Proceedings of the Seminar in Computer Science (CS-E4000), Spring 2024, ver. 2

Antti Ylä-Jääski and Sara Ranjbaran

Tutors for seminar topics

Antti Ylä-Jääski, Arsi Ikäheimonen, Blerta Lindqvist, Jaakko Harjuhahto, Jose Luis Martin Navarro, Juho Vepsäläinen, Lachlan Gunn, Matti Huotari, Matti Siekkinen, Mikko Kiviharju, Nam Hee Kim, Parinya Chalermsook, Russell W.F. Lai, Samuel Marchal, Sanna Suoranta, Sara Ranjbaran, Shuzhe Wang, Simo Aaltonen, Tuomas Aura, Vesa Hirvisalo, Yogesh Verma, and Yunhao Yuan

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

Doubly Efficient Private Information Retrieval Protocol: Construction & Optimization

Akram Aziz
akram.aziz@aalto.fi

Tutor: Russell W. F. Lai
russell.lai@aalto.fi

Abstract

Doubly Efficient Private Information Retrieval (DEPIR) protocol, proposed by [LMW22], satisfies the need for secure and private data retrieval. This paper provides a comprehensive overview of the DEPIR protocol, its construction, and optimization efforts. In order to successfully construct the protocol, we need to express the database as an m-variant polynomial, construct a single-server Private Information Retrieval (PIR) scheme, and preprocess the polynomial to enable efficient evaluation over ciphertexts. Optimization efforts proposed by the work of [OPPW23] focus on reducing preprocessing time and query runtime, with enhancements to the Algebraic Somewhat Homomorphic Encryption (ASHE) scheme identified as a primary focus for improvement.

Contents

1 Introduction

Currently, most of the knowledge exists on the internet. Over 65% of the earth's population has constant access to the internet [n.d24]. 5.3 billion people are one click away from learning something new, just like what I did to retrieve that information. Due to the increased accessibility of the internet, the importance of security and privacy has become more significant. This reason ignited security experts to find a secure way to retrieve information from the public internet. According to [LMW22], private information retrieval (PIR) allows internet users to read data from a public database held on a remote server, without revealing to the server the location the client accessed.

Lin, Mook, and Wichs, in [LMW22], successfully constructed a general doubly efficient private information retrieval DEPIR protocol and managed to prove its security and correctness. Moreover, authors of [OPPW23] tried to implement the proposed theoretical protocol trying to answer how far DEPIR is from being practical.

The goal of this paper is to provide an overview of the construction of doubly efficient private information retrieval (DEPIR) and some of the optimization efforts.

2 Preliminaries

This section provides some useful mathematical and background knowledge that would be needed to fully understand some of the following sections.

Define $\mathbb{N}$ to be the set of natural number, $\mathbb{Z}$ to be the set of integers and $\mathbb{R}$ to be the set of real numbers. For any integer $n \geq 1$, define $[n] = \{1, ..., n\}$. Unless mentioned otherwise, d and q are two large prime numbers that will be used to define groups. For any integer $q \in \mathbb{N}$, then $\mathbb{Z}_q$ is a the quotient ring $\mathbb{Z}/q\mathbb{Z}$.

A low-degree multivariate polynomial is a polynomial of more than one variable, where the highest power of any of variables is of a relatively low degree.

3 PIR and FHE

Fully Homomorphic Encryption (FHE) is an encryption scheme that allows to evaluate a new input over an encrypted system without the need of decryption. The main limitation of Fully Homomorphic Encryption (FHE) lies in its runtime complexity, which scales linearly with the system size. Additionally, encryption time is directly related to the size of the input, and decryption time is directly related to the size of the output.

If we consider the entire internet to be our database (C), the client query to be the input (x), and the domain is of polynomial-size, then Private Information Retrieval (PIR) can be viewed as a FHE problem. However, given the runtime complexity mentioned above, the runtime of each query would grow proportional to O(|C|), which is proportional to the size of the entire internet [LMW22].

The authors of [LMW22] decided to approach this problem by proposing their construction of DEPIR Protocol.

4 Basic Blocks & Related Works

In order to successfully construct their DEPIR protocol, [LMW22] introduces multiple basic blocks that will be carefully put together in order to achieve their goal.

4.1 Ring Learning With Errors (RingLWE)

The first of these blocks is RingLWE. Ring learning with errors is a well-known NP-hard problem that is constructed over the hardness of finding short vectors in ideal lattices in the worst case [LPR10]. Moreover, RingLWE is considered the basic building block of the new NIST standard for the public-key encryption scheme [NIS22]

4.2 Algebraic Somewhat Homomorphic Encryption (ASHE)

In the work of [BV11], Brakerski and Vaikuntanathan introduce their Somewhat Homomorphic Scheme SHE, which is based on RingLWE assumption. The authors of [LMW22] use the above mentioned SHE construction to build their own ASHE scheme.

