems is the cumbersome acquisition setup for users, which entails a number of electrodes put on the scalp and typically the use of conductive gel to lower skin impedance. Therefore, reducing the number of electrodes in use is an important problem that should be solved in order to enhance the user experience. However, currently, there are several EEG-based products on the market, primarily for entertainment purposes, that use only a few number of electrodes. These devices are not used for authentication; nevertheless, they serve as a proof of concept for how the number of electrodes can be decreased. Additionally, dry electrodes that do not require conductive gel have lately been made available on the market. These electrodes would reduce the discomfort of the user wearing the headsets. On the contrary, these devices are more expensive. This would increase the overall cost of the system, making it less suitable for commercial use. The main disadvantage of dry devices is that their accuracy is not as precise as other Brain Computer Interface devices used in medical areas. Therefore, it is worth researching that whether portable devices with dry electrodes are appropriate choices for recording EEG data for identifying individuals.

6 Conclusion

This paper reviewed biometric authentication using EEG signals. The brain wave characteristics associated with different frequency bands were discussed in the paper. In addition, the paper highlighted the EEG traits, which make them a suitable candidate for biometric authentication. An overview of the steps involved in EEG-based authentication system have been detailed. Further, the paper summarised the benefits and open challenges in EEG based authentication by reviewing existing studies. Addi-

tionally, the performance of the system in real world, outside of lab environment, was discussed, along with topics for further research.

References

[1] M. Abo-Zahhad, S. M. Ahmed, and S. N. Abbas. State-of-the-art methods and future perspectives for personal recognition based on electroencephalogram signals. *IET Biometrics*, 4(3):179–190, 2015.

[2] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[3] Saeid Sanei. *Adaptive processing of brain signals*. John Wiley & Sons, 2013.

[4] Patrizio Campisi and Daria La Rocca. Brain waves for automatic biometric-based user recognition. *IEEE Transactions on Information Forensics and Security*, 9(5):782–800, 2014.

[5] A.K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.

[6] Blair C. Armstrong, Maria V. Ruiz-Blondet, Negin Khalifian, Kenneth J. Kurtz, Zhanpeng Jin, and Sarah Laszlo. Brainprint: Assessing the uniqueness, collectability, and permanence of a novel method for erp biometrics. *Neurocomputing*, 166:59–67, 2015.

[7] Raafat Hammad Seroor Jadah. Basic electroencephalogram and its common clinical applications in children. In Hideki Nakano, editor, *Electroencephalography*, chapter 8. IntechOpen, Rijeka, 2020.

[8] Jan Berkhout and Donald O. Walter. Temporal stability and individual differences in the human eeg: An analysis of variance of spectral values. *IEEE Transactions on Biomedical Engineering*, BME-15(3):165–168, 1968.

[9] Daria La Rocca, Patrizio Campisi, and Gaetano Scarano. Eeg biometrics for individual recognition in resting state with closed eyes. *Proceedings of the International Conference of the Biometrics Special Interest Group*, 09 2012.

[10] Juris Klonovs, Christoffer Kjeldgaard Petersen, Henning Olesen, and Allan Hammershoj. Id proof on the go: Development of a mobile eeg-based biometric authentication system. *IEEE Vehicular Technology Magazine*, 8(1):81–89, 2013.

[11] Amir Jalaly Bidgoly, Hamed Jalaly Bidgoly, and Zeynab Arezoumand. A survey on methods and challenges in eeg based authentication. *Computers Security*, 93:101788, 2020.

[12] Peng Li, Xin Yang, Hua Qiao, Kai Cao, Eryun Liu, and Jie Tian. An effective biometric cryptosystem combining fingerprints with error correction codes. *Expert Systems with Applications*, 39(7):6562–6574, 2012.

[13] Yi C. Feng and Pong C. Yuen. Binary discriminant analysis for generating binary face template. *IEEE Transactions on Information Forensics and Security*, 7(2):613–624, 2012.

[14] Sanjay Kanade, Danielle Camara, Dijana Petrovska-Delacretaz, and Bernadette Dorizzi. Application of biometrics to obtain high entropy cryptographic keys. *International Journal of Computer and Information Engineering*, 3(3):555 – 559, 2009.

[15] Arathi Arakala, Jason Shane Culpepper, Jason Jeffers, Andrew Turpin, Serdar Boztas, Kathryn Horadam, and A McKendrick. Entropy of the retinal template. *Proceedings of the 3rd IAPR/IEEE International Conference on Biometrics (ICB 2009)*, page 10, 2009.

[16] Garima Bajwa and Ram Dantu. Neurokey: Towards a new paradigm of cancelable biometrics-based key generation using electroencephalograms. *Computers Security*, 62:95–113, 2016.

[17] Ines Goicoechea-Telleria, Ana Garcia-Peral, Anas Husseis, and Raul Sanchez-Reillo. Presentation attack detection evaluation on mobile devices: Simplest approach for capturing and lifting a latent fingerprint. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–5, 2018.

[18] Susanne Reiterer, Ernesto Pereda, and Joydeep Bhattacharya. Measuring second language proficiency with eeg synchronization: how functional cortical networks and hemispheric involvement differ as a function of proficiency level in second language speakers. *Second Language Research*, 25(1):77–106, 2009.

[19] Liuyin Yang, Arno Libert, and Marc M. Van Hulle. Chronic study on brainwave authentication in a real-life setting: An lstm-based bagging approach. *Biosensors*, 11(10), 2021.

A Survey on Data-Driven Animation Techniques

Nami Naziri
nami.naziri@aalto.fi

Tutor: Amin Babadi

Abstract

Characters in video games have complex behaviors. Different types of characters, such as bipedal and quadruple-legged characters, and their interaction with the environment, offer a wide range of possibilities for creating controllers. A variety of behaviors can be displayed by these characters, ranging from different modes of locomotion, such as walking and running, to different actions, such as sitting and carrying objects. In many fields, neural network-based deep learning methods are now widely applied, and character controllers are no exception. A survey of different methods for creating controllers for video game characters is presented in this paper, including five state-of-the-art neural networks for creating controllers for bipedal and quadrupedal characters.

KEYWORDS: Animations, Video Games, character control, deep learning, neural networks

1 Introduction

In video games, the character control system (or controller for short) is an important and critical module, which is responsible for controlling the movement of the player's character. Creating a controller that is both ro-

bust and responsive can be a challenging task for developers. In order for a character to appear natural, it must perform a variety of actions, including different modes of locomotion, avoiding obstacles, and interacting with objects and other characters in the scene. In order to accomplish these actions, a variety of animation clips are needed. Conventionally, animation graphs and state machines are used to develop such controllers. However, since there are many animations to include all the actions, designing such a controller using state machines would result in a very complex animation system that is difficult to maintain and expand.

Clavet [3] introduced motion matching to address the problem of dense state-graphs complexity. In motion matching, at each frame, the best pose of the character is chosen from an unstructured animation database using a scoring system. Motion matching suffers from two major problems. First, it finds the best matching pose by brute-forcing the database at each frame, which grows linearly as the database grows. In this approach, to search the database, the database must exist in memory, which results in high memory consumption. The second problem is that motion matching cannot synthesize new animations since it only uses the poses available in the database. Holden et al. [7] introduced learned motion matching to overcome the problem of memory usage by replacing different components of motion matching with its learned alternative ones. However, the problem of not synthesizing new animations still exists.

This paper aims to introduce the state of the art animation controllers that use supervised machine learning approaches.

This paper first introduces motion matching as it is currently one of the most used frameworks for creating character controls and animations systems in the game industry. Then, we study the state of art of synthesizing animation and character controller by comparing different deep learning approaches for creating character controllers.

This paper is organized as follows. Section 2 describes the related work done using neural networks. Section 3 provides a preliminary introduction to motion matching and neural networks. Section 4 studies five different neural network architectures for creating character controllers. Finally, Section 5 offers concluding remarks.

2 Related work

In recent years, data-driven approaches which have benefited from supervised machine learning models such as neural networks have been used to create character controllers.

Holden et al. [8] used a phase variable which represents the phase of the motion cycle for a humanoid character. This variable is then given to a function called phase function which outputs the weight of the neural network. Zhang et al. [15] proposed a novel neural network called Mode-Adaptive Neural Networks, which produces a new pose for quadruped characters based on previously given frame input. Starke et al. [11] proposed a neural network for character scene interaction , such as sitting, picking up objects, and avoiding obstacles in addition to different types of locomotion. All of these approaches use a global temporal parameter to differentiate between the different phases of the motion of the characters. Starke et al. [12] presented a new framework which uses a local phase for each body part that makes contact with the objects in the environment instead of using a global phase. This approach result in a synthesis of sharp animations for situations that need much contact between the characters or the character and an object. In addition, this framework can be generalized to be used for quadruped characters. Since these approaches use a large amount of data, the iteration time can be slow. By introducing a modular deep learning framework, Starke et al. [13] took a similar approach to learned motion matching [7], but this approach has the advantage of synthesizing new animations.

3 Preliminaries

This section first explains and defines the motion matching algorithm. After that, it introduces the basis of neural networks since they are the main building blocks for deep learning frameworks that are used to create character controllers.

3.1 Motion Matching

Motion matching is an animation selection algorithm that is currently widely used in video games. It was first introduced by Clavet [3] for the

game called "For Honor". Since then, different versions of motion matching have been used in games, such as "The Last of Us Part 2" by Naughty Dog, Inc [10], "Control" by Remedy Entertainment Plc [9], and "Madden" and "FIFA" by Electronic Arts [2].
Clavet defines motion matching as a brute-force animation selection algorithm in which the animation database is searched for the best matching pose in each frame.This algorithm can be more formally defined as follows.

Input
The input of the algorithm is a feature vector consisting of two major components and metadata. The components are the pose of the character in the previous frame and the future trajectory.
These properties can be defined as follows. $X = \{J^p_{i-1}, J^v_{i-1}, J^d_{i-1}, T^p, T^d, M\} \in \mathbb{R}^n$ where $J^p_{i-1} \in \mathbb{R}^{3N}$ are the joints $3D$ position local to the root joint of the character in the previous frame, $J^v_{i-1} \in \mathbb{R}^{3N}$ are the joints $3D$ velocity local to the root joint of the character in the previous frame, and N is the number of joints that will be matched. According to Clavet, the performance can be boosted by choosing a few joints instead of choosing all of the joints of the character's skeleton. For example, using only the feet joints and the hip joint.
$T^p \in \mathbb{R}^{2t}$ are the future $2D$ trajectory positions and $T^d \in \mathbb{R}^{2t}$ are the future $2D$ trajectory directions. The trajectory positions and directions of the character can be calculated using the values from the game-pad control stick. The trajectory of the animation can also be calculated by projecting the hip joint on the ground. Since there may exist some noises in hip joint movement, Holden [6] suggests smoothing out the position using a Savitzky-Golay filter.
M is the metadata used to match different features, such as one-hot encoded vectors for matching specific gates and the local velocity of the character. For example, the position of the sword was one of the metadata features which was used in the game "For Honor" that allowed combo animations.

Output
The output vector can be defined formally as follows. $Y = \{J^t_i, J^r_i, O\}$ where J^t_i and J^r_i are the joint translations and rotations for the current frame, and O is other additional output tasks, such as foot contact information, the position of the objects in the world, and the future position and direction of the trajectory of the selected animation.

Workflow

The workflow suggested by Clavet for using motion matching in the game "For Honor" is as follows.

The motion is captured using dancing cards. Zadziuk [14] proposed dancing cards for capturing actors' movements in motion capture scenes. Dancing cards help animators to capture effectively, meaning capturing as few moves as possible while creating the most coverage possible.

Once the motions are captured, animators tweak and clean them up, label the important parts of them, and import them into the engine.

At runtime, at each frame, the gameplay code makes a query to the animation system by specifying input variables, such as speed, trajectory, and the gate of the character. Based on the query, the animation system then tries to find the best possible matching frame (frame with the lowest cost). Finally, the animation system procedurally adjusts the pose according to the environment, other characters, and the gameplay code using techniques, such as animation warping and inverse kinematics (IK).

3.2 Neural Networks

One of the machine learning algorithms that support supervised learning is neural networks. The idea behind neural networks is to create an algorithm that mimics the operations of an animal brain.

Neural networks are extremely efficient in learning from data. As a result, once the network has learned, it can predict the output based on the provided input.

Gurney [5] has presented a pragmatic, working definition of a neural network: "a neural network is an interconnected assembly of simple processing elements, units or nodes whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns"

In order to understand how neural networks work and what the major components are, can be observed in the following example which was presented in [5].

An example of a simple neural network can be seen in Figure 1. This simple neural network is also known as a single-layer perceptron.

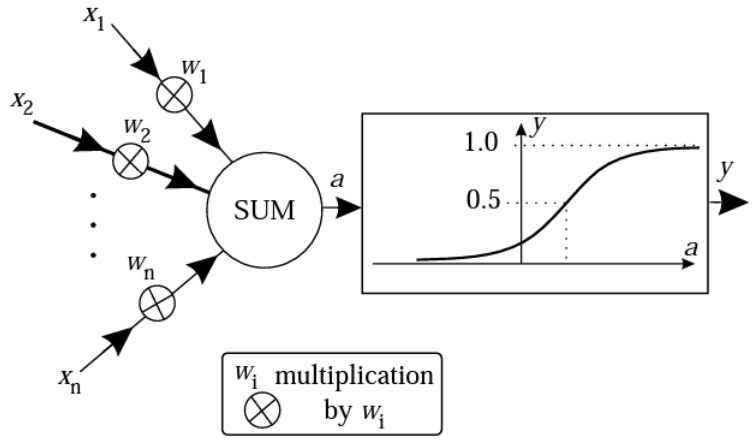


Figure 1. Single-Layer Perceptron

Inputs

The input X is one of the components of the neural networks. The input is the data that is sent to the perceptron for further processing. The input can be actual input data which consists of different features or it can be the output of other perceptrons. In this example we can see that the input $X = x_1, x_2, \ldots, x_n$ has n features.

Weights

The edge between each input and a neuron has a weight. This weight is used to compute the neuron output. Weights indicate the impact of an input x_i on a neuron, and can be both positive and negative.

Activation Variable

The activation variable is the total sum of the input and weights. It is the variable α in the Figure 1 and is calculated as follows.

$$\alpha = \sum_i w_i x_i$$

Activation Function

The activation function is a function that takes α and a bias as input and decides whether this node (neuron) is active or not. The activation function that is used in this example is known as logistic sigmoid.

$$y = \frac{1}{1 + \exp(-(\alpha - \theta))}$$

[5] [1]

4 Neural network based character controllers

In this section, five different neural network architectures for creating character controllers are discussed.

4.1 Phase functioned neural network

Phase functioned neural network is the network structure proposed by Daniel Holden, for creating humanoid character controllers. It uses a three-layer fully connected neural network with an exponential rectified linear(ELU)[4] activation function. This network structure is unique in that it computes its weights using a periodic function called the phase function. This function can be another neural network, Gaussian Process, or other functions. Holden uses cubic Catmull-Rom spline as the phase function. A phase is a scaler value between 0 and 2π that is defined based on the character's foot position and is labeled during data preparation. The locomotion of humans is mostly cyclic, and the phase is determined based on the foot landing on the ground. For example, in the clip, the first left foot land is assigned the value 0, the first right foot land is assigned the value π , and the second left foot land is assigned the value 2π. The phases in between can be calculated by interpolation.

4.2 Mode-Adaptive Neural Networks

In quadrupeds, movement is different from humanoid movement, so it is difficult to distinguish different modes of movement using a single phase variable. Therefore, using the approach in PFNN results in the artifact for quadruped characters.

The network architecture for Mode-Adaptive Neural Networks consists of two different parts. The first part is the motion prediction network. With the help of a few features as input, this network tries to differentiate between different behaviors and actions. The outputs of the network are blend weights used for expert pools. An expert is a neural network weight for the second part of the model and specializes in a single action. Motion prediction is the second part of the algorithm that determines the character's pose in the scene. In this network, the weight is calculated by blending the expert pools.

In this approach, the feet velocity is used as the feature to differentiate between different actions. According to Zhang, this feature yields the best

quality because the velocity of the feet correlates closely with the phase of the locomotion, creating a similar effect to that of the phase function in PFNNs.

4.3 Neural State Machine for Character-Scene Interactions

Neural State Machines provide seamless translation between animations using Goals and the environment surrounding the character. The goals in this architecture can be divided into two categories: high level locomotion and low level locomotion. Choosing an action, such as sitting is an example of a high-level or goal-driven mode. When an action is chosen, the network will generate a series of actions. For example, a sitting action can be divided into three phases: starting to move, moving toward the target, and then sitting. A low-level locomotion mode involves walking and running by the character.

NSM uses the same architecture as MANN [15]. The gating network is responsible for creating distinctions between different goals and actions. The network is designed in such a way that the output selects and interpolates expert weights based on the action labels and phase values. The action labels and phase values are added to the animation clip during the data preparation process. In addition, action labels can be a combination of two different actions. For example, carry and walk.

The phase is the same as PFNN approach, a scaler value between 0 and 2π. For cyclic animations the phase is defined as [8]. In the case of acyclic motions, such as sitting, the phase is determined by the time between transitions. The network produces blending coefficient for the expert pool. Experts are different network weights, each trained to be specialized in a specific goal.

As with the PFNN, the motion prediction network consists of three layers, whose weights are calculated by blending expert pools. The motion prediction module takes the encoded form of the input as its input. There are four components of the input: Frame input, Goal input, Interaction Geometry input, and Environment geometry input. A three-layer neural network encodes these inputs and then feeds them to the motion prediction module.

4.4 Local Motion Phases for Learning Multi-Contact Character Movements

Local Motion Phases aims to extend existing works by accommodating fast interaction between characters and objects. The LMP uses the same architecture as MANN and NSM, which means that the network consists of two different components, a Gating network, and a Motion Prediction Network. This approach differs from others in the features it uses for distinguishing between different actions. Due to the complexity and speed of these interactions, defining a single phase can be challenging. Consequently, rather than using a global phase for different actions as in MANN, it extracts phases locally for individual body parts. As a result, the network is able to learn fast and complex interactions.

The paper also introduces a generative control model. This model is used to convert high-level control signals into finer and sharper movement signals. As a result, more variety of motions can be created using the same input signal. Previous works would have produced the average of the motions in these situations. The paper utilizes an encoder-decoder network. Encoder-decoder networks are primarily used to compress data. Encoder-decoder networks can be created using neural networks. Such architectures have a low number of neurons in a hidden layer called the bottleneck. In this paper, the network is trained based on motion capture data, and then in runtime, it takes the controller input as its input, and noise is added to the bottleneck to produce a different trajectory.

4.5 Neural Animation Layering for Synthesizing Martial Arts Movements

The purpose of Neural Animation Layering is to address three problems that have been identified in previous works. The problems are as follows: 1. Abstract features and not being able to cover all possible motion variations 2. There is a long iteration time because the data needs to be retrained in order to cover new application areas 3. animators do not have control over the animation output.

There are three modules in the model: a control module, an interface, and a motion generator. The motion generator uses a similar architecture as [15, 8, 12]. The network is able to reconstruct animation data with a high degree of accuracy by learning from unstructured motion capture data.

The control module is responsible for learning different behavior. Idling, moving, attacking, and targeting are some examples. Behaviors can be represented by neural networks, motion matching, animation clips, or physics-based animations. In the paper, for example, a neural network similar to [12] is used for locomotion. Each behavior must produce a future trajectory as an output.

Control interfaces allow animators to change animations by using functions, such as overriding, adding, and blending. By using the override operation, different motions can be combined, for example, walking while punching. An example of an additive operation would be the addition or subtraction of height from an action. It is possible to use the blending operation to transition between different behaviors, for example, from idle behavior to walking behavior.

5 Conclusion

This paper examines different methods for creating character controllers for video games. First, state machines and animation graphs were introduced as traditional methods of creating character controllers, as well as their limitations. After that, motion matching was introduced, which is a method for overcoming the problem of complex animation systems by querying the animation database to find the closest pose of the character. Following this, five neural network-based character controllers were presented and discussed.

The PFNN [8] approach is used to create the locomotion of bipedal characters. Using a global phase variable, it is possible to distinguish biped locomotion, since it is always characterized by left foot landing, right foot landing, and again left foot landing.

Since quadruped locomotion does not follow the same pattern as bipeds, the PFNN approach cannot be used for quadruped characters. Architecture defined in MANN [15] can be used for quadruped character locomotion. This approach uses the velocity of the quadruped feet to differentiate between the phases of the movement. The system utilizes a gating network and motion prediction modules. Different types of motion are distinguished using the gating network. And motion prediction is responsible for creating the character's pose.

MANN and PFNN do not provide functionality for interacting with the environment. NSM [11] extends previous studies to include character

scene interaction. The network architecture is the same as that of MANN. Instead of using velocity as a feature, it defines a phase for each action.

LMP [12] improves the previous study by introducing a local phase for each body part. As a result, it can be used for fast-paced movement and character-to-character interaction.

Finally, NAL [13] provides an animation framework that enables animators to change the output of animations with a minimal amount of iteration. Animators can use different methods to alter different animations, combine animations together or translate between different behaviors.

References

[1] Single layer perceptron in tensorflow. https://www.javatpoint.com/single-layer-perceptron-in-tensorflow. Accessed: 2022-10-24.

[2] Henry Allen. Animation summit: Environmental and motion matched interactions; 'madden', 'fifa' and beyond! In *Proc. of GDC*, volume 2021, 2021.

[3] Simon Clavet. Motion matching and the road to next-gen animation. In *Proc. of GDC*, volume 2016, 2016.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[5] Kevin N. Gurney. Neural networks for perceptual processing: from simulation tools to theories. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 362 1479:339–53, 2007.

[6] Daniel Holden. Code vs data driven displacement. https://theorangeduck.com/page/code-vs-data-driven-displacement, 2021. Accessed: 2022-10-23.

[7] Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. Learned motion matching. *ACM Trans. Graph.*, 39(4), jul 2020.

[8] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4), jul 2017.

[9] Ilkka Kuusela and Ville Ruusutie. Animation summit: Take 'control' of animation. In *Proc. of GDC*, volume 2021, 2021.

[10] Michal Mach and Maksym Zhuravlov. Motion matching in 'the last of us part ii'. In *Proc. of GDC*, volume 2021, 2021.

[11] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6), nov 2019.

[12] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM Trans. Graph.*, 39(4), jul 2020.

[13] Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. Neural animation layering for synthesizing martial arts movements. *ACM Trans. Graph.*, 40(4), jul 2021.

[14] Kristjan Zadziuk. Animation bootcamp: Motion matching: The future of games animation...today. In *Proc. of GDC*, volume 2016, 2016.

[15] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.*, 37(4), jul 2018.

Interactive Global Illumination Algorithms based on Ray Tracing

Nicholas Saarela
nicholas.saarela@aalto.fi

Tutor: Jaakko Lehtinen

Abstract

With the recent influx of research papers on real-time ray tracing algorithms for global illumination, along with supporting hardware and software solutions, one could argue that the field is rapidly evolving. Researchers and developers new to computer graphics may find themselves overwhelmed by the amount of available research material and struggle to find a starting point upon which to build their knowledge in the area.

This paper aims to serve as an easily approachable overview of ray tracing research for interactive global illumination. It takes a bottom-up approach to build prequisite knowledge in underlying computer graphics principles before discussing more advanced material. This work provides up-to-date references for the reader to deepen their understanding of the discussed content and to find the most recent and authoritative research papers in the field.

KEYWORDS: Computer graphics, Global illumination, Real-time rendering, Ray tracing, Path tracing

1 Introduction

Interactive computer graphics, such as used in 3D game engines, often aims to produce photorealistic and immersive experiences for the user. In *interactive*, or *real-time*, computer graphics an image is rendered to the screen multiple times in a second [17, p. 1]. Between each image, also called a *frame*, the user may interact and the next rendered image is then affected by the user's action. This loop must cycle in such a rapid way that the user does not see individual images but rather experiences this as if it's happening in real-time.

Modern movies leverage a rendering technique called *ray tracing* to simulate realistic reflections, refractions and shadows, producing images that can be indistinguishable from real life [4]. Such realistic lighting effects are generally referred to as *global illumination (GI)* [3]. GI includes not only direct illumination, where light from a light source hits the rendered surface directly, but also indirect illumination, where light bounces from surrounding surfaces before hitting the surface.

Global illumination through ray tracing algorithms has historically been too demanding for computing hardware to be feasible for real-time applications [4]. Therefore developers of real-time computer graphics have long had to resort to another faster, although more limited, rendering technique called *rasterization*. In recent years though, ray tracing hardware such as the NVIDIA® RTX technology [9] has made rendering techniques based on ray tracing increasingly feasible for real-time applications [10]. Both ray tracing and rasterization will be discussed in more detail in section 2.1.

This paper reviews the current state of real-time global illumination research based on ray tracing and highlights some important challenges that will still have to be addressed by future works in the field.

This paper is organized as follows. Section 2 serves as a technological primer to help the reader understand more advanced concepts in the following sections. Section 3 surveys the most recent research conducted within the field of real-time ray tracing. Section 4 explores some of the unsolved challenges within the field as well as potential for future research. Finally, section 5 concludes with a summary.

2 Computer graphics primer

This section aims to serve as a primer in computer graphics. Starting with a discussion of the two major types of rendering techniques - rasterization and ray tracing - the following subsections review prequisites necessary to understand current real-time ray tracing research.