ASHE is a symmetric key CPA-secure encryption scheme. This scheme enables us to evaluate a low-degree multivariant polynomial f over an

encrypted ciphertext, $ct \in R$, instead of evaluating f over the plaintext input. This result becomes feasible by $lifting$ the plaintext space $\mathbb{Z}_d$ to be a subset of the ciphertext space R. The term $lifting$ maps to an algorithm of interpreting each element of a certain domain to an equivalent element of another domain.

To successfully interpret elements from the plaintext space ($\mathbb{Z}_d$) to the ciphertext space (R), we need to complete two steps. The first step involves interpreting from $\mathbb{Z}_d$ to $\mathbb{Z}_q$. This step is achieved by reducing each input modulo q. This intermediate step is necessary because the ciphertext space R is defined in terms of $\mathbb{Z}_q$. The second step is relatively straightforward, as it entails finding a constant representation of each element of the group $\mathbb{Z}_q$ to an element in the ring R.

The reason behind $lifting$ the elements to the ring R, is to prepare the elements of the ASHE scheme to be fit for the preprocessing function of [KU11] in order to yield the expected output of preprocessing polynomials.

4.3 ASHE from RingLWE

After successfully constructing ASHE from the RingLWE assumption, [LMW22] introduce a full scheme, where we get a single server PIR. the client starts preparing for the query by encrypting every digit of the input $I(i_1, ..., i_m) \in \mathbb{Z}_d$ and sends the resulted ciphertext $ct(ct_1, ..., ct_m) \in R$ to the server. The server homomorphically evaluates the low-degree multivariant polynomial f over the ciphertext, and sends the response $\alpha(\alpha_1, ..., \alpha_m)$ back to the client. The client will be able to decrypt the server's response α to get the answer of the query.

What separates the ASHE scheme from the DEPIR protocol is the preproccessing of the polynomial f, which will yield in extreme efficient evaluation of the client's future queries.

4.4 Preprocessing Polynomials

The results of Kedlaya and Umans [KU11] lays the foundation of processing polynomials $f(X_1, ..., X_m)$ over a ring R in order to get a data structure. This data structure will enable us to efficiently evaluate the polynomial over any input. [KU11] proves that for any $\epsilon > 0$, the polynomial processing has a runtime complexity $O(N^{1+\epsilon})poly(\lambda)$ and query evaluation time $poly(\lambda, logN)$. However, one of [KU11] limitations, which [LMW22]

took in consideration, is that their results are strictly conditioned by the chosen parameters.

4.5 Keyed DEPIR

In [LMW22], the constructed DEPIR is inspired by the results of [BIPW17] and [CHR17]. These two papers managed to set the foundation for the unkeyed DEPIR, by proposing keyed DEPIR protocols. In the work of [CHR17], the authors proposed secret-key DEPIR protocol. Moreover, the work of [BIPW17] uses the secret-key DEPIR protocol in order to propose a public-key DEPIR. Despite that keyed and unkeyed DEPIR might seem to be closely related, the hardness assumption that each of these protocols rely on varies. Another problem that arouses with keyed DEPIR protocols is the need for a third trusted party to create and distribute the keys. The authors of [LMW22] make a claim that the Prep algorithm, which is responsible for preprosessing the server's database, is solely for correctness and efficiency purposes, and not for proving the security. They argue regardless the server being honesty or not, the security of the client's query remains.

5 DEPIR

[LMW22] specifies two phases that their Doubly Efficient Private Information Retrieval (DEPIR) have, preprocessing and querying. The full construction of DEPIR protocol is divided into three main steps: expressing the database into an m-variant polynomial, constructing a single server PIR, preprocessing the m-variant polynomial. However, before we dive into the details of the three essential steps, we start by defining a set of parameters needed to ensure the successful construction of the proposed DEPIR. Let N be the size of the considered database, $d \in \mathbb{N}$ be a prime number, and the size of the input m. Such that, $d^m > N$.