2.1 Ray tracing versus rasterization

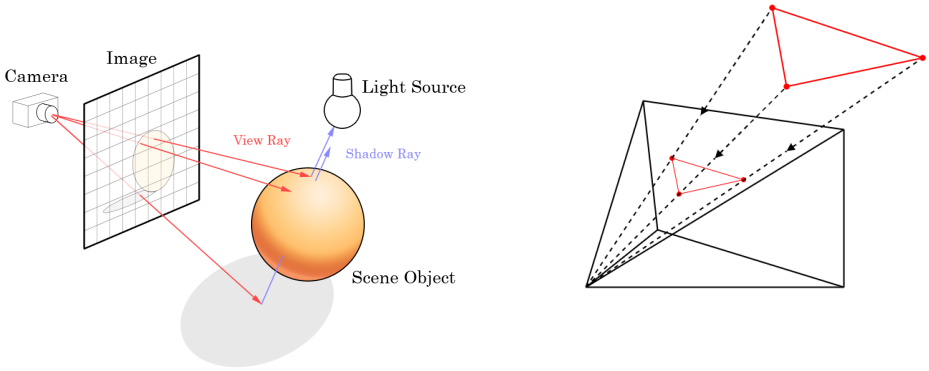
Ray tracing algorithms for offline, or non-interactive, rendering have been a subject of computer graphics research for over 40 years [18]. Ray tracing was formulated by Brian Caulfield from NVIDIA [4] as follows: "The easiest way to think of ray tracing is to look around you, right now. The objects you're seeing are illuminated by beams of light. Now turn that around and follow the path of those beams backwards from your eye to the objects that light interacts with. That's ray tracing."

The ray tracing algorithm works by tracing a ray of light through each pixel on the 2D viewing surface out into the 3D space [4]. By following this ray as it bounces from objects in the 3D scene by reflection and refraction, ray tracing captures reflections, shadows and refractions. As the ray travels, every time it hits a surface the color and lighting information at the point of impact contributes to the originating pixel's color and illumination level. By working this way, ray tracing can also simulate a variety of common techniques seen in movies - such as motion blur, depth of field and translucency. It is a rather simple algorithm yet computationally very demanding. Ray tracing is illustrated in figure 1a.

As mentioned before, the rasterization technique has long been used in real-time computer graphics due to its speed [4]. Consequently, using this technique, the *graphics processing units* or *GPU*'s in today's computers are able to render millions of polygons to a 4K display typically 30 to 90 times each second. Years of development into GPU's and rasterization techniques have led to good rendering results, although still inferior compared to ray tracing.

When using rasterization, the objects in the scene to be rendered consist of a virtual mesh of triangles. These triangles are further formed from their virtual apexes known as *vertices* [4]. There is important information associated with each vertex, such as its position in space, its normal vector, its color and texture information. This information can be used when converting the 3D triangles into pixels on the screen. The rasteri-

zation algorithm goes through every triangle in the 3D scene and checks which pixels on the screen the triangle covers [4]. Each pixel can then be assigned an initial color value based on the data stored in the triangle's vertices. Further pixel processing, also known as *shading*, then changes the color of the pixel based on how lights in the scene interact with it and applying textures if needed before determining the final color of the pixel. Rasterization is illustrated in figure 1b.



(a) Ray tracing [16] **(b)** Rasterization

Figure 1

Despite its efficiency, rasterization does have its limitations. DICE Technical Director Christian Holmquist [1] described rasterization as making it challenging for objects to interact with each other, as in order for each triangle to correctly draw itself, each triangle needs to know about a scene in its entirety. Therefore, developers of interactive computer graphics have traditionally had to resort to various heuristic techniques in order to produce realistic lighting effects. Such techniques include shadow mapping, ambient occlusion and image-based lighting [7].

2.2 The rendering equation

In his 1986 paper *The Rendering Equation* [8] James T. Kajiya published the equation of the same name that accurately models the scattering of light off various types of surfaces. His work expresses rendering in terms of *radiance transfer*, binding computer graphics more concretely to physics than previous approaches. The rendering equation is also known as the *light transport equation (LTE)* [11, p. 861]. The original form of the rendering equation is defined as follows.

$$I(z, z') = g(x, x')\Big[\epsilon(x, x') + \int_S \rho(x, x', x'')I(x', x'')dz''\Big] \tag{1}$$

where:
$I(x, x')$ is related to the intensity of light passing from point x' to point x
$g(x, x')$ is a "geometry term"
$\epsilon(x, x')$ is related to the intensity of emitted light from x' to x
$\rho(x, x', x'')$ is related to the intensity of light scattered from x" to x by a patch of surface at z'.
S is the union of all surfaces in the scene.

As can be seen from (1), the rendering equation is recursively defined and the domain of the integral is infinite-dimentional. Therefore it cannot be solved in closed form and some sort of approximation has to be used for solving the equation numerically. Randomized algorithms suitable for this task will be the topic of the following subsection.

2.3 Monte Carlo integration and path tracing

As discussed in section 2.2, some way of approximating the integral in the rendering equation is needed. *Monte Carlo integration* uses random sampling to estimate the values of integrals and is often used for this task [11, Ch. 13]. Monte Carlo integration gives different results depending on random choices made during the process. However, by averaging the results of multiple runs of the algorithm on the same input, the result eventually converges to the correct answer.

Some of the advantages of Monte Carlo over other numerical integration methods is that it can be used to estimate integrals over domains of arbitrarily high dimension and only requires that one has the ability to evaluate the integrand $f(x)$ at arbitrary points in the domain [11, Ch. 13]. Its main disadvantage is the *variance*, or *noise*, in the output resulting from the random nature of the algorithm.

Along with his famous rendering equation [8], Kajiya also introduced a technique called *path tracing* as one of the first general-purpose *unbiased* Monte Carlo light transport algorithm as a solution to (1). *Bias* as discussed here refers to a typically small systematic error produced by the estimator, such that the result never truly converges to the correct answer.

Path tracing generates paths of light scattering events starting from the camera and terminating at a light source in the scene and does so in an incremental manner [11, Ch. 14.5]. The paths, or rays, originating from a pixel on the screen are traced as they hit surfaces in the scene. At the point of intersection the integral in equation (1) is estimated. Direct illu-

mination from light sources is accounted for and a *scattering distribution function* specific to the surface properties is evaluated. Then a new ray direction for the scattered light must be randomly sampled in order to continue the light path further into the scene. This path continues bouncing off of surfaces until a predefined termination condition is met. Finally, the accumulated radiance along the path determines the color of the originating pixel.

Path tracing can be thought of as an extension to the traditional ray tracing algorithm proposed by Whitted [18] in 1980 [11, Ch. 14.5]. It is a unified rendering algorithm, meaning that it accounts for all different types of light transport and thus removes the need for separate handling of any global illumination effects [5].

As path tracing is based on Monte Carlo integration, it suffers from inaccuracies of the estimation. These inaccuracies result in noise in the rendered image. In order to produce a noise-free image, typically hundreds or thousands of *samples per pixel (spp)* may be necessary [11, Ch. 14.5]. The speed of convergence to the correct result is greatly affected by the choice of sampling methods. Therefore a lot of emphasis has been put into the development of better sampling techniques within computer graphics research. Importance sampling is one of such techniques and will be discussed in section 2.4.

Due to the the limited amount of computation that can be allocated to each frame, path tracing for real-time applications can typically only afford one sample per pixel at a maximum [5]. Therefore the resulting images will have significant noise unless sampling techniques specifically designed for real-time path tracing are used in addition to proper *denoising*. Denoising techniques will be discussed in section 2.6 and recent research in sampling methods for real-time path tracing will be explored in section 3.

2.4 Importance sampling

Importance sampling is one of the most used variance reduction techniques in rendering [11, Ch. 13.10]. Importance sampling is based on the observation that samples taken from a distribution similar to the function in the integrand result in faster convergence of the Monte Carlo estimator. Its main idea is to concentrate sampling where the value of the integrand is relatively high.

As an example, when a ray cast from the camera hits an object, an out-

going ray direction has to be chosen. The direction is chosen randomly by sampling a so called *probability density function (PDF)*. If the PDF is chosen such that it is similar to the scattering distribution function of the surface, it is more likely that a light path along the chosen direction will carry radiance, than if the sample is drawn from a uniform distribution. On the other hand, if the PDF is chosen poorly, the results can be much worse than by simply using a uniform distribution. In practice though, finding good distributions for importance sampling is not too hard for many integration problems in rendering.

2.5 Resampling

Importance sampling can be very useful for real-time applications as it reduces variance at low sample counts [10]. However, this can become a challenge when sampling from optimal distributions is impossible, as is the case with complex lighting. Instead, *resampling* algorithms based on Talbot et al.'s [15] *resampled importance sampling (RIS)* can render complex lighting with only few samples per pixel.

RIS is a technique that uses a two-pass algorithm to sample from distributions that cannot be sampled directly [14]. Recent resampling algorithms exploit sample reuse within and across frames to continually evolve a population of samples towards their optimal distribution [10]. Some of the most recent iterations of such resampling algorithms for global illumination will be discussed in section 3.

2.6 Denoising

Due to the low sample count available for real-time ray tracing, there can be significant noise in the resulting frames. Therefore, denoising is necessary to achieve good results with low sample counts [10]. A denoising algorithm, or *denoiser*, first takes a noisy path tracing output, or *signal*, and decomposes it into frequencies [6]. These frequencies can for example be the diffuse, specular and transmission components of the signal. The denoiser then processed these frequencies in their own spatial and temporal, or *spatiotemporal*, kernels before composing them back together to produce a noise-free signal. The denoiser is guided by guide buffers produced by the path tracer. These mostly noise-free guide buffers hold data such as normal, position and transmission distance information.

NVIDIA Real-Time Denoisers (NRD) [12] is one of such denoising solu-

tions. It is a spatiotemporal denoising library engineered to work with signals with one ray or less per pixel.

3 Recent research

Many of the recent studies in the field of interactive global illumination algorithms are based on RIS. This section will explore some of the most influential of these works.

3.1 ReSTIR and RTXDI

In 2020 Bitterli et al. [2] introduced their *Reservoir-based Spatio- Temporal Importance Resampling (ReSTIR)* algorithm - a Monte Carlo approach for rendering direct lighting with thousands to millions of dynamic light sources in real-time. ReSTIR is based on a generalization of RIS and allows unbiased spatiotemporal reuse of nearby samples while also providing a more efficient biased variant. It works by iteratively applying RIS using weighted *reservoir sampling*. Reservoir sampling is a method using a special sampling algorithm in conjunction with a small fixed-size data structure, called a "reservoir", that stores accepted samples. It enables a high-performance GPU implementation and helps to achieve stable, real-time performance.

The following year Wyman et al. [19] introduced algorithmic improvements to ReSTIR that led to the development of the NVIDIA RTX Direct Illumination (RTXDI) [13] technology. These improvements included reducing lighting costs by up to a factor of seven, improving memory coherence, shrinking the required ray budget and increasing rendering quality, among others.

3.2 ReSTIR GI

While the previously discussed ReSTIR algorithm and its commercial implementation RTXDI enables unbiased real-time rendering of direct lighting, they still leave indirect lighting unaccounted for. Consequently, in their 2021 paper, Ouyang et al. [14] propose path sampling algorithm building on the screen-space spatiotemporal resampling principles of ReSTIR to remedy this. This reservoir-based *ReSTIR GI* -algorithm, suitable to highly parallel GPU architectures, resamples indirect light paths produced by path tracing.

The writers show that ReSTIR GI achieves mean-squared-error (MSE) improvements up to 166x compared to ordinary path tracing while rendering at 1 spp every frame. Furthermore, when used together with a denoiser, it can produce path traced GI at interactive frame rates even with complex scenes. ReSTIR GI, like the original ReSTIR, comes both in biased and unbiased variants. The former can be used to trade off bias for improved rendering performance.

3.3 GRIS and ReSTIR PT

While providing exceptional performance improvements over traditional path tracing, the previously introduced ReSTIR algorithms and their underlying RIS theory make various assumptions, such as sample independence [10]. As interative sample reuse introduces correlation, violating this independence, ReSTIR has the tendency to invalidate most convergence guarantees RIS theoretically provide. More specifically, RIS assumes *independent and identically distributed (i.i.d)* samples, typically from a single source distribution. Sample reuse in ReSTIR violates this dependence slowing convergence or even causing divergence.

To remedy these issues, Lin et al. [10] introduce *generalized resampled importance sampling (GRIS)* that lifts the i.i.d. assumption of the original RIS theory. The authors state that some of the previous work such as ReSTIR and ReSTIR GI are special cases of this new GRIS theory. Using GRIS, resampling can be applied to correlated candidate samples with unknown PDF's, taken different domains. This allows for deriving variance bounds and convergence conditions in ReSTIR based samplers.

Building on GRIS, the authors reformulate the spatiotemporal reuse of ReSTIR to remain consistent and unbiased even for long and complex light paths and present *ReSTIR path tracing (ReSTIR PT)*, an unbiased, path traced resampler capable of running interactively on complex scenes. It can capture many-bounce diffuse and specular lighting while rendering only at 1 spp. ReSTIR PT improves on ReSTIR GI with better robustness from a consistent and unbiased algorithm as well as reduced noise.

4 Future research

This section explores some of the open problems demanding more work, as presented by Clarberg et al. [6] in their keynote presentation at the

High-Performance Graphics 2022 virtual conference.

4.1 More robust sampling

Clarberg [6] explains that content authored for real-time rendering is often optimized for raster engines and can therefore be ill-suited for path tracing. Consequently, more robust sampling methods are needed that are also able to handle poor content, such as occluded lightsources and poor use of emissive features. In addition, tools need to be developed for finding what is causing rendering issues with path tracing.

Clarberg also states that more robust sampling is needed for difficult or longer light paths, such as when rendering long blonde hair, caustics and snow on ice [6]. However, ReSTIR PT [10] was only released a year after the presentation and was not discussed therein. Perhaps ReSTIR PT could provide a part of the solution to these problems.

4.2 Better denoising

Sampling and denoising are intrinsically tied together and are both essential for real-time path tracing [6]. The denoiser expects noise-free guide buffers as input from the path tracer, but in reality guide data can sometimes be noisy, such as when rendering dense geometry, hair and fur, volumes and particle effects. In these cases the denoiser fails to denoise properly.

One possible solution could be to co-design sampling and denoising, forming a feedback loop between the denoiser and the path tracer [6]. With this feedback loop the denoiser could tell the path tracer to improve sampling where the denoiser lacks information.

5 Conclusion

This paper has provided a broad overview of ray tracing research for interactive global illumination. Starting from the basics of computer graphics, such as rasterization, ray tracing and path tracing, it has introduced the reader to increasingly more advanced concepts. After exploring some of the most recent research works in the field, such as the reservoir-based ReSTIR algorithms, this paper finished with a discussion of some yet unresolved problems within real-time ray tracing and path tracing research.

References

[1] Electronic Arts. Battlefield V: Real-Time Ray Tracing. https://www.ea.com/news/battlefield-5-real-time-ray-tracing?setLocale=en-us. Accessed: 24.10.2022.

[2] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *acm trans. graph.*, 39(4), July 2020.

[3] Wayne E. Carlson. *Computer Graphics and Computer Animation: A Retrospective Overview*. The Ohio State University, 2017.

[4] Brian Caulfield. What's the Difference Between Ray Tracing and Rasterization? https://blogs.nvidia.com/blog/2018/03/19/whats-difference-between-ray-tracing-rasterization/, 2018. Accessed 19.10.2022.

[5] Petrik Clarberg, Simon Kallweit, Craig Kolb, Pawel Kozlowski, Yong He, Lifan Wu, and Edward Liu. Research Advances Toward Real-Time Path Tracing. Game Developers Conference (GDC), March 2022.

[6] Petrik Clarberg, Simon Kallweit, Craig Kolb, Pawel Kozlowski, Yong He, Lifan Wu, Edward Liu, Benedikt Bitterli, and Matt Pharr. Real-Time Path Tracing and Beyond. HPG 2022 Keynote, July 2022.

[7] Randima Fernando et al. *GPU gems: programming techniques, tips, and tricks for real-time graphics*, volume 590. Addison-Wesley Reading, 2004.

[8] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, page 143–150, New York, NY, USA, 1986. Association for Computing Machinery.

[9] Emmett Kilgariff, Henry Moreton, Nick Stam, and Brandon Bell. NVIDIA Turing Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth, 2018. Accessed 28.09.2022.

[10] Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. Generalized resampled importance sampling: Foundations of restir. *ACM Trans. Graph.*, 41(4), jul 2022.

[11] Greg Humphreys Matt Pharr, Wenzel Jakob. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, third edition edition, 2016.

[12] NVIDIA. NVIDIA Real-Time Denoisers (NRD). https://developer.nvidia.com/rtx/ray-tracing/rt-denoisers. Accessed: 16.11.2022.

[13] NVIDIA. RTX Direct Illumination (RTXDI). https://developer.nvidia.com/rtx/ray-tracing/rtxdi. Accessed: 18.11.2022.

[14] Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and J. Pantaleoni. Restir gi: Path resampling for real-time path tracing. *Computer Graphics Forum*, 40(8):17–29, 2021.

[15] Justin Talbot, David Cline, and Parris Egbert. Importance Resampling for Global Illumination. In Kavita Bala and Philip Dutre, editors, *Eurographics Symposium on Rendering (2005)*. The Eurographics Association, 2005.

[16] Timrb. Ray trace diagram. https://commons.wikimedia.org/wiki/File:Ray_trace_diagram.png, 2008. This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/3.0/deed.en.

[17] Naty Hoffman Tomas Akenine-Möller, Eric Haines. *Real-Time Rendering*. CRC Press, 3rd edition edition, 2008.

[18] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, jun 1980.

[19] Chris Wyman and Alexey Panteleev. Rearchitecting spatiotemporal resampling for production. In *Proceedings of the Conference on High-Performance Graphics*, HPG '21, page 23–41, Goslar, DEU, 2021. Eurographics Association.

Security Issues in Service Discovery for Fog, Edge and Web

Oskari Järvinen
oskari.jarvinen@aalto.fi

Tutor: Petri Mähönen

Abstract

In the ever increasing distributed computing ecosystem, service discovery has an even more important role. Many solutions have been proposed to provide a solution for the challenges of a distributed, heterogeneous environment. While these solutions are able to provide a structure to provide these services, security has been left unattended. Currently, proper solutions do not exists. In this paper, the aim is to highlight security and other issues in some of the proposed solutions, to highlight the areas that require more research and solutions.

KEYWORDS: Edge, Fog, Cloud, Security, Service Discovery, IOTA, DNS-SD

1 Introduction

The rise of IoT devices has caused the need to be able to process and filter data on a large scale with faster speeds. This data can be used to train more efficient ML models or to act based on sensor data. This has caused the rise of Edge Computing and Fog Computing, which are meant to satisfy these requirements. However,in these models nodes can communicate with each other without being in the same LAN network while generating and processing different amounts of data. Therefore, *Web Service Discover* is required for Edge and Fog to operate properly. To sum up, these characteristics generate a complex system with a lot of heterogeneity[2][4]. Due to these characteristics, the traditional WSD with a broker and naive text search, such as Web Service Description Language, is not sufficient on its own due to the increased heterogeneity and scale of Edge and Fog systems. Either traditional WSD discovery messages are transformed through a proxy[4] to be more dynamic, web services are classified to ease web service discovery[2] or unique frameworks[13] have been developed to tackle this issue. Each of these solutions have their benefits and drawbacks ranging from regex processing to additional overhead. Additionally, each of these proposed solutions generally consist of architectural, security and implementation issues that are discussed in this paper to highlight future research topics. Furthermore, some real life examples are provided of security issues affecting some of these solutions.

2 Web Service Discovery

2.1 Traditional Web Service Discovery

Traditionally, A Semantic Web Service discover system consists of seven key features: service advertisement, service mediation, service storage, service request, service matchmaking, service negotiation, and service selection [10]. However, these can vary based on demands and needs of a system. Every SWS method consists of some kind of formalism, such as a common framework or language, on which the discovery is based on[10]. For example, WSDL which uses XML format to describe a web service and details for connecting to it. However, this method is resource consuming and thus the low-power computational resources of Edge computation are not sufficient in all scenarios, thus increasing latency and lowering qual-

ity of service. Furthermore, another commonly used method is to have a centralized registry of all web services which acts as a broker, such as UDDI[11]. This kind of approach is also used in cloud based service discovery such as Kubernetes. However, this approach is not favorable on smaller scale networks and cause a strain on load balancers. To combat this, methods for multicasting such as WS-discovery[4], which can ease the load from brokers. However, these methods are susceptible for malicious usage and can be used to generate massive DDoS attacks[9]. Additionally, these probings are CPU-intensive, because they require multiple HELLO and PROBE messages to be sent which is a limiting factor for edge computing.

2.2 Edge and Fog Optimized Web Service Discovery

As discussed in section 1.1, traditional naive document based web services are not enough in a low-power distributed environment. Therefore, some kind of classification is required to provide better ways to connect services accurately and swiftly. Currently, two ways of classifications are researched actively[2]. First method is based on data mining text while the secondary method is based on semantic annotations[2]. The text based classification can lead to inaccurate service discovery which can be used maliciously to route traffic to mimicking services while the annotation based requires a large adaptation of standards and policies, such as Web 3.0 based OWL-s. However, these classification methods are able to provide more accurate definition of the available web services. Developments on the traditional method of containing a service registry have been proposed, technology where the blockchain contains the service registry and the blockchain is passed to each node has been suggested as a way to contain a service registry without having a centralized database for it[13].

2.3 Cloud-based Classification Methodology

This method proposed by Alshafaey et al.[2] contains a classification method where there are three modules: A Concepts Preparation Module (CPM), a Tree Creation Module (TCM) and a Change, Edit, Add module (CEAM). The CPM and the TCM module are located in cloud and are responsible for the tree structure creation process. The CEAM is located in fog and its main task is to calculate and create the lowest cost for a web service to be discovered. Together these modules are able to create, modify and store a

tree based classification structure which is then stored in cloud. The priority of this model is to reduce dimensional space, take into consideration the semantic relationships in web services[2]. This model relies heavily on the cloud based networks and the storage capability in cloud services. The overarching concept of having a centralized database that holds all the web services is not a revolutionary concept and is used in current cloud solutions such as Kubernetes and AWS Cloud Map. However, this proposed model is more precise and accurate [2]. However, issues with this models are related to time response, since a query has to be send to cloud resources and data storage, since with billions of devices the tree structure is large and hard to store without scalability. Additionally, the CEAM is the crux of this methodology, since it is responsible for managing the tree structure. Making sure it is scalable, transparent and manageable could be difficult when the tree structure becomes sufficiently large.

2.4 Choreographic Approach

A method proposed by Blanc et al.[4] suggests a method based on choreography collaborative paradigm which relies on RegEx patterns to be able to handle multitude of different sensors and devices in a edge network. As in choreographic model, this model focuses on the messages passed between different processes by creating a choreographic engine inside every device in the local network through which provisions are passed to a message broker which contains a dynamic list that is used in service discovery[4]. Additionally, a proxy is created to be able to communicate with devices outside LAN through WS-discovery. In this approach, hot plugging is made possible and consumer cannot know services before connecting. However, inevitable misconfigurations in the service discover proxy could lead to possible security issues and could be used in malicious ways. Overall, this method takes advantage of resources available on the devices to do service discovery that can help to distribute the load. However, CPU utilization is on the edge which might be an issue due to the limited capability of processing power.

2.5 DNS-SD

Another approach is based on the DNS infrastructure we currently have in our DNS[5][12]. Service discovery through DNS requires new records to be added to DNS records, DNS SRV and DNS TXT[5]. By using these

formats, a service can be searched through DNS queries. However, consequently traffic to DNS providers would increase massively due to the nature of fog and edge computing. Additionally, this approach could be problematic due to the scalability of these DNS services since the need of DNS providers having a service close to the fog nodes forces the DNS providers to operate and deploy more DNS servers. This subsequently also provides more serious attack opportunites by using DNS[1], especially through DNS amplification attacks. Therefore, this implementation has security and scalability issues. Additionally, this style of solution is not the most optimized on resource usage, since it lacks discovery through context which can cause latency when two fog nodes are trying to find each other. However, two methods to combat this have been developed[12]. One method is to use geographic location based context but it lacks accuracy and flexibility[12]. However, currently more and more geographically locked cloud environments are being tested and deployed. Thus global accuracy might not be needed in all scenarios. Another solution is to use the DNS TXT to serve key-value pairs a client can request when searching for a web service. However, this can cause the TXT queries messages size to be an issue[12] which can cause increased response time due higher processing requirements.

2.6 Distributed Hash Table

Another proposed solution by Cirani et al.[6] archetype is to provide a DHT based architecture to provide an Peer-to-Peer solution. This way the whole systems capabilities increase when more nodes or devices are increased. The solution provided by Cirani et al. is automated and no additional configuration is needed by the end user. The system uses an technology called DLS. The DLS a DHT-based architecture, which provides a name resolution based on storage and retrieval of bindings between a URI[7]. DLS provides a better way to provide service compared to DNS since it applies over the URI, not only over FQDN and it has lower propagatation times[6]. Howver, P2P is extremly vulnerable to many different attack vectors. Sybil, Eclipse and Pollution attacks are common attack types on current P2P networks[14]. A Sybil attack interferes with information retrieval by using fake identifiers. An Eclipse attack targets the routing table and poisoning the data. This way the attacker can take over the whole traffic. A Pollution attacks inserts a large amount of invalid information into the index to poison the data to prevent the users from

finding the right resource. These attack types create difficulties on DHT type solutions and require methods to address these attack types. However, P2P has a history of providing stable services in distributed way, such as Skype.

2.7 An IOTA based approach

An cryptocurrency based method proposed by Tsung-Yi et al.[13] uses the IOTA tokens to provide a distributed service discovery database in fog. This reduces the amount of data that is send and processed by cloud providers which is becoming an issue in IoT and mobile businesses. Additionally, this approach reduces latency between services but at the same time complexity and decentralization is reduce. However, this method tries to combat this issue by having a IOTA based blockchain which contains the service discovery database. This means no external databases are held in cloud and no latency in communications when finding a service. However, traditional problems that exists in blockchain technologies also exists in this approach: Are malicious entities able to gain control of the blockchain and is proof how ownership is provided. Additionally, the IOTA blockchain is derived from mIOTA that is managed by the IOTA Foundation. If this foundations wallets secrets are leaked or cracked, the whole system is compromised and large downtimes can be expected, such as the 2020 security incident[8]. Additionally, if the node that is used for every transaction has an outage, the whole service is unusable. This can lead to the whole currency not be able to be used. AS a positive note, IOTA is currently being developed to provide resilience against cyberattacks such as Cybil and Eclipse[13] which means it can be more resistant compared to other distributed table approaches while providing the benefits of these solutions.