Expressing the database as a polynomial: [LMW22] aims to express the database DB into an m-variant polynomial over the field F_d, and translate the input $i = (i_1, ..., i_m)$ into a base-d representation of the input $i = \text{base}_{d,m}(i)$. Such that, $f_{DB}(i = \text{base}_{d,m}) = DB[i]$

Constructing PIR from ASHE: The second main step in the construction of DEPIR, is constructing a single-server PIR from the proposed ASHE scheme. Through PIR, the client will start by translating

its plainspace input $i \in F_d$, of the size m, to get a base-d representation of the $i = \text{base}_{d,m}(i) \in \mathbb{Z}_d$, then the client will encrypt the $\text{base}_{d,m}(i)$ to get a ciphertext $ct = (ct_1, ..., ct_m) \in R$ using a generated secret key s. On the server side, the server will have to $lift$ the encoded polynomial f_{DB} into $\widetilde{f}_{DB} \in R$. The server will evaluate $\widetilde{f}_{DB}$ over the ciphertext ct to get the answer $ct^* = (ct_1^*, ..., ct_m^*)$ for the client's query. Upon receiving ct^*, the client will be able to decrypt it using the same secret key s to recover $DB[i] = f_{DB}(i_1, ..., i_m)$.

Preprocessing: The last step remaining is to take the constructed PIR protocol, in the previous step, and construct DEPIR from it. [LMW22] simply start by preprocessing the polynomial f_{DB}. Lin, Mook, and Wichs confess that the difference between PIR and DEPIR is relatively small; from the client point of view, DEPIR is the same as PIR. Therefore, the security of DEPIR inherently comes from the security of PIR, which comes from the security of the ASHE scheme under the RingLWE assumption. the motive behind the preprocessing of polynomials is to enable the server to evaluate the polynomial f_{DB} over the ciphertext $ct = (ct_1, ..., ct_m)$. The preprocessing algorithm used here is the same algorithm proposed by the work of [KU11], which also has been proven its efficiency.

6 Optimizing DEPIR

Okada, Player, Pohmann, and Weinert in their work [OPPW23] clarify that the above mentioned DEPIR protocol is purely theoretical. Therefore, they aimed to implement an approximation of the protocol of [LMW22]. Moreover, the authors of [OPPW23] propose some optimization regarding the preprocessing time and the runtime of each query.

The authors of [OPPW23] propose an interesting approach regarding the future works in order to optimize the implementation of DEPIR. They suggest the way to improve the protocol is through improving ASHE scheme. If it became possible to construct an ASHE scheme with a better noise growth and ring size R, then the improvement of a practical DEPIR would follow.

6.1 Limitations & Tradeoffs

One of the problems of [LMW22] DEPIR is the $lift$ algorithms. Therefore, In [OPPW23], the authors aim to find precise formulas to asymptotically

understand the number of primes, storage size, and runtime of these algorithms. Moreover, the authors of [OPPW23] mention that the larger the ring size R the $lifting$ algorithm maps into, the worse the efficiency yields. In return, they proposed that by changing the ciphertext space, we will be able to implement a known and faster arithmetic ciphertexts. Additionally, [OPPW23] proposes that the isomorphism can be computed using a fast Fourier Transform (FFT).

7 Conclusion

In conclusion, this paper manages to present an overview of the work of [LMW22] and their construction of the DEPIR protocol. It also presents all of the basic schemes needed to successfully construct the DEPIR protocol. Such as, RingLWE, ASHE scheme, and PIR scheme. Moreover, this paper also integrates the optimization of the DEPIR protocol. While the answer to the practicality of DEPIR investigation of yields to an unsatisfactory no, the authors of [OPPW23] suggest possible routes for future exploration and practical implementation of DEPIR

References

[BIPW17] E. Boyle, Y. Ishai, R. Pass, and M. Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin. Technical report, Theory of Cryptography Conference, Part II, volume 10678 of Lecture Notes in Computer, 2017.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. *Fully homomorphic encryption from ring-LWE and security for key dependent messages*. In Phillip Rogaway, editor, Advances in Cryptology – CRYPTO, 2011.

[CHR17] R. Canetti, J. Holmgren, and S. Richelson. Towards doubly efficient private information retrieval. Technical report, Theory of Cryptography Conference, Part II, volume 10678 of Lecture Notes in Computer Science, 2017.

[KU11] K. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. Technical report, SIAM Journal on Computing, 2011.

[LMW22] W-K. Lin, E. Mook, and D. Wichs. Doubly Efficient Private Information Retrieval and Fully Homomorphic RAM Computation from Ring LWE. REP 1703, Cryptology ePrint Archive, December 2022. https://eprint.iacr.org/2022/1703.pdf.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. Technical report, Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT, 2010.

[n.d24] Internet User Statistics n.d. How many use the internet in 2024. *demandSage*, 2024.

[NIS22] Post-quantum cryptography: Selected algorithms. Technical report, NIS, 2022.

[OPPW23] H. Okada, R. Player, S. Pohmann, and C. Weinert. Towards Practical Doubly-Efficient Private Information Retrieval. REP 1510, Cryptology ePrint Archive, October 2023. https://eprint.iacr.org/2023/1510.pdf.