3 Discussion

A multitude of solutions exist and more are constantly being developed. As it is, service discovery in Fog and Edge is a well understood problem, but no best solution exists yet. The discussed solutions have their positives and negatives. Thus, it is likely that some kind of a hybrid of solution might arise, especially own solutions to local service discovery and global service discovery, since service discovery inside a Fog node could be im-

plemented differently than on the global network and these environments behave differently. Additionally, the trend of proprietary service discover has been gaining momentum in recent years, especially with containers and cloud infrastructure. This means that current solutions are not transparent and their implementations are not clear to the end user. Furthermore, many of these solutions could be used injunction to provide a better overall solution and thus further research. As it is, further security research is needed since traditionally IoT devices are not considered to be secure, and while positive trend regarding the state of IoT security can be seen, the appliance of this mechanisms is lacking due to the complexities of these distributed and heterogeneous environments[3]. And while these devices are being secured, it is utmost importance to secure the service discovering mechanisms as well. When device amount is scaled to billions of devices, even small percentage of these devices can cause massive security issues. Furthermore, currently many of the largest DDoS attacks have been using IoT devices to provide massive amounts of traffic. To add, with proper efficient service discovery also more devices could be infected if an device can be taken over. This means that while fast and reliable service discovery is necessary, it is also important that the benefits of it cannot be used to amplify security breaches. Additionally, managing an infrastructure that is owned by many different entities, such as DNS providers, is a hard and cause more opportunities for misconfigurations that can create security issues. This means that the centralized solutions can provide edge, since end user has to interact with less amount of service providers. However, this also means that the service provider has to be able to be trusted, meaning transparency is key. Additionally, these centralized solutions tend to be proprietary, thus the end user has no way to verify that the systems are secure.

4 Conclusion

The future of Edge, Fog and Cloud service discovery is uncertain. The current solutions are providing different interesting ways to tackle the problem of service discovery in distributed and heterogeneous environment. And while these solutions provide a way to discover services in a heterogeneous environment, the nature of heterogeneous plug and play environments is not fruitful for security side. Further research is needed to provide more robust, secure and transparent service discovery. Fur-

thermore, current energy crisis and shortage of computer hardware has risen as a important geopolitical security issues meaning these systems should be energy efficient and not require multitude of geographically located data centers that scale on hardware.

References

[1] Kamal Alieyan, Mohammed M. Kadhum, Mohammed Anbar, Shafiq Ul Rehman, and Naser K. A. Alajmi. An overview of ddos attacks based on dns. In *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 276–280, 2016.

[2] Mohamed S. Alshafaey, Ahmed I. Saleh, and Mohamed F. Alrahamawy. A new cloud-based classification methodology (cbcm) for efficient semantic web service discovery. *Cluster Computing*, 24(3):2269–2292, Sep 2021.

[3] Mardiana binti Mohamad Noor and Wan Haslina Hassan. Current research on internet of things (iot) security: A survey. *Computer Networks*, 148:283–294, 2019.

[4] Sara Blanc, José-Luis Bayo-Montón, Senén Palanca-Barrio, and Néstor X. Arreaga-Alvarado. A service discovery solution for edge choreography-based distributed embedded systems. *Sensors*, 21(2), 2021.

[5] Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery. RFC 6763, February 2013.

[6] Simone Cirani, Luca Davoli, Gianluigi Ferrari, Rémy Léone, Paolo Medagliani, Marco Picone, and Luca Veltri. A scalable and self-configuring architecture for service discovery in the internet of things. *IEEE Internet of Things Journal*, 1(5):508–521, 2014.

[7] Simone Cirani and Luca Veltri. Implementation of a framework for a dht-based distributed location service. In *2008 16th International Conference on Software, Telecommunications and Computer Networks*, pages 279–283, 2008.

[8] Leigh Cuen. Iota being shut off is the latest chapter in an absurdist history. 2020.

[9] Newman Lily Hay. Clever new ddos attack gets a lot of bang for a hacker's buck. 2019.

[10] Le Duy Ngan, Markus Kirchberg, and Rajaraman Kanagasabai. Review of semantic web service discovery methods. In *2010 6th World Congress on Services*, pages 176–177, 2010.

[11] Robert Richards. *Universal Description, Discovery, and Integration (UDDI)*, pages 751–780. Apress, Berkeley, CA, 2006.

[12] Milosh Stolikj, Pieter J. L. Cuijpers, Johan J. Lukkien, and Nina Buchina. Context based service discovery in unmanaged networks using mdns/dns-sd. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*, pages 163–165, 2016.

[13] Tsung-Yi Tang, Li-Yuan Hou, and Tyng-Yeu Liang. An iota-based service discovery framework for fog computing. *Electronics*, 10(7), 2021.

[14] Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Bittorrent's mainline dht security assessment. In *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, pages 1–5, 2011.

Adversarial attacks and defenses

Pierre Chapeau
pierre.chapeau@aalto.fi

Tutor: Lindqvist Blerta

Abstract

KEYWORDS: Deep neural networks, Adversarial attacks, Adversarial defenses

1 Introduction

DNN's (Deep Neural Networks) are now considered to be the state of the art for addressing many different problems such as Natural language processing [9], image classification [27] or speech recognition [4]. There exist however vulnerabilities to those models, indeed it is possible for models to be attacked and made to fail as it was demonstrated as early as 2004 [8]. Attacks can be seen as similar to optical illusions, with knowledge about the inner workings of human vision we can produce images or situations which reliably fools the observer.
Adversarial attacks against machine learning methods is a type of attacks which tricks models by presenting them with purposefully misleading data. This is done by exploiting the fact that complex models can produce very uneven loss landscape thus, in the loss, local maximums can usually be found around training samples, they can then be exploited to produce adversarial examples. With the constant evolution and wide

spread of machine learning methods in recent years, it has become increasingly important to study malicious attacks on models as knowledge of attacks is often the most reliable way to prevent them. Then by building more targeted defense methods one can better secure their models.
Defenses against such attacks exists and are paramount to the integrity of already deployed, sometime critical, models. With, for example, self driving cars being fooled by physical application of adversarial examples [2].
Unfortunately, while many defenses have been proposed to counter potential attacks, most have been defeated. Indeed most of the defenses that are developed seem very promising, such as distillation of model [20] for example, but after a while they are showed to be ineffective against more sophisticated attacks, such as Carlini attacks [6].

1.1 Definitions of attacks

Surveys have been done on the subject and [7] provides extensive definitions of attacks and defenses, all types will not be covered in this publication and focus will be put on **evasion attacks** and their defenses as they are the most common type of attacks. **Evasion attacks** are attacks developed after the training phase and only refer to malicious samples in the testing/deployed phase. Attacks can be categorized in two different group based on the information the attacker has on the attacked model.
If an adversary possesses specific knowledge (namely the model parameters and structure) of the target than the assault is called a **White box attack**. Some examples of attacks are PDG [15], FGSM [11], Deepfool [17] or Carlini attack [6] all of which use information about the inner working of the machine learning model such as the gradient of the objective function the model at specific inputs.
If the opponent does not possess this information directly than it is called a **Black box attack**. While hiding information about the model makes attacks more difficult, models are still vulnerable. Indeed, by training a substitute model with a decision boundary close to the original model's, one can still produce adversarial examples reliably [19, 18].

Further refinement in the type of attacks can be done, **non-targeted attacks** aim to misclassify some input while not being specific as to what kind of output the misclassification will be. **Targeted attacks** in contrast have a specific malicious output and will steer the input so that it is misclassified in the target class.

1.2 Definitions of defenses

Types of defenses are quite varied but they can be classified into two categories, the **Heuristic defenses** are defenses which effectiveness is only empirically validated without being theoretically proven. While this implies that some new attacks could be found to surpass those defense usually remain stronger [16, 24]. **Theoretically motivated** defenses provide a guarantee that they will always work under specific conditions which is exactly what is required for a certification. Unfortunately these defenses offer poor performance when compared to heuristics [22].

1.3 Structure of the paper

This publication will explain and offer a comprehensive tour of the most well know attacks and defenses and review whether their effectiveness has lasted or not.
First the most well know type of attacks will be defined and explained in sec. 2 then a rapid review of efficient and inefficient defenses in sec. 3 and finally a conclusion on the current state of research and adversarial defenses in sec. 4.

2 Most well known attacks

Attacks against machine leaning models aims to modify (in the close vicinity) an input such that it changes the output of the model attacked in a significant way (see Sec. 1.1 for the different kinds of attacks).

2.1 PGD

Projected Gradient Descent is a very general attack method that knows many variants and derivatives (eg. BIM [12] or FGSM [11], ...) The original method can be reduced to some very simple principles which are summarized in figure 1:

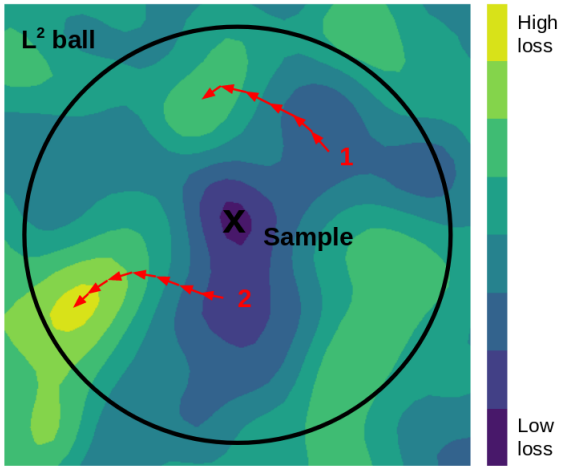


Figure 1. Loss landscape with PGD attacks [1]

1. Create a ball of size ϵ around a sample in the loss space and start at a random point (this is optional as you could start at the sample point)

2. Move in the direction of the greatest loss (this is very general and can be refined)

3. Project the perturbation in the ball if necessary.

4. Once you arrive at a local maximum stop.

The core of the technique is a very simple loom but it can still creates strong adversarial examples and is still considered one of the strongest attacks methods.

2.2 DeepFool

Deepfool is another method to find adversarial examples based on decision boundaries. It linearize the classifier, project the input point toward the closest decision boundary and add this perturbation to the input.
Has an approximation of the decision boundary has been done it is possible that the class still has not changed, if it is the case we can simply try again until the class has changed.
This guaranties that an adversarial example is eventually found but is more computationally expensive than PGD.

2.3 Carlini attack

If we view Adversarial attacks as optimization problems we can write the following :

$$\begin{aligned} &\min \mathcal{D}(x, x+\delta) \\ &\text{s.t. } \mathcal{C}(x+\delta) = t \\ &x+\delta \in [0,1]^n \end{aligned}$$

In this form what is meant is : Minimize *some* distance between the original input and the adversarial example such that it will change the classification of $x+\delta$, where $\mathcal{C}$ is the classification function.
Of course this function is highly non-linear and thus the problem is difficult to solve. Calini and Wagner propose to replace this with some function f such that $\mathcal{C}(f+\delta) = t$ if and only if $f(x+\delta) \leq 0$, the problem thus becomes :

$$\begin{aligned} &\min \mathcal{D}(x, x+\delta) \\ &\text{s.t. } f(x+\delta) \leq 0 \\ &x+\delta \in [0,1]^n \end{aligned}$$

There exist many different choices for the function f and usual optimization methods are applied to solve the problem efficiently.
One of the main advantages of such attacks is that is allays produces adversarial examples as there is no limit of the size of the space explored around the sample but it does come with increased computational costs.
As was said in the introduction (see 1) Carlini attacks are most famous for showing that distillation is an inefficient technique for defense.

3 Most well known defenses

Defenses are harder to develop as they require to be effective against every type of attack and often fail do to so. In contrast adversarial attacks have to find one weakness in a type of model or defense to be effective.

3.1 Distillation defense

Distillation network are networks which tries to learn the output probabilities of another network. This results usually in the second model being smaller and thus faster and cheaper to run while retaining most of the capabilities of the first one.

Distillation in defense against adversarial attacks consist in training a distillation network of the same size as the original one with specific parameters as to flatten the loss around the training samples. While has been showed to offer much success when it was first introduced [20], it is now proven inefficient against more persistent techniques [6].

3.2 Adversarial training

A very simple approach to defenses is to integrate 'attacks' into the training of models, that is modify images of the training set so that the modification maximizes the loss of the model. Then use those augmented images for training. This method can be seen as special case of data augmentation.
However this has several drawbacks :

- It is very costly to run such algorithm in the, already costly, training phase.

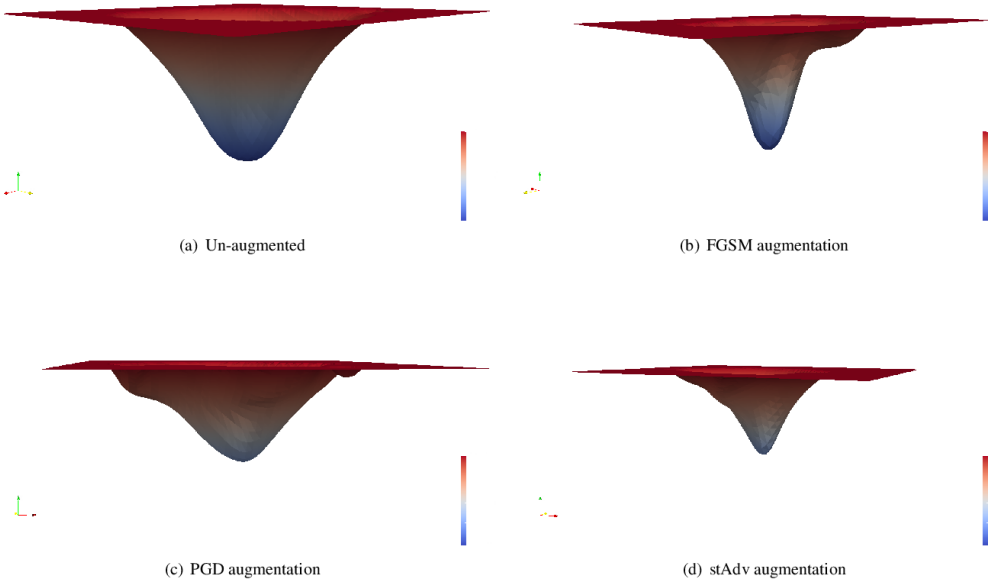
- This kind of algorithm needs to use a specific kind (or some specific kinds) of attacks in the training phase and thus may be not well suited to defend against different kinds of attacks.

- Adversarial training does not necessarily result in smoother loss landscape (but the contrary [21, 5]) and thus does not necessarily improve generalization as we can see in figure 2.

Even with those limitations Adversarial training remains one of the best defense methods.

3.3 Feature squeezing

To prevent malicious attacks another technique is to reduce the attack capabilities, that is reducing the input space such that attacks have less freedom in modifying inputs. Feature squeezing is an effective way of doing so [26].
By reducing the number of values for pixel encoding or smoothing images, machine learning models effectively reduces the search space of malicious algorithms. Unfortunately while this is an effective technique it does re-



(a) Un-augmented (b) FGSM augmentation

(c) PGD augmentation (d) stAdv augmentation

Figure 2. Loss landscapes with Adversarial training [21]

duces accuracy to a certain extent, it has also being show to fail against more persistent techniques [23].

3.4 Randomization

Some defenses use randomization as a defense strategy, indeed, machine learning model are know to be quite robust under random modification of the data. The idea is then to randomly modify the adversarial modifications in the input on in the network so that the model is not deceived by the targeted perturbation.
This can be done on several levels :

- Stochastic activation pruning [10], here the idea is to removes random subset of activation and boost the remaining one. while it has shown great promise and applicability it has been shown to fail against white box attacks [3].

- Input randomization [25], by doing randomization of the input (random resizing and random padding). The algorithm made 2nd place at NIPS 2017, it has remarkably low impact on training time, inference time and accuracy results. But, as stochastic activation pruning, it has been shown to fail against a white box attack [3].

- Random noise layers [14], By adding random noise layer at every convolution layer and then ensembling the prediction as to stabilize them

one can create a strong defense. This work remarkably well, probably because ensembling techniques tend to smooth the loss function quite well, which is arguably best defense against adversarial attacks.

4 Conclusion

Ultimately, researching attacks is as important as researching defenses as preparing against a specific type of attacks is often more efficient than blindly making defenses but research trend may give evidence to the assumption that better generalization of models would be the most general and effective defense against adversarial attacks hence it emphasise one of the main goals of machine learning. Even malicious modification of an input **in the neighborhood** of the original one should have the same/similar outputs.
Techniques have been developed to improve objective function smoothness such as [13] but research is still ongoing in the topic. Relying on heuristic defenses for the protection of an application is probably the best option at the moment but with time those defenses will need to be updated as the trend shows that most become ineffective with the development of new attacks. Defenses against adversarial attack is still a very open and rapidly moving research topic and people interested in making their particular application more robust should take an interest in this development.

References

[1] Pgd visual explanation. https://towardsdatascience.com/know-your-enemy-7f7c5038bdf3. Accessed: 2022-10-22.

[2] Three small stickers in intersection can cause tesla autopilot to swerve into wrong lane. https://spectrum.ieee.org/three-small-stickers-on-road-can-steer-tesla-autopilot-into-oncoming-lane. Accessed: 2022-10-22.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. 80:284–293, 10–15 Jul 2018.

[4] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020.

[5] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *CoRR*, abs/2102.01356, 2021.

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. 2016.

[7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021.

[8] Nilesh Dalvi, Pedro Domingos, Mausam Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. 07 2004.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[10] Guneet S. Dhillon, Kamyar Azizzadenesheli, Zachary C. Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. *CoRR*, abs/1803.01442, 2018.

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. 2014.

[12] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[13] Blerta Lindqvist. A novel method for function smoothness in neural networks. *IEEE Access*, 10:75354–75364, 2022.

[14] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. *CoRR*, abs/1712.00673, 2017.

[15] 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.

[16] 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.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. 2015.

[18] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[19] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016.

[20] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015.

[21] Vinay Uday Prabhu, Dian Ang Yap, Joyce Xu, and John Whaley. Understanding adversarial robustness through loss landscape geometries. *ArXiv*, abs/1907.09061, 2019.

[22] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018.

[23] Yash Sharma and Pin-Yu Chen. Bypassing feature squeezing by increasing adversary strength. 03 2018.

[24] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[25] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan L. Yuille. Mitigating adversarial effects through randomization. *CoRR*, abs/1711.01991, 2017.

[26] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018.

[27] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.

Using Semantic Models to Improve Interoperability of Intelligent Transportation Systems

Tarmo Asikainen
tarmo.asikainen@aalto.fi

Tutor: Lorenzo Corneo

Abstract

The domain of Intelligent Transportation Systems (ITS) aims to improve traffic safety and efficiency through interconnected networks of sensors, vehicles and computers. Interoperability of services is challenging due to differing standards and technologies used by actors in the space. The concept of semantic models, a type of information model that attaches explicit meaning to data, has been proposed as a solution to aid interoperability. This work provides an overview of current research in using semantic models to aid interoperability in the ITS domain and highlights some key challenges that are candidates for future research.

KEYWORDS: Intelligent Transportation Systems, Semantic model, Interoperability, Ontology

1 Introduction

Traffic is a large producer of polluting emissions as well as a common cause of injury or death among the population. Intelligent Transportation Systems (ITS) is a term describing the use of intelligent systems, data sensors and realtime communications with the goal of making transportation safer and more efficient. The ITS infrastructure consists of many differ-

ent kinds of devices all connected together in a network to achieve these goals.

The ITS field consists of numerous different actors, including car manufacturers, global standards organizations and public actors at different levels of government, such as public transport agencies and road infrastructure maintenance personnel. Many actors rely on their own systems that are using different, often proprietary, standards for data storage and communications. This makes the interoperability of these systems difficult in practice. However, in order to truly achieve the goals of ITS, interoperability is key.

In the recent years, the idea of semantic models has been proposed to aid in the challenge of interoperability . A semantic model is a representation of information that aims to provide explicit meaning to data that can be understood by both machines and humans. Shared semantic models can be used to ensure that different systems have a common understanding of data semantics, and thus exchanging information in different data formats becomes easier.

The purpose of this study is to provide an overview of the current research in using semantic models to improve interoperability of ITS. The result of this study is the highlighting of two key challenges in the field, and solving these challenges could be the focus of future research.

The rest of this paper is structured as follows: section 2 provides background information and definitions of key concepts, with subsection 2.1 introducing the concepts of ITS and subsection 2.2 focusing on Semantic Models. In section 3, the current state-of-the-art research on the subject is presented. In section 4 the current research and solutions are analyzed and the key challenges are listed, and section 5 concludes this paper.

2 Background

2.1 Intelligent Transportation Systems

The vision of Intelligent Transportation Systems (ITS) is to make transportation safer and more efficient through the use of intelligent systems and real time communications [9]. The main way to achieve this is through gathering of sensor data and adding intelligent functionality to both vehicles and the underlying infrastructure [8]. The concept of ITS can include

all methods of transportation, including on land, air and sea. However, this paper focuses specifically on road traffic.

An important characteristic of ITS is the requirement for many different actors from various backgrounds to be able to communicate effectively. Actors in the transportation space include car manufacturers, public transportation agencies and maintainers of road infrastructure. The multitude of different actors brings with it many different methods and standards for information exchange, while the interoperability of different systems is a key requirement. The problem of interoperability and information exchange has been identified as a key challenge in ITS in multiple studies [26] [14] [1].

There are many different kinds of services that ITS can offer, with varying requirements, providers and users. Alam et al. [2] distinguish three main categories: traffic safety, traffic efficiency and other applications. The first category includes applications such as an alert system giving road users a notification of nearby accidents, or an intelligent traffic light controller that gives priority to emergency vehicles. The second category includes for example programmable traffic signs and navigation services. The last category includes everything that is not directly related to increasing safety or efficiency, such as public transport information services.

The EU ITS Communications and Information Protocols (EU-ICIP) guide [9] provides an overview of communication and data exchange standards for use in ITS in Europe. The guide lists hundreds of related standards and it is clear that interoperability of many different data formats and data exchange protocols is necessary.

One example of a common data model in Europe is DATEX II [7]. DATEX II is an extensive data model for exchanging a broad range of different kinds of traffic related information. The aim of DATEX II is to be a unifying data model that enables ITS services in Europe, and one goal in its design is to enable services that are used across multiple countries.

A unique characteristic of ITS is the requirement for ad-hoc wireless connections between nearby devices, because physical connections are not feasible. The devices involved in such connections include devices aboard vehicles and roadside Internet of Things nodes. These networks are referred to as Vehicular Ad-hoc Networks (VANETs) [4].

Maimaris and Papageorgiou [18] evaluate some commonly used communication technologies in ITS and list some challenges in them. They conclude that distance, bandwidth, time criticality and information secu-

rity each have their own sets of challenges. With moving vehicles, distances between devices are unreliable and disconnects and reconnects are common. Bandwidth can cause problems when many devices are trying to access the network at the same time, such as during rush hour. Some applications, especially those related to emergencies, are time-critical and care must be taken to ensure they work. Information security is also a key challenge, since the wireless communication networks can be vulnerable to denial of service or malicious forged messages.

2.2 Semantic models

The concept of semantic models, or semantic data models, deals with attaching semantics to entities or concepts in the context of computer systems. While it is difficult to find a clear definition of what exactly is a semantic model, Alexopoulos [3] describes it quite well as "any representation of data whose goal is to make the meaning of data explicit and commonly understood among humans and machines".

A commonly used type of semantic model is an ontology. An ontology defines classes, their attributes, and relationships between class members. These definitions include information on their meaning as well as constraints on how they can be used. [17] An ontology describes the structure of data and gives rules for creating new entities, defining their attributes and establishing constraints [6]. A key feature of ontologies is to be able to make logical deductions based on constraints and the relationships between classes.

As an example, a simple ontology could define a class *Person* with the attributes *age* and *name*. A *Person* could also have a relationship *has_child* to another *Person*. Another class, *Parent*, can be defined which is a subclass of *Person*. A requirement of the class *Parent* is that all of its members need to have a *has_child* relationship to another *Person*. Now let us create two instances of *Person* with the names John and Mike, and define a *has_child* relationship from John to Mike. It can be deduced that because John has a *has_child* to another *Person*, it must be true that John also belongs to class *Parent*. This example is presented in figure 1.

Ontologies are used as a basis for Semantic Web technologies [5]. The term Semantic Web refers to the notion of bringing explicit meaning to web resources. Semantic Web standards include Resource Description Framework (RDF) [22] and Web Ontology Language (OWL) [21]. RDF is a data format for exchanging structured data on the web. The main

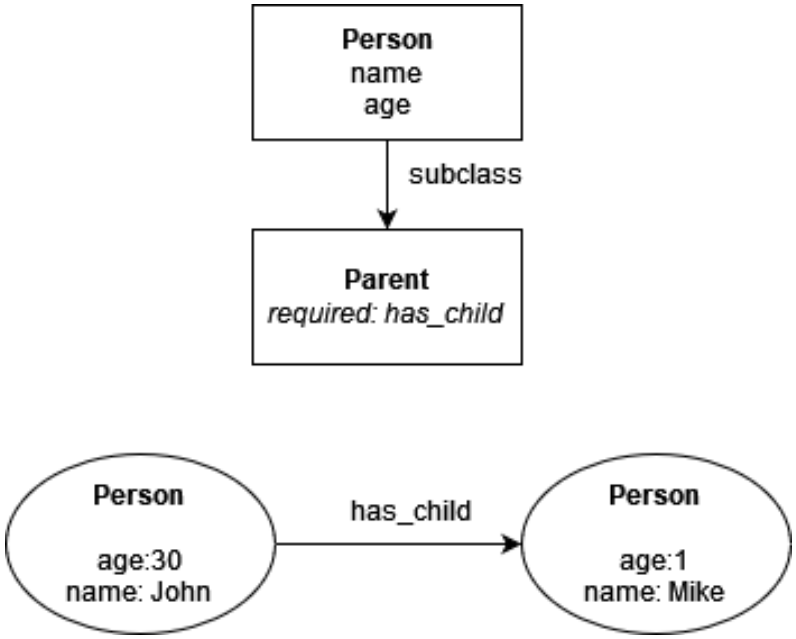


Figure 1. A simple ontology defining a parent-child relationship

construct of RDF is a triple which consists of two entities and a named relationship between them. An easy way to visualize these triples is a directed, labeled graph. OWL is a framework for describing ontologies using RDF.