Doubly-Efficient Private Information Retrieval

Amirhosein Rajabi
amirhosein.rajabi@aalto.fi

Tutor: Russell W. F. Lai

Abstract

DEPIR schemes aim to build efficient PIR schemes in terms of communication and computation complexity. Emphasizing on the key ideas, this paper summarizes the breakthrough result of Lin, Mook, and Wichs and the main tools they used. Lin et al. constructed the first unkeyed DEPIR scheme to achieve polylogarithmic communication complexity using slightly superlinear offline preprocessing, which the server can run independently once for all clients.

KEYWORDS: Private Information Retrieval, PIR, Algebraic Somewhat Homomorphic Encryption, SHE, fast multivariate polynomial online evaluation, Ring Learning with Errors, RingLWE

1 Introduction

Private Information Retrieval (PIR) protocols allow users of a database service to query database entries without disclosing any information about their query to the service. These protocols have numerous applications, such as querying patent databases [1], compromised credential checking [8], and location privacy [6, 10].

The communication and computation complexity of the client queries

to the database has been extensively studied since the notion was introduced by Chor et al. [5]. For each query, communication complexity is the amount of information communicated between the database and the client. Computation complexity addresses the running time of server computation in terms of the database size N. Doubly-Efficient Private Information Retrieval (DEPIR), coined by Beimel et al. [2], seeks to make both the computation and communication complexities (online complexities) efficient or precisely polylogarithmic in N. Nevertheless, Beimel et al. [2] proved the computation complexity of such a scheme has to be linear unless the database is preprocessed. Therefore, a DEPIR scheme needs to preprocess the database and store the resulting data structure on the server for efficient query complexity.

This paper summarizes the breakthrough result of Lin et al. [11] who constructed the first unkeyed DEPIR scheme based on the standard Ring Learning with Errors assumption (Ring-LWE). Strictly speaking, it discusses the main tools they used for their construction. Amongst these tools, it focuses on a data structure for polynomial evaluations with polylogarithmic complexity by Kedlaya and Umans [9] and an algebraic somewhat homomorphic encryption (ASHE) scheme by Lin et al. [11].

The rest of the paper is structured as follows. Section 2 overviews definitions, tools, and main construction. Section 3 introduces how a database can be encoded as a multivariate polynomial interpolation problem. Section 4 first discusses Brakerski and Vaikuntanathan's SHE scheme and then proceeds to the ASHE scheme. Section 5 presents fast multivariate polynomial evaluations with preprocessing. Section 6 first defines DEPIR security and efficiency and then explains the construction. Section 7 concludes the paper by indicating future work.

2 Overview

2.1 PIR and DEPIR

A single-server private information retrieval (PIR) scheme is a protocol that allows users to learn the i-th location $\mathsf{DB}[i]$ of the database $\mathsf{DB} \in \{0,1\}^N$ available to the server without revealing index i to the server [11]. An example could be a privacy-preserving search engine where clients can query their keywords and get results back without letting the server learn

anything about their search queries.

It is worth mentioning that the security requirement does not prevent the user from learning other entries than the ones they queried. Hence, a trivial scheme may send the entire server's database to the client so that it can read the entry they are looking for, but it is simply impractical for large databases and a major issue for communication bandwidth. As a result, only those schemes with $o(N)$ (sublinear in N) or just $\mathcal{O}(\log^c N)$ (polylogarithmic in N) communication complexity are desired.

A major limitation of PIR schemes, as mentioned in Section 1 and shown in [2], is that each query processing needs to involve all entries of the database, lest the server learns some entries are untouched, which breaks the security requirement [11]. DEPIR schemes avoid this limitation by preprocessing the database DB into a data structure $\widetilde{\mathsf{DB}}$ and responding to user query with $\widetilde{\mathsf{DB}}$. Informally speaking, $\widetilde{\mathsf{DB}}$ mix all entries in DB once for all queries to overcome the limitation. Furthermore, DEPIR schemes make polylogarithmic communication and computation complexity possible.

There are three types of DEPIR schemes: *unkeyed*, *public-key*, and *secret-key*. Unkeyed schemes refer to those where the database is preprocessed independently and deterministically by the server, unlike the secret-key and public-key variants where a trusted party needs to provide proper keys to all parties and the preprocessed database to the server [11].

The scheme constructed by Lin et al. [11] falls into the unkeyed category, which avoids needing a trusted party to provide keys and the server having a key to process the query. Only the client must generate a key for themselves and send encrypted queries to the server. Therefore, the server needs to compute over the encrypted query. To achieve this, Lin et al. [11] utilized Homomorphic Encryption (HE) schemes (specifically Somewhat Homomorphic Encryption (SHE) schemes). These schemes enable computing *allowed* polynomials over the ciphertexts as if the polynomial is computed over the corresponding plaintexts [13] .

They based their scheme on an Algebraic SHE scheme derived from the SHE scheme of Brakerski and Vaikuntanathan [4]. Additionally, they avoided the separate polynomial evaluation function $\mathsf{Eval}(c, f(\cdot))$ found in [4] by endowing ring structure to the ciphertext space so that the polynomial $f(\cdot)$ could be directly applied to the ciphertexts.

DEPIR schemes are essentially PIR schemes that are just more efficient. Lin et al. [11] approach the construction using ideas from PIR

schemes by first encoding the database DB as a polynomial f_{DB} such that $f_{\mathsf{DB}}(i) = \mathsf{DB}[i]$ for all indices i. Then, f is preprocessed in time superlinear in N into a data structure $\widetilde{\mathsf{DB}}$ that enables evaluating f over any point of its domain in time polylogarithmic in N. The preprocessing step is where DEPIR schemes are different from PIR schemes and more efficient. When a client is interested in value $\mathsf{DB}[i]$, it encrypts i with the ASHE scheme, obtaining the cipher text c. This ciphertext is sent to the server for homomorphic evaluation of f_{DB} over c, using the data structure $\widetilde{\mathsf{DB}}$. After server evaluation, based on the properties of the ASHE scheme, an encryption of $f_{\mathsf{DB}}(i)$ is sent back to the client that can only be decrypted with the secret key held by them. The following sections go through these steps in detail.

2.2 Preliminaries

In the following sections, concepts and notations, such as rings, quotient rings, polynomial rings, and (prime and cyclotomic) fields are used; therefore, the reader is referred to standard textbooks on abstract algebra [7, 3], algebraic number theory [14], and cyclotomic fields [16] for definitions and discussions. Finite fields of prime order p are denoted by $\mathbb{F}_p$. Polynomials f and f_{DB} (with or without subscript DB) are used interchangeably based on context. The individual degree for each variable of a multivariate polynomial is the maximum degree a variable attains in any of the monomials. On the other hand, the total degree of multivariate polynomial is the maximum degree of monomials. The degree of a monomial is the summation of degrees of its variables. For example, the total degree of $f(x, y) = x^2y^3 + xy^5$ is 6, while the individual degree of y and x are 5 and 2, respectively. Let $\mathbb{Z}_n^*$ be the multiplicative group of integers modulo n. Define the norm of a polynomial p as the maximum absolute value of its coefficients and denote it by $||p||_\infty$. The security parameter is referred to by λ in the text. In all of the sections, $\log_2 n$ is denoted by $\log n$ for simplicity. Finally, denote finite set $\{0, 1, \ldots, n-1\}$ by $[n]$.

3 Encoding database as a polynomial

As mentioned in Section 2, Lin et al. [11] modeled the database $\mathsf{DB} \in \{0,1\}^N$ as a polynomial f_{DB} such that $f_{\mathsf{DB}}(i) = \mathsf{DB}[i]$. Towards this end, an index $i \in \{1, 2, \ldots, N\}$ is represented as an m-digit number in base d,

i.e., $(i_1, i_2, \ldots, i_m)$ where $i = \Sigma_{j=1}^{m} i_j \cdot d^{j-1}$ and $i_j \in [d]$. For proper multiplication properties (such as existence of multiplicative inverse), Lin et al. require d to be prime so that $i_j \in \mathbb{F}_d$. Then, polynomial f_{DB} is considered a multivariate polynomial over $\mathbb{F}_d$, and the problem of encoding the database is reduced to interpolating a multivariate polynomial f_{DB} from N points of the form $(i, \mathsf{DB}[i])$. Hence, it is required that $N \leq d^m$. A benefit of modeling indices as m-digit numbers is to decrease the individual degree to less than d in each variable of the polynomial f and increase the degree of freedom by choosing an appropriate prime base d.