Knowledge graphs [10] can also be considered a type of semantic model that is used especially in big data applications [20]. The term Knowledge graph was first coined by Google, and the technology is used to enhance Google search results [24]. However, this paper mainly focuses on the use of ontologies.

3 Semantic interoperability in ITS

This section presents some recent research studies on using semantic models to improve the interoperability of ITS.

Agbaje et al. [1] present a survey of the interoperability challenges for Internet of Vehicles (IoV), which is an application of the Internet of Things (IoT) in the context of ITS. They provide a categorization of five distinct classes of interoperability problems in IoV: Interoperability of the physical IoV nodes, network interoperability, systems interoperability which includes cross-platform and cross-domain concerns, applications interoperability and finally data interoperability which is further divided into semantic and syntactic challenges. Semantic web technologies like OWL

are proposed as a possible solution for the semantic interoperability challenges. Their work highlights two problems in this area: the absence of necessary ontologies for IoV context, as well as resource concerns for IoV devices.

In [11], Fernandez et al. present an ontology-based architecture for using sensor data in an ITS context. This includes manually constructing a mapping from sensor data to semantic data that can be used in the traffic context.

García et al. [12] present a context model and underlying ontologies for use in the public transportation applications. Their solution models the whole public transportation field, with traffic at land, air and sea. Their solution has been seen real world use by the Public Transport Authority of Gran Canaria to provide services such as a smart card payment system and an information system for passengers.

Gould and Atkin [13] present an OWL ontology for facilitating data exchange in information systems for public transport passengers and other road users. They introduce the concept of personas in their semantic model. A persona can be e.g. a regular commuter or a tourist. With the data model having built-in concepts for different types of users, their goal is to allow designing applications such as a journey planner that has support for different kinds of users at the data level.

Terziyan et al. [25] propose a semantic middleware to facilitate interoperability of different ITS resources. The middleware uses ontologies to maintain a central view of all data, and each resource requires an adapter to connect to the middleware. The advantage of using semantic data models here is the ability to discover heterogenous resources and integrating them into the system. In addition, the system allows controlling the behaviour of these resources.

Gregor et al. [14] present a methodology for creating ontologies for ITS context that uses Systematic Literature Review and Information Recovery techniques. They produce an ontology of different kinds of applications in the ITS domain, and the model could be used by a semantic service for service discovery in a distributed network. The model is evaluated for performance and validated against an existing taxonomy of ITS applications and it is found to contain many applications not present in the taxonomy. Their method of constructing the model through systematic literature review techniques can be used in a somewhat automated manner, but it does require human supervision.

In [15] Jetlund et al. analyze methods of automatic conversion from geospatial data models defined in UML to OWL ontologies and raise some key challenges. The conclusion from their paper is that more advanced conversion methods are needed to preserve semantics accurately, due to the differences between the concepts in UML and OWL. They suggest some additional restrictions to apply when constructing the UML models to allow easier conversion to OWL. These restrictions include needing to define local properties that are reused multiple times as global properties, and using links to external concepts when possible instead of defining new concepts.

Seliverstov and Rossetti [23] propose a method of aggregating spatio-temporal transportation data by building a local ontology for each data source, and integrating these into a global ontology. This approach allows querying different data sources from a central endpoint.

In [19] Mirboland and Smarsly construct a semantic model of the ITS landscape for use in building information modeling (BIM) applications. The model is used to create an extension to the Industry Foundation Classes (IFC) data schema which can be used to represent ITS elements. IFC is used as a data exchange standard in the architecture, engineering and construction industry. The resulting schema extension could be used for simulating or designing ITS systems with BIM technologies, aiding interoperability of the ITS and the architecture, engineering and construction domains.

Jetlund et al. [16] study the interoperability of different data models across the domains of Geographic Information System (GIS) and ITS. They propose a set of requirements, including compatibility with existing best practice standards for GIS applications and the support for a road network, that any common data formats between the two domains should satisfy. Currently no existing data formats satisfy all the criteria, so they construct a new semantic model that meets the requirements and could be used as a prototype model for exchanging information between GIS and ITS domains. However, the model is missing some key features and requires further work to be used in real life systems.

4 Analysis and discussion

There has been much research in improving interoperability of ITS with semantic modeling. However, there are some challenges that can be iden-

tified from the existing research. Many studies provide a contribution of some new semantic model for the field, but many of these are not suited for real world use and require further work.

There are many studies that construct ontologies for specific use cases, such as [13] and [12] that focus on public transportation specifically, and [11] that focuses on sensor data. Because many of the ontologies ultimately deal with the same concepts, one way to construct an overarching ontology for the whole ITS field could be to make one higher-level ontology that will link to other, more specialized ontologies. A similar approach of combining local ontologies to a global one is presented in [23], where multiple data sources are mapped to a global ontology.

One way to bring interoperability is to construct a mapping of existing data sources to some unified semantic model, which would allow automated translating data from one data format to another. Some work on this exists, such as [15] which brings a method of translating UML to OWL. However, this method requires additional restrictions for the UML models that are not present in the original data. Additionally, a demonstration of a method translating data from one existing ITS standard to another could be useful.

Another concern is the construction of ontologies. This should be an automated process from some existing data, especially because new ITS applications could introduce novel concepts that need to be modeled. Multiple methods for ontology construction are presented, such as [11] [14] [23]. However, these methods are require some manual supervision. An automated process would also require some validation for the model. A technique for validation is presented on [14], but it is based on scientific literature and thus might not work when adding novel concepts.

Data used in the ITS domain shares concepts from the GIS or architecture, building and construction domains, and some work has been done to find common data models for these domains. Common semantic models for ITS and GIS has been studied in [16] and similarly for the architecture, building and construction domain in [19].

Thus, two major challenges in the field can be presented:

- Automated ontology construction for ITS context from existing data

- Translation between two or more existing ITS data exchange formats using a common ontology

These concepts are somewhat linked, since the existense of a common model is required before it can be used to do translation between data formats, and the construction of that model should be done in an automated manner.

In this study, the main focus is on ontologies. However, as mentioned in section 2.2, knowledge graph is another type of semantic model that could be explored in the context of ITS interoperability. Knowledge graphs are used especially in big data applications, and some ITS applications, for example those involving data from traffic flows, could be considered big data.

5 Conclusion

This paper presents an overview of the current research on using semantic models to improve interoperability in an ITS context. Many studies has been conducted in creating semantic models for this purpose, but many focus on a narrow area of ITS or a specific use case. The main contribution of this paper is two challenges that need addressing: automatic semantic model construction and automatic translation between ITS data standards using ontologies as an intermediate step. It is clear that solving these challenges would be a good topic for future research in the field.

References

[1] Paul Agbaje, Afia Anjum, Arkajyoti Mitra, Emmanuel Oseghale, Gedare Bloom, and Habeeb Olufowobi. Survey of interoperability challenges in the internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–24, 2022.

[2] Muhammad Alam, Joaquim Ferreira, and José Fonseca. *Introduction to Intelligent Transportation Systems*, pages 1–17. Springer International Publishing, Cham, 2016.

[3] Panos Alexopoulos. *Semantic Modeling for Data*. O'Reilly Media, 2020.

[4] Fan Bai and Bhaskar Krishnamachari. Exploiting the wisdom of the crowd: localized, distributed information-centric vanets [topics in automotive networking]. *IEEE Communications Magazine*, 48(5):138–146, 2010.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[6] Kurt Cagle. Taxonomies vs. ontologies. `https://www.forbes.com/sites/cognitiveworld/2019/03/24/taxonomies-vs-ontologies/?sh=31c8d7737d53`. Accessed: 2022-11-13.

[7] DATEX II - specifications. `https://datex2.eu/datex2/specifications`. Accessed: 2022-10-04.

[8] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, 2010.

[9] EU-ICIP Guide to Intelligent Transport Systems Standards. `https://www.mobilityits.eu/home`. Accessed: 2022-10-04.

[10] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. *Introduction: What Is a Knowledge Graph?*, pages 1–10. Springer International Publishing, Cham, 2020.

[11] Susel Fernandez, Rafik Hadfi, Takayuki Ito, Ivan Marsa-Maestre, and Juan R. Velasco. Ontology-based architecture for intelligent transportation systems using a traffic sensor network. *Sensors*, 16(8), 2016.

[12] Carmelo R. García, Gabino Padrón, Pedro Gil, Alexis Quesada-Arencibia, Francisco Alayón, and Ricardo Pérez. Context model for ubiquitous information services of public transport. In José Bravo, Diego López-de Ipiña, and Francisco Moya, editors, *Ubiquitous Computing and Ambient Intelligence*, pages 350–358, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] Nicholas Gould and David Atkin. Towards a semantic layer to support road and public transport user decision-making. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1498–1503, 2015.

[14] D. Gregor, S. Toral, T. Ariza, F. Barrero, R. Gregor, J. Rodas, and M. Arzamendia. A methodology for structured ontology construction applied to intelligent transportation systems. *Computer Standards & Interfaces*, 47:108–119, 2016.

[15] Knut Jetlund, Erling Onstein, and Lizhen Huang. Adapted rules for uml modelling of geospatial information for model-driven implementation as owl ontologies. *ISPRS International Journal of Geo-Information*, 8(9), 2019.

[16] Knut Jetlund, Erling Onstein, and Lizhen Huang. Information exchange between gis and geospatial its databases based on a generic model. *ISPRS International Journal of Geo-Information*, 8(3), 2019.

[17] Ling Liu and M Tamer Özsu. *Encyclopedia of database systems*, volume 6, pages 2559 – 2561. Springer, 2009.

[18] Athanasios Maimaris and George Papageorgiou. A review of intelligent transportation systems from a communications technology perspective. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 54–59, 2016.

[19] Mahsa Mirboland and Kay Smarsly. Bim-based description of intelligent transportation systems for roads. *Infrastructures*, 6(4), 2021.

[20] Open Data Science (ODSC). Where ontologies end and knowledge graphs begin. `https://medium.com/predict/where-ontologies-end-and-knowledge-graphs-begin-6fe0cdede1ed`. Accessed: 2022-11-13.

[21] OWL - Semantic Web Standards. `https://www.w3.org/2001/sw/wiki/OWL`. Accessed: 2022-10-04.

[22] RDF - Semantic Web Standards. `https://www.w3.org/RDF/`. Accessed: 2022-10-04.

[23] Alexey Seliverstov and Rosaldo J. F. Rossetti. An ontological approach to spatio-temporal information modelling in transportation. In *2015 IEEE First International Smart Cities Conference (ISC2)*, pages 1–6, 2015.

[24] Amit Singhal. Introducing the knowledge graph: things, not strings. `https://blog.google/products/search/introducing-knowledge-graph-things-not/`. Accessed: 2022-11-13.

[25] Vagan Terziyan, Olena Kaykova, and Dmytro Zhovtobryukh. Ubiroad: Semantic middleware for context-aware smart road environments. In *2010 Fifth International Conference on Internet and Web Applications and Services*, pages 295–302, 2010.

[26] Shalini Yadav and Rahul Rishi. A systematic and critical analysis of the developments in the field of intelligent transportation system. *Advances in Dynamical Systems and Applications (ADSA)*, 16(2):901–912, 2021.

Federations and trust networks for identity sharing in Member States and education in Europe

Tiia-Maria Hyvönen tiia.hyvonen@aalto.fi

Tutor: Timo-Pekka Heikkinen

Abstract

Digital identities and their management are critically important in the digital environment. They are often used in the Internet and other networks to protect resources by restricting who can accesss them. Moreover, using digital identities in a federated environment has many security and usability benefits, and an increasing number of organisations depend on collaboration when it comes to identity sharing.

This paper is a review article based on existing research, literature and specifications of federated identity management and trust networks used for identity sharing. The focus is in interoperability between EU Member States and educational environments especially in Europe.

In the EU, a significant factor in secure identity sharing and strong authentication is the eIDAS interoperability framework. In academic environment, there are multiple projects attempting to provide students identification or other identity processes that ease mobility between and within countires. A significant federation in the educational environment is eduGAIN, which consists of smaller academic federations.

KEYWORDS: digital identity, Federated Identity Management, FIdM, eIDAS, eduGAIN

1 Introduction

The root problem when IT-security incidents occur is that an unauthorised actor has gained access to data. Hence, Identity Management (IdM) is an essential part of IT-security for organisations. IdM is a framework of policies and technologies that are used for controlling and managing digital identities in computer systems, and when IdM is deployed well and combined with Access Management, the solution provides protection on confidential resources.

Organisations have varying needs for Identity Management and there is no solution that suits for every purpose. Complexity of the required solution depends on organisational structures, used technologies, policies, regulations, laws and domain-specific factors. Furthermore, it is common that organisations depend on each other and require Federated Identity Management (FIdM) solutions. With FIdM, the organisations with predefined trust relations can distribute identity information in a decentralised environment in order to provide access to cross-domain services.

Even though FIdM is quite a mature topic in research, there is also plenty of room for study and comparison. This is due to the fact that the umbrella of IdM is wide, the federation solutions in different sectors have specific needs for collaborations and they are constantly evolving. There is a high demand for cross-border services which require secure authentication in the European Union. In addition, there are initiatives and regulations attempting to provide these services. Based on existing research, this paper gives an overview of the state of identity sharing federations and networks in the international and educational environments in the EU.

Next section of this paper will cover the background for Federated Identity Management, including related research and relevant concepts, such as digital identity, IdM roles, Singe Sign-On, federation and underlying protocols. The topic of the third section is identity sharing in EU Member States and fourth in educational environments. The last section combines conclusion and suggestions for future work.

2 Background

This section explains important concepts in FIdM. It introduces the topic in general and gives necessary background for the rest of the paper.

Related work

Previous research on eIDAS includes survey of technology trends for notified eIDAS Schemes [2] and analysis on the eIDAS revision process [12] among others. There is also existing research on eID schemes of some specific countries, such as Estonia, Netherlands, UK and Hungarian. During the last few years, there has been research on how to use eIDAS in academic services [4] [17].

Variety of research papers of EU level projects attempting to provide identity sharing in educational environments exists, especially for the Erasmus Program, but it is difficult to find research that summarises the related initiatives, projects and regulations. Although, the European Student eID Framework Proposal [13] has listed many of them.

Digital identity and its lifecycle

At the heart of Identity Management is the concept of a *digital identity*. It is a digital presentation of information about a person, company, device, web element, software or other entity. This paper mainly focuses on digital identities of people. Generally digital identities consist of three categories of information: identifiers, credentials, and attributes [11]. The identifier can be an email address, a string, a public key, or something else that can be used to locally or globally identify an entity. Credentials are used for *authentication*. Authentication refers to the process of verifying that the entity is someone or something they claim to be. There are many credential options such as username-password-pairs, PIN-codes, graphical passwords, biometrics, one-time password, smartcards and other security tokens. The authentication system might require one or more credentials to grant access to the protected resources. When multiple credentials are used, the process is called Multi-Factor Authentication. [5] Attributes include all other information that describes the entity, including an entity's characteristics (name, date of birth, ..), access rights (i.e. roles in organisation), restrictions and possibly dynamic information (i.e. current location). [11] It is good to acknowledge that digital identity does not have one universal definition and for instance, electronic identity and eID can be used interchangeably with

digital identity. In this paper, eID refers to the electronic identity in contexts of eIDAS regulation and Student eID project which the paper discusses later. [12]

In order to better understand identity management, it is useful to understand the lifecycle of digital identities. Silander [11] divides the lifecycle into four stages in his paper: provisioning, use, updating, and deprovisioning. Additionally, there is a constant process of auditing and maintenance of the lifecycle and all identities according to agreed policies. The first stage, provisioning, includes the creation of the identities and forwarding them to the target systems. During the second state, identity information is used for identification, authentication, authorization, signatures and other processes. The third state is required, if attributes or credentials of the identity change, then they need to be updated. [11] For example, if an employee gets a promotion, her/his role and access rights most likely change and need to be updated to correspond to the new position in the company. Lastly, the stage that all digital identities should go through before the end of the lifecycle is deprovisioning, where they must be deleted from all target systems. Deprovisioning is important, because old unused user accounts in IT-systems cause security issues. [11]

Roles in Identity Management

In IdM systems, there are typically three types of entities interacting with each other: a user, an Identity Provider (IdP), and a Service Provider (SP). The user is a consumer of the services and always operates through an *user agent* which is typically a web browser. The user agent either allows the identity information passively flow through or actively mediates it [10]. The SP provides services to users and consumes identity services provided by IdP. The Identity Provider typically issues and manages users' identities, authenticates them and processes requests from SPs. Since SP depends on authentication services provided by IdP, it is also referred to as Relying Party (RP).[16] [10] An entity can be both, SP and IdP, at the same time.

Single Sign-On

Single Sign-On (SSO) refers to a mechanism that allows users to authenticate once and gain access to multiple protected resources. That is, the user can log in to a service and is able to use other independent services without need for further authentications during the session. [14]

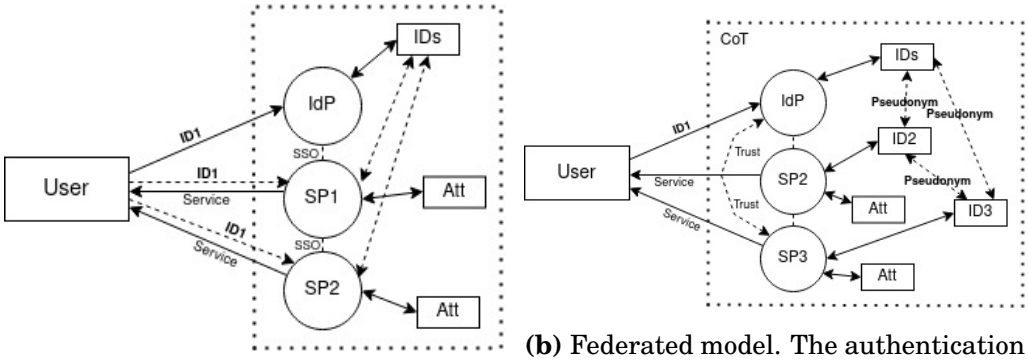
Currently a well recognized problem is that users have more web accounts than they are able to remember credentials, which leads to poor password choices and management practices. SSO reduces the number of user accounts and also improves usability of services since credentials do not need to be typed repeatedly. For SPs, SSO decreases costs, since authentication and other identity management processes are delegated to IdP [10]. Although SSO comes with many benefits, it is also important to recognize its disadvantages. For example, if a user account gets compromised in the SSO system, the attacker gains access to all services instead of one. SSO solutions can be classified on how they are deployed: Web SSO is often used in Internet to provide access to applications deployed on web servers, Enterprise SSO provides access to services in the enterprise or organisation (ESSO) and multi-domain SSO connects services of enterprise and its business parteners. [14]

Federated IdM

Traditionally SPs have managed the user identities themselves in *isolated IdM model*. That is, the web services have been SP and IdP at the same time. In fact, many web services still use an isolated model even though it means that users must handle numerous accounts themselves. Figure 1a illustrates *centralised IdM model*. In the centralised model, the role of IdP has been separated from SPs and identity information is stored and maintained in a centralised location. This model supports the use of SSO. [9]

Figure 1b presents *federated IdM model*. In FIdM systems, member organisations allow users to access each other's resources without centralised identity control. This model always requires trust relations built on agreed business contracts and common technology platforms between attending organisations [9]. The group of IdP and SPs in the federation is called Circle of Trust (CoT), and the actors in CoT share metadata [10]. In a FIdM system, only the users' home organisation needs to maintain their identities, but users can still access all SPs in CoT. Moreover, one of the most basic functions that FIdM systems provide is multi-domain SSO [10].

Identity information is typically highly sensitive and it is used to access protected resources. Especially in a federated environment where the identities are shared over organisational boundaries, it is important to take care of the security and privacy. To address these needs, pseudonyms are used in exchange of identity information between SPs and IdPs, and the rule of minimal disclosure is applied to shared attributes in FIdM systems. [7]



(a) Centralized model. Multiple SPs authenticate users against central IdP.

(b) Federated model. The authentication is directed to users' home organisation's IdP.

Figure 1. Centralized and federated IdM models. Redrawn from [9].

Protocols

There are three widely used industry standard identity protocols: OAuth 2.0, OpenID Connect and SAML 2.0 [15]. These protocols are shortly introduced, because they are important in FIdM, but this paper will not explain how they work in detail. OAuth 2.0 is an authorization framework that enables third-party applications to access HTTP-services and OpenID Connect implements an authentication layer on top of OAuth 2.0 [15]. SAML is an XML-based framework for exchanging identity and security information between entities. [3] It provides cross-domain SSO and identity federation [15].

3 EU Member States

Significant factor in international identification systems in the European Union is a regulation called eIDAS, which stands for electronic IDentification, Authentication and trust Services for electronic transactions in the internal market. The eIDAS regulation enables interoperability between Member States (MS) by permitting a user of eIDAS-enabled SP to authenticate in their home country instead of the country that provides the SP.

Interoperability in eIDAS is achieved by defining technical interfaces between *eIDAS-nodes*. There are two different kinds of eIDAS-nodes: eIDAS-Connector and eIDAS-Service. Latter can be categorised further to eIDAS-Proxy-Services and eIDAS-Middleware-Services. Depending on which kind of eIDAS-service is used, the architecture is called Proxy-based-scheme or Middleware-based scheme. [6]

In the eIDAS architecture receiving MS provides services and relies on au-

thentication and identity information of sending MS. The eIDAS-enabled RP is connected to eIDAS-Connector of receiving MS which again is connected to eIDAS-Service of the sending MS. The eIDAS-Service of sending MS is connected to national electronic identification system: eID scheme. Figure 2 illustrates simplified connections of eIDAS architecture. In reality, all MSs have both eIDAS-Connector and -Service. Communications between eIDAS-nodes use SAML-protocol. [6]

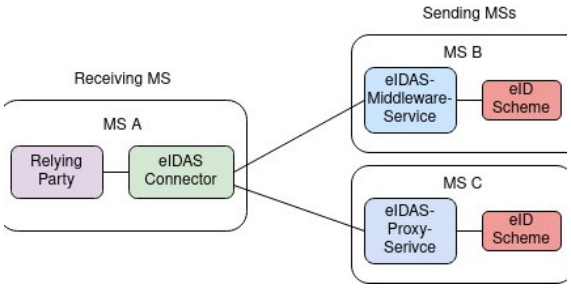


Figure 2. The eIDAS-Connector of the receiving MS A is connected to other countries eIDAS-services. [7]

The eIDAS regulation has been compulsory for all MSs since 2018, but many MSs have been unsuccessful when trying to integrate the regulation to their national eID-schemes [12]. This is because MSs have developed their eID management systems independently trying to meet their internal goals of secure authentication prior to trying to have interoperability with other countries' eID schemes. Thus, the national identification solutions between MSs have significant differences that have led to obstacles in terms of cross-border interoperability [7]. Besides interoperability issues and differences between national eID implementations, some authors have suggested that eIDAS implementations have difficulties because of the complexity of the eID concept. As a result, the European Commission proposed an amended draft of the eIDAS regulation based on collected feedback in July 2021. The new draft includes functionality called EU digital identity wallet, and harmonisation with other EU regulations and standards among other things. [12]

4 Education and reaserach environment

Academic institutions typically form national federations in different countries. In Finland HAKA federation is responsible for enabling SSO between research and educational institutions with SAML-based software called Shib-

boleth which is commonly used in academic environments. When an organisation has successfully applied to HAKA, metadata of its servers is added to the resource register and the organisation can join the SAML metadata exchange of the whole federation. [8] Other countries also have their own federations and trust networks for academic institutions. For example, Germany has DFN-AAI and Sweden has SWAMID [1].

EduGAIN is a large, international federation formed by smaller federations [13] such as HAKA. In eduGAIN, SSO is enabled with Shibboleth with no extensions. [8] [4] After SP or IdP has registered to one of the subfederations, it can also register to eduGAIN. Subfederation might have varying levels of trust built into their systems and therefore entities in eduGAIN do not automatically transfer identity attributes between each other. In contrast, in HAKA it is the default behaviour that all IdPs and SPs share attributes between each other. Thus, entities in eduGAIN can choose with whom they form communications agreements. [8]

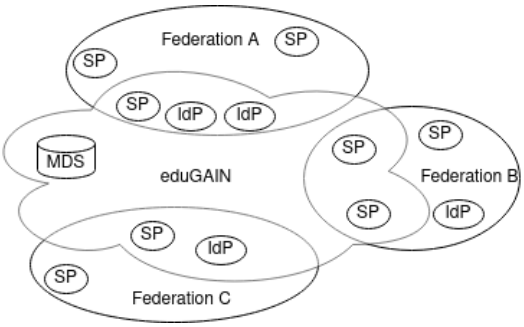


Figure 3. SPs and IdPs can be part of eduGAIN or only the subfederation. Metadata Distribution service provides details of entities.