Lin et al. [11] introduce a recursive algorithm to interpolate polynomial $f(X_1, \ldots, X_m)$ with indivisual degree in all variables less than d over $\mathbb{F}_d$ given d^m points $\{y_x\}_{x \in \mathbb{F}_d^m}$ in time $\mathcal{O}(d^m \cdot m \cdot \mathsf{poly}\log(d))$ such that $f(x) = y_x$. (If $N < d^m$, let $y_x = 0$ for all points x that do not represent a database index.) The idea is to eliminate each variable in each iteration. The first iteration is as follows. Univariate polynomials $g_{(x_1, x_2, \ldots x_{m-1})}(X_m) = y_{(x_1, x_2, \ldots, x_{m-1}, X_m)}$ for all $(x_1, \ldots, x_{m-1}) \in \mathbb{F}_d^{m-1}$ are determined. The degree of these polynomials will be less than d. This is the well-known problem of univariate polynomial interpolation. Then, the problem reduces to find f such that $f(X_1, X_2, \ldots, X_m) = g_{(X_1, X_2, \ldots, X_{m-1})}(X_m)$. In other words, if $g_{(x_1, x_2, \ldots x_{m-1})}(X_m) = \Sigma_{i=0}^{d-1} c_{(x_1, x_2, \ldots, x_{m-1}, i)} X_m^i$, then $f(X_1, X_2, \ldots, X_m)$ should be defined as $\Sigma_{i=0}^{d-1} f_i(X_1, X_2, \ldots, X_{m-1}) X_m^i$ such that $f_i(X_1, X_2, \ldots, X_{m-1}) = c_{(X_1, X_2, \ldots, X_{m-1}, i)}$, i.e., eliminating variable X_m. This reduces the problem to find polynomials f_i such that $f_i(X) = c_{X,i}$ for all $X \in \mathbb{F}_d^{m-1}$ and $i \in [d]$, which can be solved recursively. Refer to Appendix B in [11] for more details.

4 Somewhat Homomorphic Encryption (SHE)

A symmetric encryption scheme ($\mathsf{Gen}(1^\lambda)$, $\mathsf{Enc}(m, k)$, $\mathsf{Dec}(c, k)$) is somewhat homomorphic encryption if the plaintext space is a ring and there is an evaluation function such that given an *allowed* polynomial $f(x_1, \ldots, x_n)$ and ciphertexts $c_i = \mathsf{Enc}(m_i, k)$, $\mathsf{Eval}(f, c_1, \ldots, c_n)$ outputs encryption of $f(m_1, \ldots, m_n)$, i.e., $\mathsf{Dec}(\mathsf{Eval}(f, c_1, \ldots, c_n), k) = f(m_1, \ldots, m_n)$ [13].

Ciphertexts in HE schemes are *noisy* as these schemes rely on the Ring-LWE assumption in which the adversary can not distinguish between two distributions that one of them involves random bounded noises. This noise grows with the homomorphic operations (additions and multiplications) and can cause decryption to fail if they are too large. Hence, *al-*

lowed polynomials prevent this failure by restricting the set of supported polynomials and the maximum number of multiplications.

In the following two Subsections, elements of $\mathbb{Z}_q$ are represented with that of $(-q/2, \ldots, q/2] \cap \mathbb{Z}$.

4.1 Brakerski and Vaikuntanathan's SHE scheme (BV scheme)

In their work [4], Brakerski and Vaikuntanathan introduce an SHE scheme with security parameter λ, prime number $q = q(\lambda)$ and degree $n = 2^{g(\lambda)}$ cyclotomic polynomial $h(x) = x^n + 1$. The plaintext space is $R_t = \mathbb{Z}_t[x]/\langle h(x) \rangle$ where $t \in \mathbb{Z}_q^*$ is prime. Plaintext space can be seen as degree $n-1$ polynomials with coefficients in $\mathbb{Z}_t$. Since degree $n-1$ polynomials over any ring can be seen as n-tuples over that ring and vice versa, their formulation is just an algebraic way to allow encryption of n plaintexts. A public error distribution χ over $R_q = \mathbb{Z}_q[x]/\langle h(x) \rangle$ is defined in the scheme for secret key and error sampling. Finally, the ciphertext space in this scheme is $R_q^{\leq D+1}$ (tuples over R_q with at most $D+1$ elements) where D is the maximum number of multiplications of fresh ciphertexts (a function of q, h, χ). Note that ciphertexts are $D+1$-tuples and can be viewed as degree D polynomials which is the case for ASHE.

The key generation algorithm samples secret key s from χ, denoted by $s \leftarrow\!\!\$\, \chi$. The encryption algorithm is given message $m \in R_t$ and $s \in R_q$. It outputs ciphertext $c = (c_0 = as + te + m, c_1 = -a) \in R_q^2$ by sampling $a \leftarrow\!\!\$\, R_q$ and error $e \leftarrow\!\!\$\, \chi$. Note that adding $m \in R_t$ with $as + te \in R_q$ is allowed since $t \in \mathbb{Z}_q^*$ and there is a natural embedding from R_t into R_q. The evaluation algorithm suffices to define how addition and multiplication should be handled. Adding two ciphertexts with the same length D is straightforward via a coordinate-wise addition. Namely, $c + c' = (c_0 + c'_0, c_1 + c'_1, \ldots, c_D + c'_D)$. (Shorter ciphertexts are padded with zeros.)