The eIDAS framework can be used in academic services and other domains, such as, banking and healthcare. In fact, many academic e-services provided by higher education institutions offer the possibility for students to authenticate by using their national eIDs [17]. However, the problem with using eIDAS in cross-border identity sharing for certain domains is that countries provide only the restricted basic set of identity attributes, not anything additional. [4] [17] That is, a service relying on authentication through eIDAS will only gain a minimum dataset of user profile, which contains citizens' personal and legal attributes, not academic or other domain-specific attributes that may be required to use certain services. [4] [17] Solutions for this problem must address two challenges; (1) Since eIDAS does not support any additional attributes besides the minimum dataset, relevant academic attributes must be recognized and eIDAS specification extended to support

these attributes. And (2) third-party attribute providers are needed in order to add academic attributes to eIDAS profiles since IdPs connected to eIDAS-nodes provide only personal and legal information according to the national eID scheme. [17] Even though these challenges are still an open issue [4], there already exist working implementations of extensions that address these challenges.

Traditionally High Education Institutions (HEIs) have had their own standalone solutions for eID schemes varying between and within the countries which makes the digital interoperability and mobility between HEIs difficult or even impossible [13]. European Campus Card Association (ECCA) has been promoting research for the development and implementation of eID credentials for HEIs to allow interoperability for students in Europe. Furthermore, ECCA's recent Student eID project is progressing to support provision of secure identification and authentication of students on a cross-border basis. There is a wide range of related projects and initiatives aiming to provide better mobility for students, and the Student eID project recognized 13 other projects that are relevant for the delivery of European eID credentials, including EUROLogin, eID4U and European Digital University Card Student. Additionally, the credentials should be interoperable with eIDAS and other EU policies. [13]

5 Conclusion and Future work

The increasing mobility worldwide and in the EU increases the demand for secure digital identity sharing. FIdM addresses this need by enabling large-scale decentralised identity sharing between organisations with predefined trust relations, in addition, many security and usability benefits come with it. In many sectors, FIdM is already widely used or in development, including e-government and other services that use national eID schemes. There are also multiple federations and EU-level attempts to provide mobility for students in Europe by allowing cross-border identity sharing. These federations and networks are constantly developing, which requires active research and effort to create secure and usable solutions.

As already stated in the introduction, there has been much research on IdM already but there is also room for further study. Based on this paper, some ideas for future work includes more detailed comparison of eID schemes of different countries which could be expanded to outside of the EU. Additionally, more detailed comparison of federations and trust networks in academic

environments, also expanded further from Europe and other sectors. Lastly, even though there is much literature on earlier introduced widely used protocols in FIdM, the comparison of IdM solutions in software level is an interesting research topic.

References

[1] eduGAIN. GÉANT. https://www.example.com/url_with_underscore. Accessed: 2022-11-13.

[2] Amir Sharif, Matteo Ranzi, Roberto Carbone, Giada Sciarretta and Silvio Ranise. RSoK: A Survey on Technological Trends for (pre)Notified eIDAS Electronic Identity Schemes. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–10, August 2022.

[3] Brian Campbell. Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. Request for Comments. https://www.example.com/url_with_underscore, May 2015. Accessed: 2022-11-13.

[4] Diana Berbecare, Antonio Lioy and Cesare Cameroni. Electronic Identification for Universities: Building Cross-Border Services Based on the eIDAS Infrastructure. June 2019.

[5] Arunava Roy Dipankar Dasgupta and Abhijit Nag, editors. *Advances in User Authentication*. Springer Cham, 1 edition, 2017.

[6] eIDAS eID Technical Subgroup. eIDAS Interoperability Architecture v 1.2 . August 2019.

[7] Jesus Carretero, Guillermo Izquierdo-Moreno, Mario Vasile-Cabezas and Javier Garcia-Blas. Federated Identity Architecture of the European eID System. In *IEEE Access*, pages 75302 – 75326, 2018.

[8] Kari Laalo. Haka-käyttäjätunnistusjärjestelmä. eduuni wiki. https://www.example.com/url_with_underscore, August 2016. Accessed: 2022-11-15.

[9] Maryline Laurent and Samia Bouzefrane, editors. *Digital Identity Management*. ISTE Press - Elsevier, 2015.

[10] Maha Aldosary and Norah Alqahtani. Federated Identity Management (FIdM) Systems Limitation And Solutions. In *Cornell University*, April 2021.

[11] Jon Silander. Katsaus identiteetinhallinnan teknologioihin ja niiden tulevaisuuden näkymiin. Master's thesis, Aalto University, 2013.

[12] Silvia Lips, Natalia Vinogradova, Robert Krimmer and Dirk Draheim. Re-Shaping the EU Digital Identity Framework. In *The 23rd Annual International Conference on Digital Government Research*, pages 13–21, June 2022.

[13] Sinead Nealon, Tor Fridell, Eugene McKenna and Jorge Lanza. European Student eID Framework Proposal. In *The 28th International Congress of European University Information Systems*, volume 86, pages 12–21, 2022.

[14] V. Radha and D. Hitha Reddy. A Survey on Single Sign-On Techniques. In *International Conference on Computer, Communication, Control and Information Technology*, pages 134–139, 2012.

[15] Yvonne Wilson and Abhishek Hingnikar, editors. *Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0*. Apress, 1 edition, 2019.

[16] Yuan Cao and Lin Yang. A Survey of Identity Management Technology. In *IEEE International Conference on Information Theory and Information Security*, 2010.

[17] Álvaro Alonso, Alejandro Pozo, Aldo Gordillo, Sonsoles López-Pernas, Andrés Munoz-Arcentales, Lourdes Marco and Enrique Barra. Enhancing University Services by Extending the eIDAS European Specification with Academic Attributes. January 2020.

Biometrics of Intent

Timo Nappa
timo.nappa@aalto.fi

Tutor: Sanna Suoranta

Abstract

Biometrics of intent utilize the different biological properties of the human body to attempt to define the intentions of a person. As a large amount of cybersecurity breaches come from internal infiltration, more secure methods are needed for authentication. Biometrics of intent can detect when a person is about to use their authentication for nefarious purposes and prevent this from happening. Electrodermal activity (EDA), electroencephalogram (EEG) and electromyography (EMG) monitor the electrical properties of the skin, the brain and the striated muscles to detect both deception and the intent of the person. These methods are currently utilized in laboratory conditions and the results of the experiments have been promising. However, more research is needed, especially in the area of applying these technologies in field conditions.

KEYWORDS: biometrics, intent biometrics, authentication, electrodermal activity, electromyography, electroencephalogram, deception detection

1 Introduction

Biometrics is the measurement of physical characteristics to verify identity. The technology has been used for more than a century with the

first paper on fingerprint recognition used in identifying people published in 1880 [4]. In popular media, fingerprint and DNA identification are technologies commonly seen in police procedural television shows. With the development of mobile phone cameras and fingerprint sensors, facial recognition and fingerprints have become a common mode of securing access on mobile phones. Similar methods are also used in modern biometric passports.

These methods as well as passwords, another very popular method of authentication, offer no protection against cases where a person is forced to authenticate under duress or when a person authenticates with malicious intent. These types of breaches of security can be counteracted with biometrics of intent. These technologies apply certain types of biometrics, such as electrodermal activity (EDA) [1, p. 478-490], electroencephalogram (EEG) and electromyography (EMG). These types of biometrics can be utilized to detect involuntary reactions within the body of a person to establish whether their intention matches their claim or not [18, p. 688-703].

The aim of this paper is to present and review three important biometrics of intent. This paper will next explain the different types of biometrics utilized in biometrics of intent. The third section will outline deception detection tests and how biometrics of intent are utilized in detectiong deception. Section four includes conclusions and a short discussion on possible future avenues to explore.

2 Biometrics

Although there are a very large number of different types of biometrics, this paper focuses on the three aforementioned: EDA, EEG and EMG. These three have been the focus of research on biometrics of intent. An important term to understand in neural response to triggers is the concept of different types of neuron firings. Tonic firing refers to a sustained response, which activates during the course of the stimulus, while phasic firing refers to a transient response with one or few action potentials at the onset of stimulus followed by accommodation [20].

2.1 Electrodermal Activity

Electrodermal activity refers to all the electrical properties and their changes within the skin. The change within the properties is caused by sweat glands controlled by the autonomous nervous system [18, p. 159-177]. The glands secrete sweat, which causes changes in skin conductance, capacitance and potential [19]. EDA measures the excretion of palmar surfaces, as these are thought to be more responsive to psychological sweating [18, p. 159-177]. Different methods are used to measure changes in different electrical properties of the skin: 1) the endosomatic method without applying external current, and the exosomatic methods of applying either 2) direct current or 3) alternating current via electrodes.

EDA measurements are taken with electrodes placed upon the skin of the subject. The placement of the electrodes varies depending on the type of method used [5]. The measured conductance will change when the subject is exposed to external stimuli leading to autonomic nervous system activation and hence a change in the skin conductance [8]. It is important to note that skin conductivity is not influenced by parasympathetic (unconscious) activation; hence it can be considered as a measure for both cognitive and emotional activity [3]. The level of conductance on the skin varies between 2 and 20 microsiemens while the observed changes usually range from 1 to 3 microsiemens [18, p. 159-177].

EDA is measured from the surface of the skin using electrodes. A small current with a constant voltage is passed between the electrodes. The level of the current will alternate and can be used to measure the conductance of the skin according to Ohm's law $R = V/I$, where R is resistance, V is voltage and I is current, while conductance is the inverse of R, i.e., $G = I/V$ [18, p. 159-177]. The electrodes are placed on the hand of the subject. There are three location pairs which are commonly utilized; however the measurements are not comparable with each other, as the number of sweat glands varies between the locations [6]. Commonly, the measurements are taken from the non-dominant hand, as the skin is assumed to have less wear. The location should not be cleaned or abraded, as this might alter the natural electrical properties of the skin. The atmospheric properties, such as the ambient temperature and the humidity of the test environment should be kept as constant as possible to ensure the comparability of the measurements [1, p. 189-213].

2.2 Electroencephalogram

Electroencephalogram (EEG) is a technique which measures electrical activity in the brain. More specifically, it is used to measure the total activity of neurons in a localized area of the brain. This activity is synchronous and hence creates frequency oscillations. The oscillations induce an electrical field, which can be detected through the skull on the scalp. The inducted potentials can be detected as a potential difference between different parts of the scalp and thus; a reference electrode is needed as well [3]. Intracranial EEG can be recorded as well, and while the signal recorded would be stronger and clearer, the method is extremely invasive and hence not viable in most experiments [9].

There are different types of metrics recorded using EEG: spontaneous activity recorded during rest or sustained stimulus and evoked brain potential (EP) induced by phasic stimulus. Often both are measured in parallel, as they offer complimentary information [9]. The oscillations range from 1 to 44 Hz and is separated into five different bands based on the types of brain activity they reflect [18, p. 59-60]. The different frequency bands reflect different types of brain activity. The voltages measured range from 5 to 100 microvolts.

2.3 Electromyography

Electromyography (EMG) is the detection of electrical activity within the striated muscle cells [18, p. 267-291]. Contraction of these muscles can reveal the mental states of examined people, as emotional expressions reflect on the facial muscles. Facial expressions are often observed and assessed based upon the facial expression coding method. However, EMG is more effective in identifying subtle expressions that are not detected by facial coding. EMG has been shown to be primarily effective in separating positive expressions from negative ones [14].

There are two varieties of EMG: surface where the signals are recorded with electrodes from the skin and intramuscular where the signals are recorded with needles from within the muscle [18, p. 267-291]. As biometrics of intent are concentrated only on facial EMG, this paper focuses on surface EMG. The signal EMG records is created by motoneurons firing within the muscle fiber, as it contracts creating muscle action potential (MAP). A measure of this potential transfers to the skin and the electrodes detect the change. However, as numerous different MAPs affect

the potential on the skin, the change cannot be tracked back to singular actions or origins. Hence, considerable attention must be given to electrode placement as well as recording and analysis of the signal.

Surface EMG measurements require two electrodes. Prior to attaching, the skin is cleaned from dirt and oil as well as abraded reducing the impedance to between 5 and 10 kΩ [18, p. 267-291]. As the measured signals are small in both current and voltage, they are very susceptible to electrical noise. Hence, it is important to ensure that the connection between the electrode and the ground is stable and the impedance between the two measuring locations is low. The frequencies of the signals vary from several to 500 Hz, while the amplitude can be lower than one microvolt or as high as a millivolt. Due to these factors, EMG signals are vulnerable to noise, especially from electrical sources such as AC power lines operating on a 50 or 60 Hz frequency. Because of the noise, it is vital to properly shield and ground the equipment as well as the subject.

Establishing a baseline is important in EMG. A true baseline in EMG would be only the background noise of the measuring location; however, achieving zero activity in muscles is difficult to accomplish. The baseline is attempted to record with either prestimulus and pseudotrial recordings, which mimic the actual trials except for the external stimuli, or with a closed-loop baseline procedure, where stimuli is random on low enough levels of activity on the recording locations [18, p. 267-291].

Additionally, EMG is greatly affected by the social aspect involved in the recording as well as the mental state of the test subject. Tassinary & al. [18, p. 267-291] mention several social factors that affect EMG recordings, such as the audience effect, where the subject will react differently to stimuli depending on whether they are being observed or not, or the mimicry effect, where the subject will mimic some of the muscle movements they perceive on a face they are observing.

3 Deception detection

3.1 Deception detection tests

Deception detection or polygraph, is an interview where the interviewer attempts to detect if, when and where the interviewee is lying. The questions in these interviews are formulated as yes or no questions.

Deception tests can be categorized into two categories: knowledge-based tests and deception-based tests [1, p. 478-490]. Deception-based tests can be further divided into two categories: comparison question tests (CQT) and the relevant-irrelevant test (RI). Boucsein further states that the RI tests are completely useless as they do not satisfy the requirements of a psychological test and should not be used in a professional setting to make final decisions. The CQT is a technique developed after the RI test and it has improved upon its model. In addition to the relevant information and irrelevant information questions of the RI test, CQT also includes comparison questions, which are designed to create a large response from non-deceptive individuals, while deceptive individuals would react most strongly to the questions relevant to the matter.

Knowledge-based tests are called either guilty knowledge tests (GKT) or concealed information tests (CIT) and they rely on the deceptive person to react strongly to the information they know, while the non-deceptive person will not react strongly to the information they do not know[1, p. 478-490]. Of course, due to the nature of the test, the GKT cannot be used for every type of deception detection. The GKT was developed for forensic applications and hence it is most applicable in these situations.

3.2 Using EDA in deception detection

As Boucsein [1, p. 478-490] states in his book, EDA has been used in traditional polygraphs for a long time. Despite the longevity, the results have been mixed. While EDA is the most accurate of traditional polygraph detection methods, Boucsein notes that it is defeated by countermeasures such as covert muscle contractions or discreetly self-inflicted pain. Due to this and cases where a guilty party has been deemed innocent or an innocent deemed guilty by a polygraph, polygraphs are deemed inadmissible in court or are not used by the police forces [10].

A study conducted in The Netherlands, involving 97 students in two separate experiments, focused on discovering whether the intent to lie could be seen in the EDA signals of the subject the same as actual lying [17]. Two different experiments were conducted: one where the deception was regarding perceived emotions and one where it was an arithmetic task was performed. The EDA measured was exodermal skin conductance. When the test subject was lying, both tests clearly showed an increase in EDA activity. However, the results were not the same on the intention to lie. The emotion recognition test did not show an increase in EDA activity

while the arithmetic task test did.

3.3 Using EEG in deception detection

A study in the People's Republic of China, involving 33 subjects, discovered a link between theta synchronization and deception [21]. Three electrodes at three different sites on the scalp detected similar theta band oscillations when subjects were withholding information in a CIT.

Another study in People's Republic of China, involving 30 individuals, concentrated on the information flows between different cortices of the brain instead of the usual time, frequency, or temporal features [7]. The test setup had 64 electrodes recording signals during a GKT. 24 regions of interest were selected and the sources of the signals were estimated with a standardized low-resolution brain electromagnetic tomography [13]. Next, the effective connectivity was analysed with partial directed coherence [2] and differences were extracted with graph theoretical analysis [7]. The model was tested with test datasets of different frequency bands and the rate of deception detection was very high, ranging from 96 to 99% depending on the frequency band. However, there are issues with the study. A notable issue is the inability to simulate a real situation. The subjects did not expect the type of emotions, such as fear or anxiety, an actual interrogated person would. The presence of these emotions would definitely influence the EEG signals. Another issue is the lack of study on the effectiveness on lie countermeasures, which usually easily fool GKT [1, p. 478-490].

3.4 Using EMG in deception detection

A study in Australia tested 14 individuals utilizing EMG along with a polygraph. The EMG recordings were taken from the belly of the masseter muscle [15]. The participants were measured with a card test as well as an affirmative test. In the results, the researchers claim EMG helps in detecting deception as the differences in amplitude were clearly visible while the subjects were being deceptive during the test. However, more than 67% of the deception in the card test was unnoticed.

A study, with 45 participating individuals, by Shuster & al. [16] did not utilize a polygraph, but used a different method: participants acted as sender and receiver of messages. The sender would hear one of two words and would repeat either the one they heard (non-deception) or the

one they did not hear (deception) to the receiver, who would then decide whether they believed the sender was being truthful or deceptive. After a number of trials, a monetary incentive was introduced, where the sender would receive money for successfully deceiving the receiver, while the receiver would gain money, if they were able to tell correctly whether the sender was being deceitful or not. The EMG recordings were taken from the corrugator superclii muscle region (frowning) and the zygomaticus muscles region (smiling). The data was analyzed with a support vector machine classifier using a least squares cost function. The results of the study indicated that the EMG recording reliant algorithm was much better at detecting deception than the human participants. It was also notable that while slightly more than half of the participants mostly showed their deception through their zygomaticus region, a minority expressed their deception through their corrugator superclii region.

4 Discussion

It is evident from the small sampling of studies in this paper that the methods mentioned are effective at detecting deception, when the tests are done within optimal circumstances. Especially EEG had very notable detection rate. However, it is also evident that using biometrics of intent outside of a laboratory environment proposes several challenges. First and foremost is the noise bound to be present in a practical situation. Electrical devices and wiring are everywhere and the 50/60 Hz frequency is within the frequency range utilized by EEG and EMG. Hence there will be considerable noise present in the measurements. This interference can be mitigated to an extent, but in a field environment, it might prove a significant challenge. Countermeasures are another issue that might prove difficult to overcome. While Boucsein [1, p. 484] states that EMG can be used in conjunction with EDA in polygraphs to counter countermeasures, he also states that the effectiveness of it is limited.

All of the studies performed consisted of experiments where the subjects were carefully coordinated in their performance. This is ideal for initial tests to clearly indicate implication between deception and the recorded signal. However, the situation in a field use case will be remarkably different. The subjects, both innocent and guilty, will be substantially more anxious and nervous and less cooperative. Rigorous testing must be conducted prior to these kinds of experimental tests utilized as proof in courts

of law. Another significant issue is the lack of realistic stake for the subjects in the experiments. In a real life case of deception, the party performing the deception has something significant to lose, if their deception is detected. In addition, it is very probable an innocent person would experience anxiety within an interrogation situation. Hence, more research is needed to allow these methods to be utilized in real life field conditions.

One final issue is morality. Detecting the intentions of a person prior to them actually voicing them out or committing the deeds could be considered a serious breach of privacy as defined in the United Nations Declaration of Human rights [11]. Moreover, the use of a polygraph is considered unlawful in Switzerland[10] as taking a polygraph, even if willingly, can be seen as a form of self incrimination [12]. The ideas in these experiments do not venture into actual mind reading or thoughtcrime, however, these ideas are not a distant future.

References

[1] Wolfram Boucsein. *Electrodermal Activity*. Springer US, New York, NY, 2nd edition, 2012.

[2] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009.

[3] Benjamin Cowley, Marco Filetti, Kristian Lukander, Jari Torniainen, Andreas Henelius, Lauri Ahonen, Oswald Barral, Ilkka Kosunen, Teppo Valtonen, Minna Huotilainen, et al. 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*, 9(3-4):151–308, 2016.

[4] Henry Faulds. On the Skin-Furrows of the Hand. *Nature*, 22(574):605, October 1880.

[5] Society for Psychophysiological Research Ad Hoc Committee on Electrodermal Measures. Publication recommendations for electrodermal measurements. *Psychophysiology*, 49(8):1017–1034, 2012.

[6] Lauren Freedman, Angela Scarpa Scerbo, Michael E Dawson, Adrian Raine, William McClure, and Peter Venables. The relationship of sweat gland count to electrodermal activity. *Psychophysiology*, 31(2):196–200, 1994.

[7] Junfeng Gao, Xiangde Min, Qianruo Kang, Huifang Si, Huimiao Zhan, Anne Manyande, Xuebi Tian, Yinhong Dong, Hua Zheng, and Jian Song. Effective connectivity in cortical networks during deception: A lie detection study based on eeg. *IEEE Journal of Biomedical and Health Informatics*, 26(8):3755–3766, 2022.

[8] Shanshi Li, Billy Sung, Yuxia Lin, and Ondrej Mitas. Electrodermal activity measure: A methodological review. *Annals of Tourism Research*, 96:103460, 2022.

[9] Lasse Paludan Malver, Anne Brokjær, Camilla Staahl, Carina Graversen, Trine Andresen, and Asbjørn Mohr Drewes. Electroencephalography and analgesics. *British Journal of Clinical Pharmacology*, 77(1):72–95, 2014.

[10] Ewout Meijer and Peter Koppen. *Lie detectors and the law: The use of the polygraph in Europe*, pages 31–50. 01 2008.

[11] United Nations. Universal declaration of human rights. December 1948. Available at: https://www.un.org/en/about-us/universal-declaration-of-human-rights [Accessed 18 November 2022].

[12] Council of Europe: European Court of Human Rights. Guide on article 6 of the european convention on human rights - right to a fair trial (criminal limb), August 2022. Available at: `https://www.echr.coe.int/documents/guide\_art\_6\_criminal\_eng.pdf` [Accessed 18 November 2022].

[13] Roberto Domingo Pascual-Marqui et al. Standardized low-resolution brain electromagnetic tomography (sloreta): technical details. *Methods Find Exp Clin Pharmacol*, 24(Suppl D):5–12, 2002.

[14] Niklas Ravaja. Contributions of psychophysiology to media research: Review and recommendations. *Media Psychology*, 6(2):193–235, 2004.

[15] Selwin Gabriel Samuel, Tanushree Chatterjee, Himadri Thapliyal, and Priyanka Kacker. Facial psychophysiology in forensic investigation: A novel idea for deception detection. *Journal of forensic dental sciences*, 11(2):90, 2019.

[16] Anastasia Shuster, Lilah Inzelberg, Ori Ossmy, Liz Izakson, Yael Hanein, and Dino J Levy. Lie to my face: An electromyography approach to the study of deceptive behavior. *Brain and Behavior*, 11(12):e2386, 2021.

[17] Sabine Ströfer, Matthijs L. Noordzij, Elze G. Ufkes, and Ellen Giebels. Deceptive intentions: Can cues to deception be measured before a lie is even stated? *PLOS ONE*, 10(5):1–17, 05 2015.

[18] Louis G. Tassinary and John T. Cacioppo. *Handbook of Psychophysiology*. Cambridge University Press, 3rd edition, 2007.

[19] Marieke van Dooren, J.J.G. (Gert-Jan) de Vries, and Joris H. Janssen. Emotional sweating across the body: Comparing 16 different skin conductance measurement locations. *Physiology Behavior*, 106(2):298–304, 2012.

[20] Lei Wang, Pei-Ji Liang, Pu-Ming Zhang, and Yi-Hong Qiu. Ionic mechanisms underlying tonic and phasic firing behaviors in retinal ganglion cells: a model study. *Channels*, 8(4):298–307, 2014.

[21] Min Zhao, Nini Rao, and Chunlin Zhao. Theta synchronization and erps in deception detection. In *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5, 2019.

Beyond container security: Hybrid VM solutions

Valentin Ionita
valentin.ionita@aalto.fi

Tutor: Mario Di Francesco

Abstract

Current container security practices in the form of namespaces and control groups are considered insufficient or hard to use. For this reason, security profiles and modules have been used as an incremental improvement for container security, without being designed into the original approach. Hybrid virtual machines are the latest solution that offers a combined strategy between container usability and security, in which containers are run inside or abstracted to a virtual machine optimised to start fast and offer only a minimal layer around the container. The existence of such projects shows that the superior security model of virtual machines can be refitted to harden container isolation.

KEYWORDS: container isolation, security, virtual machine, hybrid container virtualization

1 Introduction

Application containers are a technology tightly linked with the rise in popularity of cloud deployment among developers, offering the possibility of migrating providers without affecting the application, generally bringing the DevOps side of a software business closer to the development team.

The significant ease of use coming from this *"write once deploy everywhere"* strategy made containers pervasive in all use cases, from packaging software for the advanced home user, to cloud-scale systems like Kubernetes, at their core being the concept of containers. While the security of this environment has been built around UNIX-like abstractions such as control groups (*cgroups*) and namespaces, their weaknesses resulted in different strategies needed to further security, such as security profiles or kernel security modules. Separate from the above, there has been a push to merge the usability of container technologies with the security offered by virtualization, running either a thin virtual machine layer around a container for hardware-enabled isolation or abstracting out the container runtime completely and offering it only as an API layer for controlling and monitoring the working process inside a virtual machine.

In the following, we will describe security-related projects and technologies, to help follow with further comparisons. First of all, the security model for containers currently relies, as described, on *cgroups*, which can monitor and limit resource usage, *namespaces*, which isolate processes from one another (and related resources like PIDs) and *seccomp-bpf*, a kernel security module that restricts syscall access. *KVM*, or kernel-level modules, is a technology that has been integrated into the Linux kernel that is used to turn a Linux instance into a bare-metal hypervisor for virtualization. *QUEMU/KVM* is a project forked from the general purpose emulator and virtualizer *QEMU* and enhanced with kernel-level virtualization capabilities coming from *KVM*. As such, it is a complex project with a functionally complete set of capabilities, and a codebase size to reflect it, over 1.4 million lines of code [1]. *gVisor*, a Google project, is a sandboxed container runtime, which isolates containers from the host without being a fully-fledged virtualization solution, but by presenting a similar interface to the virtualized targets (strategy named **paravirtualization**), being a good comparison target to the discussed solutions. *runC* and *crun* are two lightweight container runtimes following the OCI (Open Container Initiative) specification, the first being a project that started as part of Docker's source code for running and managing containers, and the second being a competitor to the first, with lower memory consumption and written in C, compared to Go. *Podman* (Pod Management tool) is a container engine including a runtime and all the necessary tools for developing and management of containers, but, differently from Docker, it does not have a model based on system daemons and can run completely

in non-root mode.

This paper focuses on such technologies, going into detail on their strategies and advantages in adopting the advantages of containers and the security of hardware-enabled virtualization. The two main projects we will approach are Kata Containers [5] and Firecracker [8] (with its control layer enabled by Ignite), both open-source codebases supported either by the community, through the Open Container Initiative, or by the currently largest on-demand cloud provider, AWS.

2 Solution basics

The two main solutions when it comes to hybrid VM containers are represented by Kata containers, which run containers themselves on a virtual machine and Firecracker, which runs the container image itself as an actual virtual machine with a dedicated kernel. While the first option is closer to a layered onion security model in which the nature of the container stays the same, it has been criticised for slow starts due to the need to wait for the virtual machine boot [10], so projects like Firecracker have created micro VMs with stripped down kernels that reduce startup times to under 125 ms [1].

A clear downside of using a dedicated minimal kernel is the possibility of not being able to run some types of applications [9], which impacts the use cases surrounding a less specialised deployment environment. When the functional needs require system calls that are not handled (or cannot be virtualized directly), then the bottleneck platform (in this case Kata container's solution) has to be replaced or, in case enough resources can be found on the user side for regular maintenance, a custom kernel can be compiled for specific operations.

2.1 Firecracker

Firecracker is a virtual machine manager that is used to control lightweight virtual machines called `microVMs` [8], and an alternative to QEMU. Using KVM-based virtualization and reducing the container image capabilities, in what they call a *minimal device model*, it improves the security model by removing part of the attack surface (as well as unused kernel functionality). Firecracker runs as a userspace program, using kernel-based virtualization (KVM) to separate the host OS from the custom OS of the

guest. Unlike other solutions like Kata containers, it does not also use QEMU or a modified version of it, deciding to reduce the possibility of security vulnerabilities from the breath of code by having a slim codebase written in Rust [1]. Firecracker is not characterized by a paradigm change, compared to its alternatives. That is because it comes with the specific purpose of running serverless functions, which need quick startup times and a large number of simultaneous containers, so Firecracker acts as an integration to many other products in the space, including Kata.

The minimal guest kernel results in low container startup times (lower than 125 ms) and high startup rates (up to 150 microVMs per second per host) [1]. Being a technology targeted as a quality-of-life improvement for clients of AWS, it is built with the scope of reducing the overhead of managing a single container as much as possible (reported at 5 MB), to provide a high-capability multi-tenancy. Outside network and storage abstractions and management, it also comes with rate-limiting capabilities, for easy management of individual containers. Their chosen security model specifies that all workloads are considered malicious at all times [1], with the point of them being isolated completely against one another and against the control level itself. Firecracker can be controlled through a REST API with access to configuration and rate-limiting. The second level of security is at the container level through a separate process called *jailer*. This *jailer* can additionally offer *cgroup* level controls inside the guest kernel, such as namespaces and limits for file sizes and file descriptor counts. However, its main development team has mentioned it is both less capable and offers fewer options than control groups themselves [1]. Its operation is more user-friendly than manually managing the mentioned resources, as it does a cleanup and startup sequence automatically to reduce the attack surface.

2.2 Kata Containers

Kata Containers are a lightweight VM solution that integrates with OCI containers seamlessly [6], and it is the merger of two previously existing technologies: *Intel Clear Containers* (container runtime solution that works on Intel processors with Intel® Virtualization Technology enabled, using *QEMU*/*KVM*), focused on short startup times (under 100ms) and Hyper's *runV* (a virtualized container runtime). One advantage of using Kata containers compared to Firecracker, coming from the utilization of *runV*, is the breadth of deployment targets, in various hardware archi-

tectures: ADM64, ARM, IBM z-series and p-series, outside the standard x86_64. Moreso, it also integrates with multiple hypervisors: QEMU, Cloud-Hypervisor and even the Firecracker hypervisor [5]. It enables multi-tenancy (when the same orchestrator can be shared by multiple tenants) and supports Container-as-a-Service capabilities, not only for Kubernetes but also for Swarm and OpenShift.

To settle performance and compatibility issues, post version 1 Kata has adopted containerd's shimv2 architecture [4], which is OCI runtime compatible but resolves various state-handling issues (like synchronisation). It is using a single runtime binary for all containers, and that communicates with the container manager through a socket using gRPC calls instead of regular API calls (that have no connection established). The final result is improving the runtime performance and eliminating commonly appearing byzantine problems on synchronization between deployments on multiple hosts. The virtualization offered is on two levels: the VM root specified by the hypervisor OS image and the VM container root, specified by the container base image. Each container environment is managed by the configured hypervisor, which manages a `kata-agent` and the workload (the actual deployed container). The `kata-agent` is the sole supervisor of all containers of a VM instance. Back on the host level, a `kata-runtime` acts as a utility for admin control. In terms of limitations, it is mostly in line with the *runC* implementation used by Docker (however that implementation does not follow the OCI standard specification itself), with some notable differences: host network usage is not accessible, host storage is not allowed unless extra privileges are offered to the container and *Podman* support is not yet achieved.

3 Security considerations

3.1 Threat scenarios

The overview of container security models in [11], outlines four use cases for container security, which should each be treated as if any level in this vertical deployment (host, container, and application) can either be honest (secure), semi-honest (would disclose information, but not give up control) or malicious (give up control to attackers): 1. protecting containers from applications 2. protecting containers from each other 3. protecting host

from containers 4. protecting container from host

Usage of hybrid VM containers clearly can defend more thoroughly, if not completely, in protecting containers from each other, and by the way of their construction, it can also avoid attacks coming from the use of untrusted images, or image misconfiguration. Making containers work inside VMs, as in the case of Kata containers, can protect the host (at least up to par with regular virtualization solutions) from malicious processes running at the application and container level. The core of the effort is then moved to securing the container runtime itself. If any vulnerabilities are to be found in the control level (such as unrestricted port mapping, which is a missing feature in the case of Kubernetes) of a hybrid container solution, then the virtualization does not bring any additional security. However, that is a much more limited attack surface than the complete OS interface to which a regular container's application has access.

4 The performance question

The rise of hybrid virtualized container technologies above the conceptual and scholastic realm shows that, at least in theory, the model brings enough security improvements over the currently insufficient mechanisms. Most of the previous discussion has been focusing on the problems that can arise from practical implementations of these concepts. These two layers of isolation, container isolation and virtualization, are the root cause of subsequent usability problems of such container runtimes: the startup and runtime speed of processes inside this setup is often insufficient. As a result, many roadblocks have been removed in the form of reduced feature sets and minimal operating system images, all to bring the application performance closer to the bare metal. To see if this strategy has been successful, comparative benchmarks can bring in quantitative data.

In a standard Kubernetes deployment, the default *runC* runtime outperforms Kata by 5 times [12] in a benchmark using real-life applications, at both executing the applications and container startup. Comparatively, in the same benchmark, gVisor has been found to deploy containers 2 times faster than Kata (version 1.9), but execute apps 1.6 times slower than it. This data comes from actual deployable (and quite popular) apps which are often used in a container form: Redis, an in-memory data store; Spark, a big data processing pipeline; and TeaStore, a microservice benchmark. In microbenchmarks focused on specific code paths, system calls or a sub-

set of application runtimes (Node, Ruby, Apache), one can see that even if optimisation is made for those functionalities to be comparable with the base *runC* (as in the case for gVisor [7]), I/O transfer limitations translate in overall application performance loss. Using Python performance standard benchmark (represented by the `pyperformance` package used as a standard for Python performance benchmarking), it has been shown [3] that Kata containers perform similarly compared to *crun* when talking about basic operations (math, logging, SQL execution), but that is not the case for more complex applications, where Kata performs up to 16% worse than `crun`. A deeper analysis shows that the filesystem model used inside Kata (*DAX*, Linux Direct Access filesystem, meant to be used to access host caches directly) is creating real-life performance issues, and it barely registers for compute-only jobs. These tests are done using *Podman* as a container manager and *crun* as a runtime as a comparison baseline for evaluating Kata containers, and only regard the runtime execution, not the container startup. In a CPU-bound performance benchmark (AV1 video decoding) [13], Kata QEMU performs around 4% worse than the bare-metal, almost identical to *runC*, while *gVisor* performs, depending on configuration, either the same as the host or significantly worse, by 13%, showing that, for computationally difficult tasks that need extra security, Kata containers are a reliable choice.

One practical strategy adopted by Firecracker in treating container resource usage has been `soft allocation`, which means that it will overcommit resources, but only offer to a container what they use (dynamically at runtime), not what has been requested [1]. This has helped in the target required by Amazon's Lambda in having 99.9% or higher uptime. In a comparison with *gVisor* and Linux containers [2], it has been shown that the CPU speed has not been affected by running on any of the above-mentioned runtimes compared to the host (comparison on up to 10 concurrent instances). However, Firecracker has shown more consistent bandwidth usage than *gVisor*, slightly higher network latency and significantly lower memory overhead footprint. Finally, while its write throughput is up to four times higher than *gVisor* depending on its configuration because it does not write data to persistent storage, its read throughput is significantly slower than *gVisor* (whose performance is comparable to the one of the host), because it is expensive to copy data from the host's space outside the container. This can be improved by having warm startups inside the `microVM`, with the data being kept in the slot's cache.

5 Conclusion

The hybrid VM solutions mentioned represent innovative approaches to fixing the security and performance problems of containers without harming usability. They have been steadily gaining traction as deployment options, while they are still missing out on important features and have a limited choice when it comes to actual supported images and bare-metal. This can be often overlooked by products which fit within those constraints, and it shows that hybrid VMs still need increased capabilities and adoption before they can become de-facto solutions for the security needs in container isolation.

References

[1] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. Firecracker: Lightweight virtualization for serverless applications. In *17th USENIX symposium on networked systems design and implementation (NSDI 20)*, pages 419–434, 2020.

[2] Anjali, Tyler Caraza-Harter, and Michael M. Swift. Blending containers and virtual machines: A study of firecracker and gvisor. In *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '20, page 101–113, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Fredrik Björklund. A comparison between native and secure runtimes: Using podman to compare crun and kata containers, 2021.

[4] OpenInfra Foundation. Kata containers architecture. https://github.com/kata-containers/kata-containers/tree/ee189d2ebebdef676d682d1c286476a8fe38148d/docs/design/architecture.

[5] OpenInfra Foundation. Kata containers - open source container runtime software, 2017. https://katacontainers.io.

[6] OpenInfra Foundation. Kata containers 1 pager, 2017. https://katacontainers.io/collateral/kata-containers-1pager.pdf.

[7] The gVisor Authors. Performance guide - gvisor. https://gvisor.dev/docs/architecture_guide/performance.

[8] Amazon Web Services Inc. Firecracker, 2018. https://firecracker-microvm.github.io.

[9] Ngadhnjim Plaku. Online Platform for Interactive Tutorials: Provisioning Virtual Environments. Master's thesis, Aalto University. School of Science, 2020. http://urn.fi/URN:NBN:fi:aalto-202008245174.

[10] Liz Rice. *Container Security*. O'Reilly Media, Inc, 2020.

[11] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7:52976–52996, 2019.

[12] William Viktorsson, Cristian Klein, and Johan Tordsson. Security-performance trade-offs of kubernetes container runtimes. In *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–4. IEEE, 2020.

[13] Xingyu Wang, Junzhao Du, and Hui Liu. Performance and isolation analysis of runc, gvisor and kata containers runtimes. *Cluster Computing*, 25(2):1497–1513, 2022.

Linguistics of automated cryptographic verification

Veli-Matti Rantanen
veli-matti.rantanen@aalto.fi

Tutor: Chris Brzuska

Abstract

This paper presents select language features from two programming languages – EasyCrypt and F – used for automated verification of cryptographic proofs. These languages differ in their capabilities and intended applications, yet have passing resemblances as functional programming languages. The most distinct language features of the languages are discussed and related to their use in cryptographic verification.*

*programming language, cryptography, automated proofs, verification, EasyCrypt, F**

1 Introduction

In the ever more digital and connected world of today, computer security is more relevant than ever. Fridges, cars, phones and other devices are all connected to each other as well as the Internet, enhancing their traditional features with ones such as remote monitoring and management. However, a larger number of online devices also present more opportunities for malicious actors to leverage.

Although developments in cryptography may not be seen as particularly impactful in the daily life of the average person, cryptographic algorithms

and schemes form the basis upon which the security of many of the aforementioned devices rely upon. As such, it is vital that the security of these algorithms is verified thoroughly. The security of an algorithm is verified with a *cryptographic proof* [1].

Traditionally, cryptographic proofs have been written by hand, but a recent trend has been so-called automated verification, where some or all of the proof is generated by a computer or the verification of the proof is automated [2]. Errors and vulnerabilities can be effectively found with a computer program looking for contradictions between the proof and its assumptions [3]. Automated verification requires a description of the algorithm and the associated assumptions, in a form that a computer can understand. Many domain-specific languages (DSL) have been developed to support different types of cryptographic verification; most tools come with their own languages [3].

This report presents language features of two state-of-the-art cryptographic verification tools: EasyCrypt and F*. A third tool – CryptoVerif – was reviewed, but was omitted from the comparison as it was found to bring nothing additional to the comparison. These three tools and their languages represent three different application domains within the field of computer security and cryptography, and were named as promising candidates for computational verification in [3].

EasyCrypt is both a tool primarily intended to aid in the construction and verification of cryptographic proofs [4], while F* is a general-purpose programming language that focuses on verification and, consequently, supports cryptographic proofs [5][6]. The third tool, CryptoVerif, focuses on protocol-related proofs and is capable of aiding the user by generating parts of the proof [7][8]. Both EasyCrypt [9][10] and F* [6][3] have been used to verify standardized cryptography, and support the extraction of a verified program from the proof.

Beyond this introduction, this report consists of three sections. The next two sections hereafter describe EasyCrypt and F*, respectively, their unique features in particular. The third section concludes the report with insight into whether and how the presented features could be made use of in the other language.

2 EasyCrypt

EasyCrypt is described in [1] as a framework for machine-verifying the security of cryptographic constructions and is notable for how closely it resembles traditional manual proofs. The language can be used in most cryptographic proofs, including for primitives, protocols and systems, and the framework supports the extraction of an implementation from the proof. [11][1]

Under the hood, EasyCrypt uses a proof engine that utilizes an SMT solver. This engine determines whether the proof is valid by generating a *goal* from the specification, which is then resolved by applying *tactics* that either map goals to subgoals or resolve them. If all goals can be resolved, then the proof is valid — otherwise, the proof is rejected [11]. A deeper inspection of the proof engine is not within the scope of this report.

The EasyCrypt language is, at its core, a functional programming language. This functional language is referred to as *core language* both in this report and other works [3]. From a practical perspective, this means that all functions are deterministic and never modify the program state, except by their return value [12].

EasyCrypt has two particularly notable features. The first feature is the *module*, which presents an embedded imperative programming language, and the second is the treatment of values as distributions.

2.1 Modules

As described in [11], EasyCrypt utilizes *modules* defined in an imperative language (i.e. each operation modifies the current state). These modules resemble the 'class' data type of object-oriented languages such as C++ and Python, with data and procedures enclosed within. The key difference is that only a single instance can ever exist for a given EasyCrypt module.

These modules are used to describe Games and Oracles, as commonly seen in cryptographic proofs [11]. The imperative language that these modules are written in resembles very closely pen-and-paper proofs [3], with the exception that all symbols are written with ASCII characters, with some becoming keywords and others becoming a multi-symbol operator – though this limitation is shared with the core language [11]. For the sake of illustrating the syntax of modules and the symbol mappings, figure 1 shows a simple example in both pseudocode and in EasyCrypt.

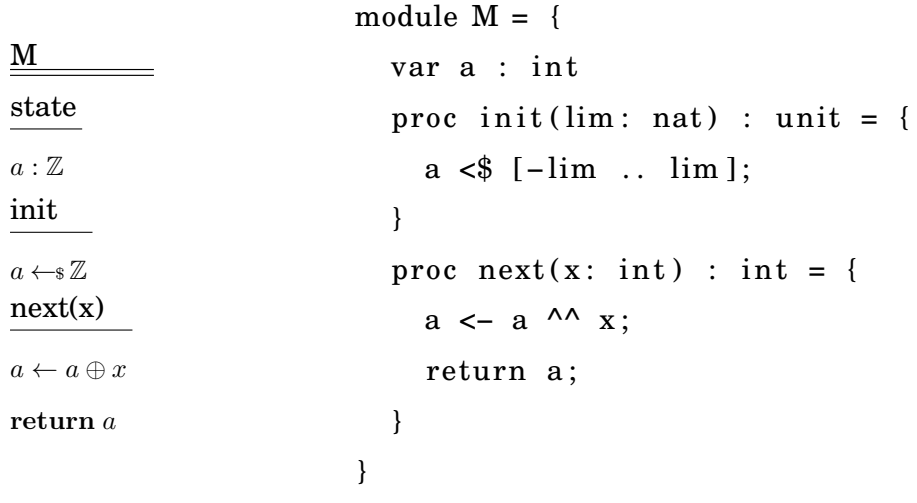


Figure 1. A simple package in pseudocode (left) and in EasyCrypt (right)

2.2 Values as distributions

One feature in EasyCrypt that is not strictly a feature of the language is the fact that its proof engine treats values as distributions of the declared type. This allows EasyCrypt to not only determine whether and adversary is successful, but also the probability of success. Consider the example given in figure 1 — the variable a is actually initialized to be a uniform distribution of given precision. Subsequent calls to `next()` then modify that distribution. [11]

3 F*

F* (F star) is a general-purpose functional programming language that lends itself to cryptographic verification. After a program written in F* is verified, it can be extracted to program code that is also verified. Beyond the verification capabilities of EasyCrypt, F* also supports verification of side-channel resistance. [3][5]

F* bears striking resemblance to EasyCrypt both superficially, with the syntactic and lexical elements of the two languages being nearly identical, with one exception being the imperative sublanguage of EasyCrypt. Another point of similarity is that the verification engine at the heart of F* also uses an SMT solver, i.e. applies tactics to resolve goals to prove that the program fulfills the programmer's specification. Tactics and goals are a type in F*, and can therefore be defined and used by the programmer. [5]

According to the developers, the driving force behind the language is the development of Project Everest [13], which aims to produce a fully-verified HTTPS implementation [6]. This project has also verified and implemented the QUIC record layer [14] and the Signal messaging protocol [6]. A byproduct of the project has been EverCrypt, which is a collection of cryptographic algorithms and data structures, such as SHA-3 and Merkle trees [6].

Similar to the other two languages, F* has a very flexible type system, extended further by its *propositions*, which are described at length in Section 3.1. Another peculiar feature of F* is its support for implicit code, described in Section 3.2. Lastly, one feature of F* that deserves a mention, but unfortunately did not fit into the scope of this work, is its remarkable metaprogramming capabilities — beyond defining types and operator overloads, F* supports defining new goals and tactics for the SMT solver [5].

3.1 Types and propositions

Most programming languages have some notion of types, as does F*. What sets the type system of F* apart from those of many other programming languages, is that an F* type is a value. For example, a function can accept a type as an argument and types can be used in expressions. It is also possible to compose new types from pre-existing types, e.g. vectors and tuples. [5]

F* incorporates proof-irrelevant propositions [5], a notion under which any two proofs of a given proposition are considered equal [15]. In F*, propositions resemble the boolean type in that they can be resolved to either 'true' or 'false', but may also be undecidable. It is particularly important to note that propositions are simply another type under the F* semantics, allowing the programmer to define functions that operate on or return propositions. Two examples of this are `forall` and `exists`, both of which are functions returning a proposition. [5]

Propositions can be attached to types and functions, and can also be used in assertions and assumptions, which define the constraints of the program. When attached to a type, a proposition allows one to define a novel type from a pre-existing type, such as the set of odd numbers or the elements of the fibonacci sequence. In functions, a proposition can be used to ensure that a recursive function always terminates – a notion which is required by the verification engine. [5]

3.2 Implicit code

F* allows the programmer to omit values or small parts of code to let the verification engine to determine the appropriate code. In the simplest case, a small piece of code can simply be omitted — for example, if an integer is assigned to a name, the type definition for that name can be omitted [5].

More importantly, there are two ways for a programmer to indicate that values should be accepted implicitly. Firstly, some or all of the arguments of a function can be declared as implicit by prefixing them with a '#'. Secondly, parts of expressions or statements can be omitted explicitly by utilizing a language concept coined by the developers as *program hole*, represented with a '_' token and implying a default behaviour. In the case of a function argument, an omitted argument would be inferred from the other arguments of the function — or possibly the return value.

4 Conclusions and discussion

While both languages are intended for different purposes, they are very similar in terms of syntactic and lexical elements. This similarity is likely due to a translation of mathematical symbols to an ASCII representation. To reiterate, the features highlighted in this report are: modules, values as distributions, the F* type and proposition system and implicit code. In this section, both languages are considered again – this time, considering if and how the features from the other language would suit the lanugage in question.

EasyCrypt appears to be very focused on game-based cryptography, with its embedded imperative language and interactive flow. However, it is not hard to imagine ways in which the propositions of F* could be used in the language. Rather than stating facts about variables in assertions (or assumptions), one could attach these facts to the variables themselves at the site of declaration. Given its smaller scope, however, it is unlikely to benefit to a significant degree from the code inference and metaprogramming capabilities of F*.

On the other hand, F* is a much broader language, and seems to have sufficient metaprogramming capabilities to be able to define the EasyCrypt language as an embedded DSL. Of the two presented features of EasyCrypt, the treatment of values as distributions is clearly more attrac-

tive – whether F* is reasonably capable of this kind of analysis, however, remains unclear. The language has a deceptively simple appearance and is difficult to fully grasp.

As a more general note, many papers discussing these languages and others utilize pseudocode with mathematical symbols, rather than the actual source code. This is counterintuitive and potentially even harmful, as then a paper showcasing the capabilities of a programming language ends up simply displaying the pen-and-paper solution, rather than a strong argument why the language excels at presenting said solution.

References

[1] Gilles Barthe, Francois Dupressoir, Benjamin Gregoire, Cesar Kunz, Benedikt Schmidt, and Pierre-Yves Strub. Easycrypt: A tutorial. *Foundations of security analysis and design vii*, pages 146–166, 2013.

[2] Shai Halevi. A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Paper 2005/181, 2005.

[3] Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. Sok: Computer-aided cryptography, 2021.

[4] Ran Canetti, Alley Stoughton, and Mayank Varia. Easyuc: Using easycrypt to mechanize proofs of universally composable security. In *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, pages 167–183, 2019.

[5] Nikhil Swamy. Proof-oriented programming in f*, not in print.

[6] Karthikeyan Bhargavan, Barry Bond, Antoine Delignat-Lavaud, Cédric Fournet, Chris Hawblitzel, Catalin Hritcu, Samin Ishtiaq, Markulf Kohlweiss, Rustan Leino, Jay Lorch, Kenji Maillard, Jianyang Pang, Bryan Parno, Jonathan Protzenko, Tahina Ramananandro, Ashay Rane, Aseem Rastogi, Nikhil Swamy, Laure Thompson, Peng Wang, Santiago Zanella-Béguelin, and Jean-Karim Zinzindohoué. Everest: Towards a verified, drop-in replacement of HTTPS. In *2nd Summit on Advances in Programming Languages*, May 2017.

[7] Bruno Blanchet. Cryptoverif: A computationally-sound security protocol verifier. *Tech. Rep.*, 2017.

[8] Bruno Blanchet and David Cade. Cryptoverif computationally sound, automatic cryptographic protocol verifier user manual, not in print.

[9] Jose Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Matthew Campagna, Ernie Cohen, Benjamin Gregoire, Vitor Pereira, Bernardo Portela, Pierre-Yves Strub, and Serdar Tasiran. A machine-checked proof of security for aws key management service. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 63–78, 2019.

[10] Jose Bacelar Almeida, Cecile Baritel-Ruet, Manuel Barbosa, Gilles Barthe, Francois Dupressoir, Benjamin Gregoire, Vincent Laporte, Tiago Oliveira, Alley Stoughton, and Pierre-Yves Strub. Machine-checked proofs for cryptographic standards: Indifferentiability of sponge and secure high-assurance implementations of sha-3. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1607–1622, 2019.

[11] Alley Stoughton, Francois Dupressoir, Pierre-Yvois Strub, Yi Lee, Christian Doczkal, and Benedikt Schmidt. Easycrypt reference manual, not in print.

[12] J. Hughes. Why Functional Programming Matters. *The Computer Journal*, 32(2):98–107, 01 1989.

[13] F* home page.

[14] Antoine Delignat-Lavaud, Cédric Fournet, Bryan Parno, Jonathan Protzenko, Tahina Ramananandro, Jay Bosamiya, Joseph Lallemand, Itsaka Rakotonirina, and Yi Zhou. A security model and fully verified implementation for the ietf quic record layer. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1162–1178, 2021.

[15] Jason Reed. *Proof irrelevance and strict definitions in a logical framework*. School of Computer Science, Carnegie Mellon University, 2002.