To motivate the definition of multiplication, observe that $te + m = c_0 + c_1 s$ and $te' + m' = c'_0 + c'_1 s$. Hence, $(te + m)(te' + m') = (c_0 + c_1 s)(c'_0 + c'_1 s)$. Expanding $(te + m)(te' + m')$ gives $t(em' + e'm + tee') + mm' \equiv mm' \mod t$. If the norm of the $(c_0 + c_1 s)(c'_0 + c'_1 s)$ as a polynomial in R_q is less than $q/2$, the error does not render the recovery impossible. (For further discussion, refer to Section 2 and Section 3 in [4]. Theorem 1 of their paper summarizes the error upper bound for message recovery.) However, expanding $(c_0 + c_1 s)(c'_0 + c'_1 s)$ yields $c_0 c'_0 + (c_0 c'_1 + c_1 c'_0)s + c_1 c'_1 s^2$. This entails including $c_0 c'_0$, $c_1 c'_0 + c_0 c'_1$, and $c_1 c'_1$ in the ciphertexts are enough to recover mm' using the secret key s. More importantly, anyone

can compute an encryption of mm' given c and c' without knowing the secret key s or finding new information of m or m', which is expected from the public evaluation function. Therefore, $c \times c'$ is defined as $(c_0c'_0, c_1c'_0 + c_0c'_1, c_1c'_1)$, which can be further generalized to longer ciphertexts using polynomial multiplication, i.e., multiplication of $c = (c_0, \ldots, c_l)$ and $c' = (c'_0, \ldots, c'_{l'})$ is defined as the coefficients vector of $(\Sigma_{i=0}^{l} c_i v^i) \cdot (\Sigma_{i=0}^{l'} c'_i v^i)$ for some indeterminate v.

Decryption of fresh ciphertexts (i.e., (c_0, c_1)) is as simple as $c_0 + c_1 s \mod t$, while decryption of (c_0, c_1, c_2) is computed via $c_0 + c_1 s + c_2 s^2 \mod t$. This can be immediately generalized to $\mathsf{Dec}\big(c = (c_0, \ldots, c_D), s\big) = \Sigma_{i=0}^{D} c_i s^i \mod t$, provided that norm of polynomial $\Sigma_{i=0}^{D} c_i s^i$ as a member of R_q is less than $q/2$. Remarkably, this upper bound is crucial for correct message recovery as computation in R_q and taking modulo t may not be compatible with large errors. Multiplication hints at why the scheme lets the ciphertext space be $R_q^{\leq D+1}$. Fresh ciphertexts generated by Enc have a length of 2. Addition leaves the length of ciphertexts unchanged, whereas multiplication increases the length of ciphertexts.

Finally, Brakerski and Vaikuntanathan prove their scheme is secure under the worst-case hardness of approximating shortest vectors on ideal lattices within a factor of $\mathcal{O}(2^{n^\epsilon})$ (Theorem 2, [4]). This assumption implies the RingLWE assumption when the quotient polynomial is cyclotomic [12].

4.2 Algebraic Somewhat Homomorphic Encryption (ASHE)

As hinted in the overview, Lin et al. [11] avoided the need for an evaluation function by endowing ring structure to the ciphertext space of BV's SHE scheme so that polynomial f_{DB} can be directly applied to the ciphertexts, i.e., $\mathsf{Dec}(f(c_1, \ldots, c_n), k) = f(m_1, \ldots, m_n)$ where $\mathsf{Enc}(m_i, k) = c_i$. They interpret the ciphertext tuples as polynomials. Namely, any tuple in $R_q^{\leq D+1}$ corresponds to a polynomial over R_q of degree at most D. Since Lin et al. [11] are interested in evaluating polynomials of total degree less than D over the ciphertexts, they set the ciphertext space of ASHE scheme as $C = R_q[y]/\langle y^D + 1\rangle = \mathbb{Z}_q[x, y]/\langle x^n + 1, y^D + 1\rangle$. However, the plaintext space is chosen to be $\mathbb{Z}_d$ instead of $R_t = \mathbb{Z}_t[x]/\langle x^n + 1\rangle$. (Recall from Section 3 that digits of database indices belonged to $\mathbb{F}_d$.)

All public parameters (i.e., q, n, and β) of the scheme are derived deterministically by $\mathsf{ASHE.Setup}(1^\lambda, 1^d, 1^D, N)$ from the security parameter λ, plaintext space size d, the number of terms N in the multivariate poly-

nomial f with coefficients in $\mathbb{Z}_d$, which is evaluated over ciphertexts, and the total degree of the polynomial less than D. They require $q \gg d$ to be relatively prime to d but not necessarily prime. (cf. [4] where q was chosen as prime.) They also introduce a notion of *lifting* that refers to interpreting members of $\mathbb{Z}_d$ as members of Z_q, R_q, and C through the natural embedding of these rings into each other. For example, coefficients of f are lifted to C before evaluating over ciphertexts. Analogous to BV's SHE scheme, a public β-*bounded* error distribution χ over R_q is defined where $\Pr[||e||_\infty \leq \beta : e \leftarrow\!\!\$\, \chi] = 1$. The reader is referred to Section 3.1 of the works of Lin et al. [11] for a detailed discussion of the choice of parameters.