Draft 1
Container Network Security

Vishal Verma
vishal.verma@aalto.fi

Tutor: Mario Di Francesco

Abstract

In recent years, containers have emerged as a lightweight alternative to virtual machines, especially in the context of cloud computing. They are now considered an industry standard to implement and deploy mircoservices. Nevertheless, some companies are reluctant to switch to container technology because of its complex networking and security concerns. This paper focuses on different methods that can be deployed at different levels within containers to address network security concerns.

KEYWORDS: Network, Security, Firewall, Container, Docker, Kubernetes, Container network interface (CNI), Cluster, Virtual private cloud (VPC), Network policies, Pods, Nodes

1 Introduction

The majority of the Internet uses the client-server model, where the server hosts an application generally running on a virtual machine. However, in recent years, the demand for containers and their application has been popular. A container is a standardized unit of software that creates a virtualized environment for an application by encapsulating all its dependencies thus it can be easily deployed and run in any environment [5]. The

concept of containers has been introduced early as 2008, but its rampant growth has been directly linked to the introduction of Docker, an easy-to-use command line tool first released in the year 2013 as a free software [2].

Containers have become the industry standard for cloud computing. In fact, several containers can run inside a single cluster that hosts multiple applications. However, few companies oppose the idea of container technology and consider it insecure because of its complex network security concerns. Therefore, it is vital to address container security and specifically how containers interact inside a cluster to ensure the security of the system. A container is the smallest segment in a cluster, and securing the container helps enforce network security at a more granular level.

To facilitate the task of securing container networks, this paper reviews the current methods available to limit and secure container access. Furthermore, it discusses the challenges involved in implementing and enforcing network security methods in dynamic container configurations.

The paper is structured as follows: Section 2 briefly introduces the need for container network security; Section 3 covers different methods that help enforce container network security and their implementation; Section 4 addresses the related challenges; finally, Section 5 provides concluding remarks.

2 Background

Containers offer many advantages in terms of flexibility, scalability, and portability. As shown in Figure 1, fewer resources are required compared to traditional counterparts, virtual machines. Because of this their pop-

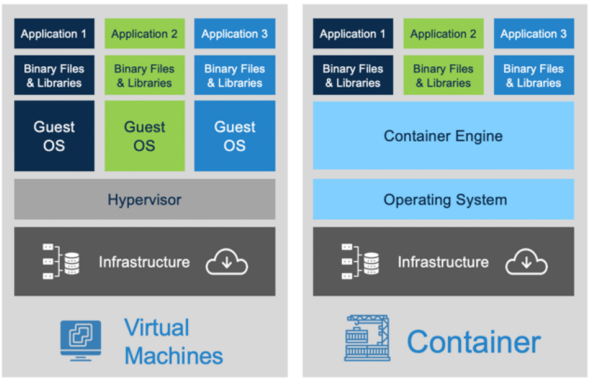


Figure 1. Virtual Machines vs Container. Source: [7]

ularity and adoption have significantly increased in recent years. Most companies based on cloud technologies are transforming their traditional infrastructure to utilize container technology [22]. The global software container market is expected to reach 9.643 billion US dollars by 2032, three times higher than in 2022 [6].

2.1 Kubernetes

Container design frameworks, such as Kubernetes have emerged as the preferred standard for dynamic, on-demand delivery of edge applications to end users and third parties [8]. Kubernetes provides a framework for persistently running distributed applications. It supports scaling and failover, offers deployment patterns, and more features, such as service discovery and load balancing, storage orchestration, automated rollouts and rollbacks, self-healing as well as secret and configuration management [5].

A Kubernetes cluster is a set of worker machines known as nodes. A node can be a physical or virtual machine depending on the cluster, which runs containerized applications by using pods. Kubernetes has mainly two types of nodes called: Control-plane nodes and Worker nodes [11]. Each cluster has at least one worker node. Pods are considered the smallest deployable units of computing that can create, deploy and manage using Kubernetes [5]. A pod is defined as a set of one or more containers that uses shared storage, network resources and contains instructions for how to run a certain container. In Kubernetes, a namespace provides means for isolating group of resources within a single cluster.

2.2 Kubernetes Networking Model

Once network security measures are deployed in a container they proactively restrict unauthorized communication and prevent applications from being compromised [1]. All pods and nodes can communicate with other pods without having to use Network address translation (NAT). Each pod is allocated a unique IP address across the entire cluster. Given so there are three different types of networking as illustrated in Figure 2:

Container-to-Container Networking. Inside a pod, there can be multiple containers, and communication between them is straightforward. Containers inside a pod share the same network namespace and efficiently the same virtual network stack, i.e., network interfaces and net-

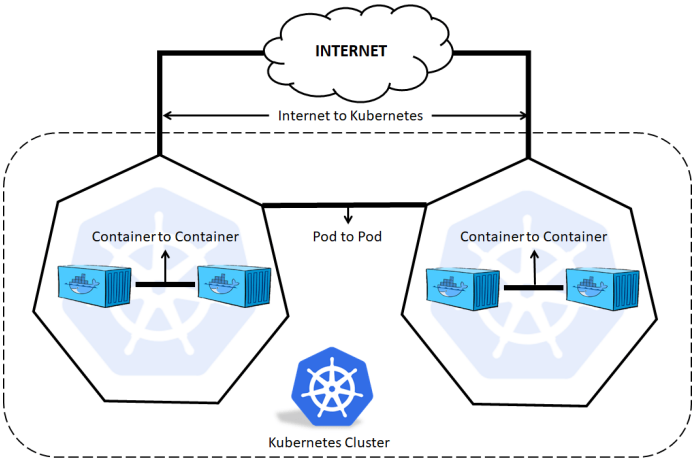


Figure 2. Kubernetes Networking Model. Source: [19]

work properties. A compromised container has access to other containers that are running inside the same pod [15].

Pod-to-Pod Networking. Communication between pods leads to two scenarios: pods communicating within the same worker node or to different worker nodes. Pods within the same worker node have a virtual Ethernet interface and a bridge that handles communication using the address resolution protocol. Hence, multiple pods inside a worker node can exchange network packets through the virtual bridge. In case the pods are in different worker nodes, there is an overlay network that keeps track of the network properties of the pods. This is used to manage and update a routing table, which references to subnet and node for a given pod. The virtual bridge sends the packets to the overlay network, which exchanges data packets from the other pod in a different worker node [15].

Pod-to-Service Networking. A Kubernetes service is a logical abstraction that exposes a group of deployed pods in a cluster as a network service [18]. Kubernetes supports three distinct types of services:

- *ClusterIP:* This service is used to assign a cluster-wide unique IP address to the application, which makes the application's service reachable only from inside the cluster.

- *NodePort:* It is used to assign the service to a static port on each node within the cluster. The service created can be accessed from outside the cluster using the node's IP address and the port number that was statically assigned.

- *Load Balancer:* Kubernetes exposes the service using a cloud provider's load balancer. Once the request is received at the cloud provider's load balancer, it redirects the request to the subsequent NodePort service, which routes it to a ClusterIP service to reach a certain application [15].

2.3 Container network interface (CNI)

In Kubernetes, a network policy is used to define certain network parameters on how a specific pod is authorized to communicate with other networking components i.e., pods and services. These policies are not enforced by Kubernetes, but by a network plugin that implements the container network interface (CNI). CNI plugins running on each node are used to retrieve network policies stored in the database and enforce them. Creating a network policy without the CNI plugin will not affect cluster traffic.

A CNI plugin is an effective way to enforce and manage network policy on a Kubernetes cluster. There are various CNI agents, such as calico, Flannel, Kube-route, weave, and others [13]. Different CNI agents operate on different layers of the Open Systems Interconnection (OSI) network model. These offer features, such as low latency, fewer resource requirements, encryption support, and several other security features [17].

3 Methods and implementation

A firewall is the main method used to restrict container network communication. A Firewall can be introduced at different layers of the OSI network model, i.e., transport, network, and data link layer, which can be enforced by CNI plugins. Deploying a container on a virtual private cloud can also help to strengthen container network security [20]. A firewall can also be introduced on an application layer using Application Programming Interface (API) firewalls also termed WAF or web application firewalls.

3.1 Firewall

A firewall can help limit the traffic that flows to and from a certain set of containers. A container firewall is generally referred to as network policies in Kubernetes, which are then enforced by the CNI plugin. The idea is to restrict container network traffic to and from approved endpoints

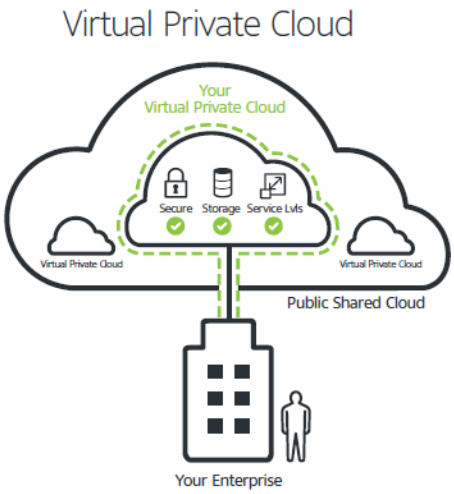


Figure 3. Virtual Private Cloud. Source: [4]

with a set of rules defined by network policies. A firewall is not exactly a new concept, but using it in conjunction with other security tools and practices can be highly effective.

3.2 Deploying containers in Virtual Private Cloud

Deploying containers with Virtual Private Cloud (VPC) can increase security. Although a VPC is part of the public cloud, it is isolated. Therefore, the data and resources do not associate with other customers in the cloud [21]. Users can have full privilege over how to allocate resources and access. VPC also provides additional benefits, such as performance upgrades, agility, and even cost reduction compared to the public cloud [14].

3.3 Network Policies in Kubernetes

Network policy can be created, managed, and enforced using any number of CNI plugins, some standardized practices should be followed to ensure a certain level of security.

Default denies: This follows the principle of least privilege assigned to restrict any unwanted traffic. Set up a network policy that denies all incoming traffic by default in each namespace, and only allows traffic to permitted endpoints.

Default denies egress: Egress is also referred to as outgoing traffic. One compromised container can infect its neighboring containers, to restrict this deny all outgoing traffic by default in each namespace, and add

network policies for authorized egress traffic.

Restrict pod-to-pod traffic: Pods represent applications, and communication between pods should be limited, thus only required pods can communicate with other pods to ensure application functionality.

Restrict ports: Exposed ports can lead to a compromised system. Restrict all traffic in order to only documented ports with adequate reasons that are authorized to accept incoming and outgoing traffic.

4 Challenges

Network security issues have been cited as one of the major problems preventing many companies from successfully adopting container technology [22]. Legacy storage architectures can be complex, and they lack API functionality to compete with modern automation [10]. Storage cannot be easily scaled with applications and in turn, lower the performance. Implementing the right set of tools with container technology is vital and failing this can lead to more challenging problems [9].

Different CNI plugins provide distinctive features and levels of performance. Creating complex network policies and enforcing them using the wrong CNI plugin can add latency to the overall network performance [12]. The Performance of CNI plugins depends on a variety of factors, such as resource allocation, network architecture, and protocols. According to one of the studies that focused on finding the CNI plugins with higher throughput and less consumed resource-usage index (CPU and memory) in Kubernetes. In an experimental environment for exchanging data between hosts, the Kube-Router achieved the highest throughput for Transmission control protocol (TCP) data transfers at over 90% of the nominal link bandwidth, followed by Flannel and Calico [16].

Monitoring and maintaining run-time container network security can be challenging. It requires additional resources and attaching sidecar containers with unlimited access and control over deployed resources. Malicious tools with access can compromise container security. Since there is a need for a centralized system to manage all sidecar containers deployed for various monitoring and logging tools, an Open policy agent (OPA) can be a use full tool that provides seamless support for policy management.

OPA is an open-source engine that supports Policy as Code, which can track the standard development lifecycle and provide policy change history [3]. OPA integrates with a variety of tools, allowing many parts of

the system to use a standard policy language rather than relying on multiple vendor-specific technologies. For example, OPA can be deployed as a host-level daemon or sidecar container. OPA provides centralized policy management accessible via an API, runs in parallel with application services, and is designed to work with any type of JSON input. Therefore it can be easily integrated with any tool that produces JSON output.

5 Conclusion

As container technology becomes the industry standard for cloud computing, there needs to be focus on container network security, as this poses a significant challenge to the adoption of container technology. This paper has reviewed different network security measures to provide the best model for container network security, as there are open-source solutions, such as open policy agents, CNI plugins, and third-party container security tools. These open-source solutions can help mitigate the current challenges when setting up complex networking models and network security concerns. This approach can help container technology become more secure, feasible, and easy to set up and could encourage more companies to transition to container technology.

References

[1] Container security 101: Understanding the basics. Accessed: 20-10-2022. [Online]. Available: https://www.paloaltonetworks.com/resources/guides/prisma-container-security101.

[2] Docker. What is a container? Accessed: 20-10-2022. [Online]. Available: https://www.docker.com/resources/what-container/.

[3] Introduction of Open Policy Agent. Accessed: 20-10-2022. [Online]. Available: https://www.openpolicyagent.org/docs/latest/.

[4] Introduction of virtual private cloud (VPC). Accessed: 20-10-2022. [Online]. Available: https://tudip.com/blog-post/introduction-of-virtual-private-cloud-vpc/.

[5] Kubernetes. Accessed: 20-10-2022. [Online]. Available: https://kubernetes.io/.

[6] Software containers market. Accessed: 20-10-2022. [Online]. Available: https://www.futuremarketinsights.com/reports/software-containers-market.

[7] Eric Arrington. What's the difference between VMS containers? Accessed: 19-10-2022. [Online]. Available: https://akfpartners.com/growth-blog/vms-vs-containers.

[8] 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, 2021.

[9] Marco De Benedictis, Antonio Lioy, and Paolo Smiraglia. Container-based design of a virtual network security function. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 55–63, 2018.

[10] Surya Kant Garg, J. Lakshmi, and Jain Johny. Migrating VM workloads to containers: Issues and challenges. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 778–785, 2018.

[11] Md. Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, and Akond Rahman. Xi commandments of Kubernetes security: A systematization of knowledge related to Kubernetes security practices. In *2020 IEEE Secure Development (SecDev)*, pages 58–64, 2020.

[12] Zhuangwei Kang, Kyoungho An, Aniruddha Gokhale, and Paul Pazandak. A comprehensive performance evaluation of different Kubernetes CNI plugins for edge-based and containerized applications. 2021.

[13] Narūnas Kapočius. Performance studies of Kubernetes network solutions. In *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pages 1–6, 2020.

[14] Wen-Hwa Liao and Shuo-Chun Su. A dynamic VPN architecture for private cloud computing. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pages 409–414, 2011.

[15] 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.

[16] Siska Novianti and Achmad Basuki. The performance analysis of container networking interface plugins in Kubernetes. In *6th International Conference on Sustainable Information Engineering and Technology 2021*, page 231–234, New York, NY, USA, 2021.

[17] Youngki Park, Hyunsik Yang, and Younghan Kim. Performance analysis of CNI (container networking interface) based container network. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 248–250, 2018.

[18] Arnaldo Pereira Ferreira and Richard Sinnott. A performance evaluation of containers running on managed Kubernetes services. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 199–208, 2019.

[19] Prabhu Rajagopal. Understanding of Kubernetes networking models. Accessed: 20-10-2022. [Online]. Available: https://digitalvarys.com/kubernetes-networking-models.

[20] Liz Rice. *Container Network Security*. O'Reilly Media, Inc., 2020.

[21] Dan Sullivan. *Networking in the Cloud: Virtual Private Clouds and Virtual Private Networks*, pages 337–360. 2019.

[22] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7:52976–52996, 2019.

Efficient methods for uncertainty in Deep learning

Yejun Zhang
yejun.zhang@aalto.fi

Tutor: Trung Trinh

Abstract

Bayesian neural networks (BNNs) are neural networks (NN) whose weights are represented by a distribution. Compared to a deterministic NN, BNNs theoretically can produce more accurate and better-calibrated predictions. However, due to the sheer amounts of parameters in modern NNs, BNNs are difficult to train and require massive amounts of computation. Methods have been proposed to improve the efficiency of BNNs, such as Markov chain Monte Carlo and variational inference. In this paper, we will survey the motivation of BNNs and two basic Bayesian inference algorithms.

KEYWORDS: Machine learning, Deep learning, Bayesian neural network, Uncertainty.

1 Introduction

Deep neural networks have obtained astonishing performance on machine learning tasks [1] and have been increasingly popular in many domains such as image classification, video recommendation, social network analysis, multimedia concept retrieval, text mining. Despite the outstanding performance in supervised learning, neural networks are not good at measuring uncertainty and prone to overfitting, which adversely affects their

generalization capabilities [2]. Many approaches have been put forward to mitigate the issue. The majority of those use a Bayesian formalism [3], where the parameters are given prior distributions and update with data, and the posterior distributions over the parameters is computed.

Bayesian neural networks(BNNs) are referred to a neural network where weights are probability distributions, as shown in figure 1. Generally, Bayesian inference is intractable due to doubling the number of parameters and complex integration. However, variational approximation can be used to exact Bayesian updates to make exact integration.

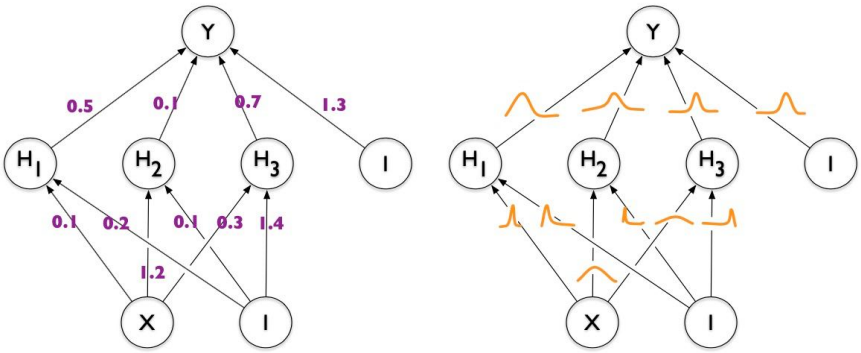


Figure 1. Left: each weight has a deterministic value. Right: each weight is a distribution.

The paper is constructed as follows. Section 2 introduces basic structure of BNNs. Section 3 presents the advantages of BNNs. Finally, section 4 will aim to contain two efficient Bayesian inference methods.

2 Bayesian neural networks

A BNN uses approximate Bayesian inference for uncertainty estimation[4]. For a supervised learning task, the goal of deep learning is to fit a neural network $y = f_\theta(x)$ with parameters θ to dataset $D = (x_n, y_n)_{n=1}^N$, corresponding to a Maximum Likelihood Estimation (MLE) of parameters:

$$\theta^* = \arg\max_\theta \mathbb{E}_{(x,y)\ D}[\log p(y|x,\theta)], \tag{1}$$

with the likelihood term defined as

$$\text{for regression: } p(y|x,\theta) = \mathcal{N}(y; f_\theta(x), \sigma^2 I) \tag{2}$$

$$\text{for classification: } p(y|x,\theta) = Categorical(logit = f_\theta(x)) \tag{3}$$

In BNN, however, the parameters θ are regarded as random variables,

and we conduct approximate Bayesian inference, and we define a prior distribution $p(\theta)$, which leads to a posterior with Bayes' rule (under the i.i.d data setting):

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)},\ p(D|\theta) = \prod_{n=1}^{N} p(y_n|x_n, \theta) \tag{4}$$

Supposing knowing $p(\theta|D)$, in the prediction time, given new test input x^*, we can obtain:

$$p(y^*|x^*, D) = \int p(y^*|x^*, \theta)p(\theta|D)d\theta \tag{5}$$

However, $p(\theta|D)$ or $p(y^*|x^*, D)$ can not directly be computed. This is where approximate Bayesian inference distribution comes in. The method solves the problem in the following three steps: First, design an approximate posterior: design a distribution family $\mathcal{Q}$ such that for each $q(\theta) \in \mathcal{Q}$, we can compute its density given any θ, and $q(\theta)$ is easy to sample from; second, fit the approximate posterior: find the best q distribution in $\mathcal{Q}$ so that $q(\theta) \approx p(y^*|x^*, D)$ well according to some criteria; third, approximate predictive inference with Monte Carlo: approximate $p(y^*|x^*, D)$ by replacing the exact posterior $p(\theta|D)$ with $q(\theta)$ and estimating the integral with Monte Carlo:

$$p(y^*|x^*, D) \approx \int p(y^*|x^*, \theta)q(\theta)d\theta \approx \frac{1}{K}\sum_{k=1}^{K} p(y^*|x^*, \theta_k),\ \theta_k \sim q(\theta). \tag{6}$$

It is still challenging to find an approximation $q(\theta) \approx p(\theta|D)$, we will discuss more in section 4.

3 Advantage of BNNs

There are mainly three advantages of BNNs. First, BNNs provide an approach to quantify uncertainty since BNNs offer better calibration than neural networks [5]. Second, BNNs can differ between the epistemic uncertainty and the aleatoric uncertainty [6]. Third, BNNs have good robustness and generalization. In theory, BNNs are more robust against out of sample data.

3.1 Uncertainty

In Bayesian modeling, there exist mainly two types of uncertainty[6]. Epistemic uncertainty, also named model uncertainty or systematic un-

certainty, is resulting from a lack of knowledge and thus can be reduced by training with more data. This uncertainty refers to the uncertainty of model weights themselves. Each time training the model can produce slightly different results. Aleatoric uncertainty, also called statistical uncertainty, means the uncertainty of model outputs.

3.2 Regression curves

A concrete example can show how BNNs measure uncertainty. The data can be generated synthetically from curve:

$$y = x + 0.3sin(2\pi(x + \epsilon)) + 0.3sin(4\pi(x + \epsilon)) + \epsilon \tag{7}$$

where $\epsilon \sim \mathcal{N}(0, 0.02)$. Figure 2 represents two concrete examples of training a neural network. On the left, Bayesian Neural Networks have effect on predictions: when there are no or few data, the confidence intervals become larger, meaning more uncertainty. On the right, it is the standard neural network. In this case, BNNs can tell where is highly uncertain as there are no nearby data, as opposed to the classic neural network. In BNNs, it can be seen that when the data points appear intensively, the predictions are more deterministic near the those data points. That is also how BNNs measure uncertainty.

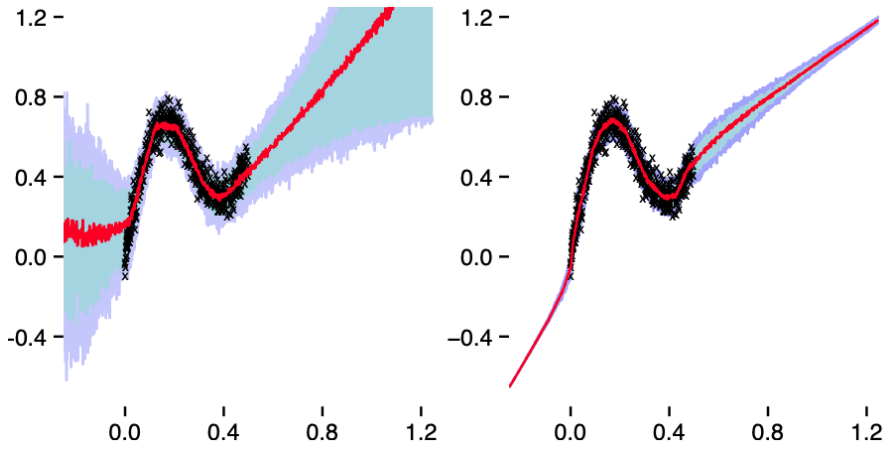


Figure 2. Black points mean training datapoints. Red lines are median predictions. Blue/purple region is interquartile range. Left: BNN. Right: standard neural network.

4 Bayesian inference algorithms

This section reviews two basic inference methods. The first one is Markov chain Monte Carlo(MCMC) methods[7] mainly drawing samples to evalu-

ate the integral directly. The second is variational inference[8] converting integration problem into optimization problem.

4.1 Markov chain Monte Carlo(MCMC)

Markov chain is a sequence of random datapoint which only depend on the previous datapoint. The idea is to use a Markov chains to perform Monte Carlo estimate.

MCMC methods can be one of the best and most popular solutions for sampling from exact posterior distributions [9]. But not all MCMC algorithms are suitable for BNNs. For example, Gibbs sampling is popular in some areas but is unsuited for BNNs. Metropolis-Hastings algorithm is the most relevant MCMC method for BNNs, as it does not need any prior information about the probability distribution to sample. This algorithm can compute posterior distribution easily in addition to the evidence term However, the Metropolis-Hasting algorithm has its own drawback. When the proposal distribution is too large, the rejection rate will be too high, leading to low efficiency. When the proposal distribution is too small, the samples will be more autocorrelated. Hamiltonian Monte Carlo algorithm (HMC) has been put up to solve this impact. HMC belongs to Metropolis-Hasting algorithms family. It uses Hamiltonian dynamic, widely used in computational physics and chemistry. The motivation for HMC is to sample the state space more efficiently, so that larger movements from the current state could be made in one step than what is possible in Metropolis-Hastings sampling. This comes at the price of increased computation per time step.

In MCMC algorithms, there usually exist a burn-in time before calculating, as Markov chain does not maintain convergence all the time. Moreover, the sequential samples maybe autocorrelated, meaning that lots of samples have to be generated and subsampled to get independent samples.

4.2 Variational inference

MCMC algorithms are the best choices for sampling from a exact posterior. However, they are expensive and don't scale to large datasets. For example, HMC which is still 'gold standard' for doing accurate posterior inference in BNNs, is inherently a batch algorithm and updates with the entire training dataset, which has made them less popular for

BNNs. Variational inference[8], which has better scalability than MCMC algorithms and cheap, received huge popularity. Compared with MCMC, variational inference is a less accurate method. Instead of directly sampling from the exact posterior, the variational inference is to construct a variational distribution, parameterized by designed parameters. Those parameters are learned in order that the variational distribution is as close as possible to the exact posterior distribution. To measure the closeness of those two distributions, the Kullback–Leibler divergence is the most common method[10], mainly based on Shannon's information[11].

Bayes by backpropagation

Variational inference provides an approach for Bayesian inference, but in order to train it as classic neural network, it needs some modifications. The main challenging is that stochasticity stops backpropagation from functioning at the internal nodes of a network [12]. Bayes-by-backprop [13] has proposed to tackle this problem and has been seen as a breakthrough in probabilistic deep learning. Variational inference is implemented by combining with a local reparametrization trick[14] to ensure backpropagation work.

The local reparametrization trick works in this way. It moves the parameters to be learnt out of the distribution function for any weight. For example, in a Gaussian distribution, μ is the mean and σ is the standard deviation ,and ϵ is one sample, multiply it with the standard deviation σ and add the mean μ. By doing so, two parameters of interest are incorporated in every weight value and both calculate the derivative of it and rewrite into a probability distribution. Each step of optimization proceeds as follows:

$$\Delta\mu = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu}$$

$$\Delta\sigma = \frac{\partial f}{\partial w}\,\frac{\epsilon}{\sigma} + \frac{\partial f}{\partial \sigma}$$

$$\mu \leftarrow \mu - \alpha\Delta\mu$$

$$\sigma \leftarrow \sigma - \alpha\Delta\sigma$$

$$\theta^{opt} = (\mu^{opt}\,, \sigma^{opt})$$

Thus, the usual gradients found by backpropagation can be calculated as normal, then scale and shift them as above.

5 Conclusion

This paper presents overview of BNNs. Although the idea is simple, just training a neural network with probability distribution on weights, Bayesian inference is still challenging in time-efficient and data-efficient. The paper also covers the advantages of BNNs and two basic inference methods, MCMC and variational inference. Those methods have been seen as huge process and made BNNs a hot topic. Another challenging is that there are massive amounts of parameters in BNNs, making BNNs difficult and expensive to train. However, some methods has put up and achieved notable performance and we hope this paper can provide an overview of basic and classic methods.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013.

[3] J. M. Bernardo and A. F. Smith, *Bayesian theory*, vol. 405. John Wiley & Sons, 2009.

[4] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on bayesian neural networks—a tutorial for deep learning users," *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[5] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," vol. 32, 2019.

[6] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[7] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," 1970.

[8] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[9] R. Bardenet, A. Doucet, and C. C. Holmes, "On markov chain monte carlo methods for tall data," *Journal of Machine Learning Research*, vol. 18, no. 47, 2017.

[10] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[11] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[12] W. L. Buntine, "Operations for learning with graphical models," *Journal of artificial intelligence research*, vol. 2, pp. 159–225, 1994.

[13] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*, pp. 1613–1622, PMLR, 2015.

[14] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," *Advances in neural information processing systems*, vol. 28, 2015.

CS-E4000 Seminar: Misinformation classification and community detection on Twitter

Yue Wan
yue.wan@aalto.fi

Tutor: Ali Unlu

Abstract

KEYWORDS: social media, misinformation, health, data mining

1 Introduction

Twitter, as one of the most popular social media in the world, is changing the way people communicate. An increasing number of people choose Twitter as a platform to read and share the breaking news and their opinions in public [11]. The convenience and openness of the social media enable and motivate people to communicate freely online, but many inaccurate statements also cause new problems. Recently, the wide spread of misinformation on social platforms has received much attention from the public and academia. Especially, the health information in Twitter is considered a major issue [6].

According to WHO, there are over 6000 people hospitalize and 800 people even have died due to the misinformation in first 3 months of 2020. The misinformation ranges from the suspicions about the COVID-19 could alter the structure of human DNA to the vaccine will make people sterile. The real danger presented by the misinformation is even worse than the virus itself. An ongoing project between Aalto University and Finnish

Institute for Health and Welfare (THL) is focusing on the misinformation detection related to Covid 19. The aim of this paper is to review the existing misinformation detection methods and give insight based on the current research.[7].

This paper will introduces the classification, detection and intervention methods in term of the misinformation in Twitter and discuss the possible future challenges. The rest of the paper is organized as follows. Section 2 defines the different types of misinformation. Section 3 analyses the main parts and characteristics of misinformation posts. Section 4 presents the main classification and detection methods based on the characteristics, which focus on the accuracy and earliness. Section 5 presents some concluding remarks.

2 Misinformation Definition

To investigate misinformation in social media, this section organizes different type of misinformation according to the intention of user spreading misinformation.

Unintentionally-Spread Misinformation:

Some misinformation is some misinformation which is not intended to deceive its recipients. Due to their trust of information sources, people tend to spread such information to their friends, family or colleagues in their social network. Instead of deceiving, people usually want to help and inform their social network of the underlying issues.[9]

Intentionally Spread Misinformation:

Some rumors and fake news are created and spread intentionally by malicious users who aim to deceive their recipients, cause public anxiety and mislead. There are usually a group of spreaders or writers behind the popularity, who have clear purpose and agenda to promote misinformation for improper profit.[9]

3 Model Information

The Social network users are defined by the content they create and spread. Thus, by using this feature, modeling their content information is an effective method to identify the misinformation spreaders.[9] This section is to analysis the features of the misinformation based on its content and

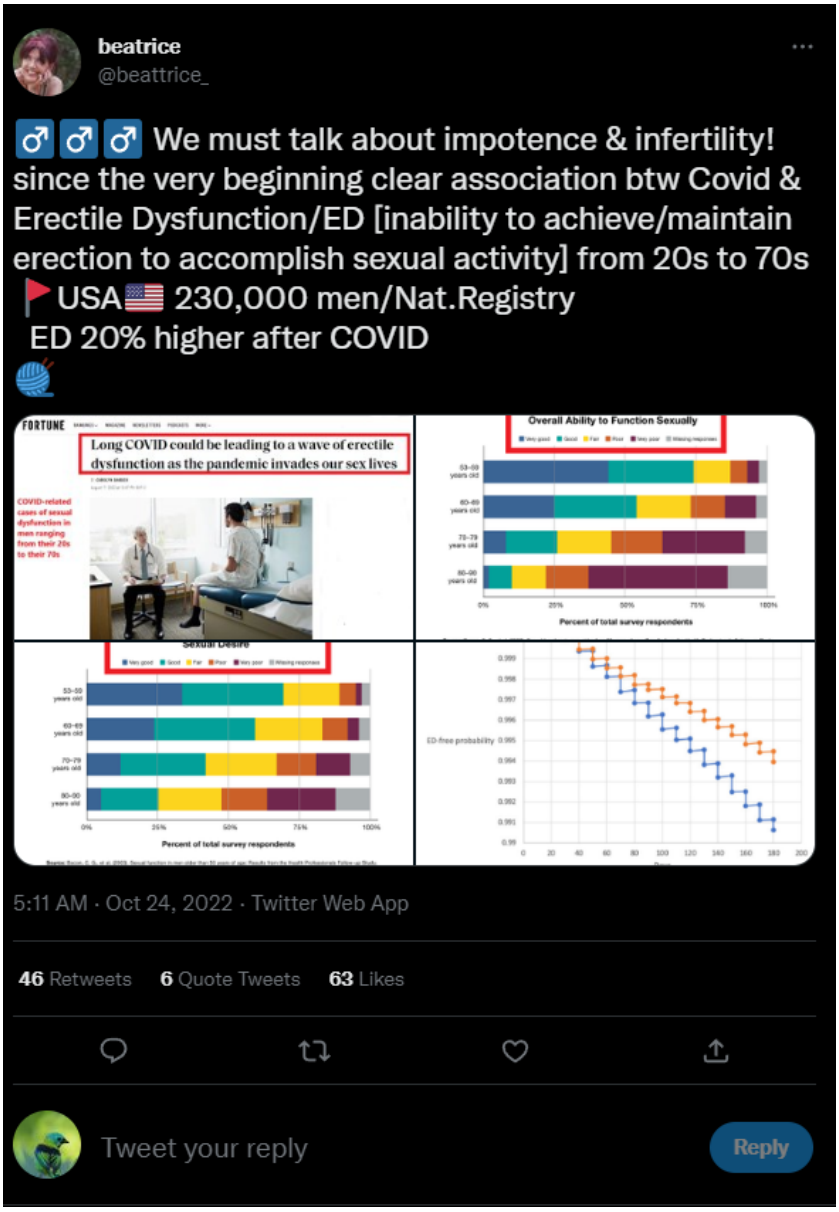


Figure 1. An example of misinformation in Twitter

aims to identify distinguishing characteristics.

Figure 1 illustrates an example of misinformation in Twitter. From the figure, it shows that a tweet consists of four sections: misinformation spreader, content of misinformation, context of misinformation and propagation of misinformation.

The section for the misinformation spreader consists of the profile and account description and they can be used to identify the malicious signals. For instance, the name length and longevity of accounts are jointly used for spreader detection [5].

The misinformation content consists text, URL, image and video which are the main carriers of disinformation. In early research [5], the text classifiers were mainly used to classify malicious users. However, s re-

search has progressed, the external links in content, which direct normal users to websites, are also seen as the key to detection. For example, although the content of a post is varied, researchers found that the embedded links in their posts may lead to the same target. By recording the links of these web pages, the data can be used to distinguish the authenticity of the information.

In terms of context and propagation of misinformation, such as release time and forwarding number, can also be used as key factors for detection. For example, during the 2020 U.S. presidential elections, the posting behavior of political misinformation often contained long hibernation and burst peaks [2].

4 Detection Methods

The detection of misinformation is generally regarded as a classification problem, which classifies media according to their content and distinguishes true from false [9]. However, misinformation posts are deliberately made seemingly real and accurate, which increases the difficulty of detection based on the content. Therefore, based on these characteristics, this paper classifies the detection methods into four categories: content-based misinformation detection, context-based misinformation detection, propagation-based misinformation detection and early detection of misinformation.

4.1 Content-based misinformation detection

Content-based misinformation detection aims to identify the content of information directly, and distinguish the authenticity by extracting keywords in information for classification detection.This method usually collects tweets and hashtags from social media as data-sets and and then train a text classifier based on the collected content and labels [1]. Due to the huge demand for data sets, this method is suitable for detecting popular information and news.

4.2 Context-based misinformation detection

Context-based misinformation detection is to identify information tweets based on their release time and geographical location, and identify them by capturing the features of time and location. This approach is often

used in combination with other approaches [10].

4.3 Propagation-based misinformation detection

Propagation-based misinformation detection is to detect information based on the propagation patterns, such as specific user groups. Misinformation can be identified by searching the commonalities among the audiences of misinformation. Since many users will follow back when they are followed by some users, the spreader can form a numbers of groups by connecting with legitimate users. Therefore, the current research is focusing on how to use the source of network information to identify the misinformation spreaders [8].

4.4 Early detection of misinformation

The purpose of early detection of misinformation is to identify and process information in the pro-phase stage. This focuses on the speed and efficiency of error information detection and blocking misinformation before it spreads widely [4]. However, in the early stages of misinformation dissemination, the spread of postings is usually fragmented and takes a long time to develop. To shorten the waiting time, the researchers suggested that three types of structural information, including hashtags, web links and content similarity, could be discussed to help analyze the authenticity of information.[3]

5 Conclusion

As social media platforms make everyone increasingly connected, misinformation is also spreading faster and more widely than ever before, which impacts the real world. Fake tweets can have a negative impact on communities and even trigger catastrophic results. The purpose of this paper was to explore the causes and characteristics of misinformation, and through the review of previous related research, to detail the existing work of identifying misinformation. Meanwhile, through the further discussion of different detection methods, more accurate, efficient and optimized solutions can be found to overcome this "information pandemic".

References

[1] Ponnurangam Kumaraguru Aditi Gupta, Hemank Lamba and Anupam Joshi. *Detection and analysis of 2016 us presidential election related rumors on twitter*. PhD thesis, 2017.

[2] H. Mao J. Bollen and A. Pepe. Determining the public mood state by analysis of microblogging posts. *ALIFE*, pages pages 667–668, 2010.

[3] L. Wu J. Sampson, F. Morstatter and H. Liu. Leveraging the implicit structure within social media for emergent rumor detection. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, page pages 2377–2382, 2016.

[4] D.Kim J.Kim and A.Oh. Homogeneity-based transmissive process to model true and false news in social networks. Technical report, November 2018.

[5] B. D. Eoff K. Lee and J. Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. *ICWSM*, 2011.

[6] Yang-Jun Li and Christy M.K. Cheung. Health misinformation on social media: A literature review. Technical report, 2018.

[7] Victor Luckerson. Fear, misinformation, and social media complicate ebola fight. Technical report, 2014.

[8] M. Mccord and M. Chuah. Spam detection on twitter using traditional classifiers. *international conference on Autonomic and trusted computing*, page pages 175–186, 2011.

[9] Liang Wu, Fred Morstatter, Kathleen M. Carley, and Huan Liu. Misinformation in social media: Definition, manipulation, and detection. Technical report, 2019.

[10] H. Wang Z. Chu, S. Gianvecchio and S. Jajodia. Who is tweeting on twitter: human, bot, or cyborg? Technical report, 2010.

[11] H. Guo Y. Zhang Y. Wang Z. Jin, J. Cao and J. Luo. *Faking sandy: Characterizing and identifying fake images on twitter during hurricane sandy*. PhD thesis, 2013.

Low Latency, Low Loss, Scalable Throughput (L4S) Protocol

Monday 21st November, 2022
Zunaira Salam
`zunaira.salam@aalto.fi`

Tutor: `Miika Komu`

Abstract

Low Latency Low Loss Scalable Throughput (L4S) is a new technology in the internet service. It introduces new ways for flow-rate control mechanisms over the internet. L4S makes few architectural changes in the already deployed protocols over the internet. These changes modify the components in a network such that non-L4S traffic as well as L4S traffic can be sent over the same channels. In L4S, host has a scalable congestion control mechanism, the intermediary nodes have DualQ Coupled AQM and ECN protocol. The L4S traffic will face low latency and minimum loss incase congestion occurs, and throughput is decreased. Low Latency and Low Loss is a major requirement in time-critical and data intensive applications, where data is generated and sent over internet in real-time.

KEYWORDS: L4S, Active Queue Management (AQM), ECN

1 Introduction

Over the past years, advancement in network technology has increased the use of computer systems in many new types of services. This has led to an

increase of usage in real-time application in various fields, such as, industrial sector, medical science, telecommunication, and entertainment. These applications require the delivery of contents in a fraction of microsecond. In interactive applications like online gaming, Augmented Reality (AR) and Virtual Reality (VR), efficiency in terms of latency and throughput is critical. The Low Latency, Low Loss and Scalable Throughput (L4S) architecture is currently being under discussion in the IETF [1]. Latency is the time taken by a data packet to travel from source to destination and loss is the failure of data packet to arrive at destination. Throughput is defined as the number of data packets sent successfully per second.
The aim of this paper is to explore L4S protocol. L4S protocol can play an important role in interactive applications like online gaming, AR and VR applications where latency and throughput can be critical factors. While it is possible to achieve high throughput to deliver a better user experience in these applications due to the advancements in ultra-reliable low-latency communications in advanced wireless networks. However, the challenge in these applications lies in providing a high throughput and low latency at the same time [2]. This is challenging to achieve because when throughput is increased, the amount of traffic is increased which can further congest the network and this, in turn, increases the latency. The main objective of L4S, as evident from its name, is to balance this contradiction between low latency and high throughput.

2 L4S Architecture

In a high-volume traffic network, latency is increased due to packet loss and congestion. Congestion control algorithms are implemented at the endpoint (hosts) to manage this problem. The L4S host uses scalable congestion controller to achieve low latency. Scalable Congestion Control is defined by the IETF when the average time between two congestion signals (the recovery time) remains unaffected with scaling in flow rate, provided all other factors remain same[3]. The network intermediary nodes such as routers and switches use Active Queue Management (AQM) policy to drop packets inside a buffer associated with a network interface controller (NIC) before that buffer becomes full [4]. The goal of this buffer management is to reduce network congestion or improve end-to-end latency. However, this brings another problem, that is, queuing delay. The queuing delay occurs when the rate of input packets in a network node (end node or intermedi-

ary) exceeds the output capacity. L4S technology works by minimizing or eliminating the queuing delay, hence lowering latency without lowering throughput [2]. L4S achieves this task by using the Explicit Congestion Notification (ECN) bits in the IP header. The ECN bits inform the sender about congestion as soon as the queues start building up. This is known as ECN marking [1]. The L4S architecture comprises of three major parts:

a. Network elements: Isolate L4S traffic from regular traffic in the network and send suitable congestion signals to both sender and receiver. In case of congestion give a signal by marking the ECN bits [1].

b. Protocol features: Enable nodes to separate L4S traffic and allow communication between receiver and sender to notify about congestion [1].

c. Host support: The sending node(host) has a congestion control mechanism. Congestion control manages the entry of data-packets into network [3].

2.1 Network

The L4S architecture delivers low latency by using the following arrangements in the network.

Dual Queue Coupled AQM
The DualQ Coupled AQM [5] comprises of coupling of L4S AQM and Classic AQM and provides latency isolation and bandwidth pooling. Some important properties are described below:
Latency isolation means that two distinct queues are employed to separate L4S queuing delay from the larger queue that Classic traffic requires for full utilization of bandwidth.
Bandwidth pooling means that both L4S and Classic queues behave in a way that it is seen as a single pool of bandwidth. All traffic get almost equal throughput irrespective of the traffic type (Classic or L4S), without any need for the scheduler to classify the flows. This becomes possible by maintaining an AQM in every queue, however, the Classic queue gives congestion signal to ensure a consistent response about congestion control from both queues.

2.2 Protocol

The use of Explicit congestion signals is an important aspect of scalable congestion control. The Explicit Congestion Notification (ECN) protocol [6] is simply a congestion signaling or notification mechanism. It has 2-bit IP-ECN field which maps to four possible signals: ECT(0), Not ECT, ECT(1) and CE. Here, ECT reefers to ECN-Capable Transport and CE refers to Congestion Experienced. The original ECN protocol treats ECN signal equivalent to packet drops. In L4S ECN field is ECN notifies the sender about congestion so that suitable measures can be taken.

2.3 Host

The host in L4S architecture has a scalable congestion control. The host in a L4S supported network must be capable of instantaneous signaling and handling of congestion through scalable congestion control [1]. The scalable congestion control is already implemented in TCP and other transport layer protocols like QUIC, SCTP and RTP/RTCP. For instance, in TCP scalable congestion control is achieved by TCP Prague. Transport layer protocols need to implement scalable congestion control before they can use L4S service. Transport protocols can indicate the congestion control response by using the ECT(1) bits.

3 4. L4S use cases

The L4S transport layer protocol can resolve latency and packet loss which are currently affecting many intense user interactive applications. Using the L4S protocol in the following domains can significantly impact the user experience. These are: cloud-based gaming, video streaming, VoIP, video conferencing, cloud-based VR and AR [1] and 5G network [2]. In cloud-gaming timing is critical as delay and packet loss can adversely affect the user-experience. Latency and packet loss leads to freezing of games and inability to keep up with other players and events taking place in real time. Video Streaming and VoIP services cannot achieve their main goal due to increased latency and packet loss. A live video or voice with disruptions in between is hard to understand and packet loss in this case means absence of chunks of video. Virtual and Augmented reality also sends large streams of data which is time sensitive. Hence, they can not

bear any delay or data loss.

4 Security Analysis

As time passes by new evolution in network technology brings about changes in wired (e.g., ethernet or fiber) or wire- less technologies (e.g., LTE, 5G). One of the major goals of these developments is to provision high throughput and low latency in the network infrastructure, and this leads to emergence of new technologies in the network infrastructure. The Low Latency, Low Loss and Scalable Throughput architecture is a new addition to the network infrastructure. The performance of new technologies like L4S is decent under normal circumstances, however, L4S based network is prone to many security vulnerabilities that can occur due to malign usage of L4S. Malicious attackers can exploit the shortcomings in newer technologies such as L4S to adversely affect the Quality of Experience (QoE) in latency sensitive applications [7].
Letourneau et al [7] use the term 'undesirable flows' to describe misbehaving, malformed and unresponsive flows, either due to legitimate traffic or attempt by an adversary. The IETF has acknowledged the undesirable flows as a problem for L4S [1]. Although IETF has identified these security flaws and stated some countermeasures against them, to date, not enough practical implementations of the countermeasures and active studies are going on to identify the issues and their possible fixes [8].

4.1 Misbehaving Flows

Misbehaving flows comprise of both legitimate and attack flows. Misbehaving flows can be divided into two types: low-rate DoS and Protocol manipulation.

Low-Rate DoS Attacks
It is more difficult to detect Low-Rate DoS (LDoS) attacks compared to DoS or DDoS attacks. The attacker sends regular bursts of packets that are synchronized with the victims Re-transmission Timeout to overflow the router's queue and eventually increase latency. An in-depth study of LDoS attack [9] shows that the proportion of attack traffic needs to be only about 10%-20% of the legitimate traffic flow. Due to its small and concentrated attack footprint, it is concealed in the normal traffic and hard to pin-point. Multiple LDoS attack pulses from different sources gather to

form a concentrated attack. LDoS creates congestion at the target network, creating a bottleneck which results in blocking or affecting the quality of service of end user.

Protocol Manipulation

A deeper study into protocol manipulation atacks by Kothari et al [8] defines protocol manipulation as the ability of the network participants to disrupt the protocol without intermediary nodes realizing it. The study focuses mostly on TCP centered attacks that are carried out by manipulating acknowledgements in the protocol stack for example hacked TCP ACK and hacked ECN.

Misbehaving Flow in L4S

M. Letourneau [10] discusses the possible attacks against the L4S protocol. In this study misbehaving flow was implemented by targeting ECN signaling and making the protocol unresponsive to congestion notification. The protocol was manipulated by eliminating the congestion window reduction when the coefficient of reduction is updated in TCP Prague. As a result, the Low Latency (LL) queue saturates, due to increase in the number of packets that are ECN-marked. Host resends the marked packets but due to false marking of regular traffic flow which was not sent by L4S host the LL queue becomes full. This eventually leads to some of the packets being dropped. This creates a much larger delay which is sufficient to make latency sensitive applications unusable. A malicious user can use this attack to steal the user bandwidth.

4.2 Malformed Flows

Most of the time Malformed flows [10] are legitimate traffic but unwanted from the perspective of L4S. Malformed flows occur when the sending buffers in network stack of an operating system within endpoints are waiting to be filled before transmitting data over the network. L4S architecture is found to be sensitive to the bursty nature of the kernel of the operating system and to the L4S's own architectural burstiness.

4.3 Unresponsive Flows

An Unresponsive Flow [7] does not answer to the congestion control signals. These congestion signals may be generated by ECN marking, packets dropping or delay. This unresponsiveness can occur due to L4S or Classic queue which does not implement congestion control and can lead to overloading of the L4S queues or saturation of congestion signals. In such a case the malicious user sends traffic bursts with altered ECN flags via the classic and low latency queue. The attack traffic is then directed towards the classic queue. As a consequence of saturation in classic queue, packets that belong to the classic queue are directed to Low latency queue which results in saturation of LL queue. This attack targets the coupling mechanism in the DualQ coupled AQM that exists in a network to accommodate both classic and LS4 traffic. As a result, the marking probability of ECN bits increases and causes heavy fluctuations in the traffic flow over the network.

5 Conclusion and Future Work

he L4S protocol is a promising technology that can deliver latency sensitive content under a few milliseconds. L4S can be used in the current network architecture with little changes and it equips the network to cater both types of traffic, L4S and non-L4S. Hence it can be easily implemented but to deploy this protocol in real world operational networks requires L4S to be robust to attacks and handle undesirable flows. Due to these attacks, latency is increased, which is the key aspect of L4S. Further studies are required to be carried out to simulate these undesirable flows in an L4S architecture The combined affect of two or more types of undesirable flows also needs to be studies. Mechanisms using pattern analyses of traffic or machine learning need to be designed to detect possible attacks. Moreover, countermeasures are needed to mitigate the identified security flaws and this will allow secure and stable operation of L4S in future internet.

References

[1] M. B. B. G. W. B.Briscoe Ed, K. De Schepper, "Low latency, low loss, scalable throughput (l4s) internet service: Architecture," 2022.

[2] D. Brunello, I. Johansson S, M. Ozger, and C. Cavdar, "Low latency low loss scalable throughput in 5g networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pp. 1–7, 2021.

[3] B. Briscoe and K. D. Schepper, "Resolving tensions between congestion control scaling requirements," *CoRR*, vol. abs/1904.07605, 2019.

[4] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on tcp enhancements using p4-programmable devices," *Computer Networks*, p. 109030, 2022.

[5] B. Briscoe, K. De Schepper, O. Tilmans, M. Kühlewind, J. Misund, O. Albisser, and A. S. Ahmed, "Implementing the'prague requirements' for low latency low loss scalable throughput (l4s)," *Netdev 0x13*, 2019.

[6] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP." RFC 3168, Sept. 2001.

[7] M. Letourneau, G. Doyen, R. Cogranne, and B. Mathieu, "A comprehensive characterization of threats targeting low-latency services: the case of l4s," 2022.

[8] N. Kothari, R. Mahajan, T. Millstein, R. Govindan, and M. Musuvathi, "Finding protocol manipulation attacks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, p. 26–37, aug 2011.

[9] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate dos attacks, detection, defense, and challenges: A survey," *IEEE Access*, vol. 8, pp. 43920–43943, 2020.

[10] M. Letourneau, K. B. N'Djore, G. Doyen, B. Mathieu, R. Cogranne, and H. N. Nguyen, "Assessing the threats targeting low latency traffic: the case of l4s," in *2021 17th International Conference on Network and Service Management (CNSM)*, pp. 544–550, 2021.