The key generation algorithm $\mathsf{ASHE.Gen}(1^\lambda)$ samples $s \leftarrow\!\!\$\, R_q$ uniformly. The encryption algorithm $\mathsf{ASHE.Enc}(s, \mu)$ inputs secret key $s \in R_q$ as well as message $\mu \in \mathbb{Z}_d$, samples $a \leftarrow\!\!\$\, R_q$ and error $e \leftarrow\!\!\$\, \chi$, interprets $\mu \in \mathbb{Z}_d$ naturally as a constant polynomial in R_q (lifts μ to R_q), and outputs $\mathsf{ct}(y) = -a \cdot y + a \cdot s + d \cdot e + \mu \in C$. Notice that $\mathsf{ct}(s) = d \cdot e + \mu \equiv \mu \mod d$ and fresh encryptions are polynomials of degree 1 in y. Multiplication and addition of ciphertexts are defined naturally as polynomial multiplication and addition, respectively. Ciphertexts correspond to the artificial polynomials with indeterminate v, which were introduced when the multiplication of ciphertexts in BV's scheme was defined. Generally, for decryption of $\mathsf{ct}(y)$ under the secret key s, the algorithm $\mathsf{ASHE.Dec}(\mathsf{ct}, s)$ computes $g = \mathsf{ct}(s) \in R_q$ and outputs $g(0) \mod d$. Observe that messages belong to $\mathbb{Z}_d$ and were naturally embedded as constant polynomials in R_q; therefore, $g(0) \in \mathbb{Z}_q$ (the constant term) is computed modulo d. Similar to BV's scheme, decryption is correct as long as $||\mathsf{ct}(s)||_\infty \leq q/2$.

Security definitions of ASHE schemes are defined with indistinguishability under chosen plaintext attack (IND-CPA) and it follows directly from the scaled error variant of the RingLWE assumption. Refer to Section 3 of [11] for more details of the correctness and security proof.

5 Fast multivariate polynomial evaluation with preprocessing

Theorem 5.1 (Theorem 2.1, [11])**.** *Let $C = \mathbb{Z}_q[x, y]/\langle h(x), g(y)\rangle$ for some $q \in \mathbb{N}$ and arbitrary monic polynomials h over x and g over y. Let $f \in C[X_1, \ldots, X_m]$ be a polynomial of individual degree less than d in every variable. There is a* ***preprocessing algorithm*** *that takes the coefficients*

of f, runs in time

$$d^m \cdot \mathsf{poly}(m, d, \log|C|) \cdot \mathcal{O}(m(\log m + \log d + \log\log|C|))^m,$$

and outputs a data structure of at most the same size, and there is an ***evaluation algorithm*** *with random access to the data structure that computes $f(X)$ for all $X \in C^m$ in time* $\mathsf{poly}(d, m, \log|C|)$.

Lin et al. [11] prove the theorem in three steps. They first prove a simpler case by considering polynomials over $\mathbb{Z}_q$, then polynomials over $R_q = \mathbb{Z}_q[x]/\langle h(x)\rangle$, and finally polynomials over $C = R_q[y]/\langle g(y)\rangle$. The following Subsections explain the proof idea behind these steps.

5.1 Polynomials over $\mathbb{Z}_q$

The high-level idea is as follows. First, interpret the coefficients of $f \in \mathbb{Z}_q[X_1, \ldots, X_m]$ in $\mathbb{Z}$ (lifting f to f^{lift} by interpreting its coefficients in $\mathbb{Z}$ instead of $\mathbb{Z}_q$). Then, store evaluations of f^{lift} over all elements of $\mathbb{Z}_{p_1}, \ldots, \mathbb{Z}_{p_h}$ for small distinct prime numbers $p_1, \ldots, p_h$, i.e., for every $i \in \{1, \ldots, h\}$, store $(x, f^{\mathsf{lift}}(x) \mod p_i)$ in table T_i for all $x \in \mathbb{Z}_{p_i}$. Set $\widetilde{\mathsf{DB}} = (p_1, \ldots, p_h, T_1, \ldots, T_h)$. Lin et al. use the FFT-based fast multivariate multipoint evaluation algorithm described in Theorem 4.1 in the work of Kedlaya and Umans [9] to compute $f^{\mathsf{lift}}(x) \mod p_i$ for all $x \in \mathbb{Z}_{p_i}$ in time $\mathcal{O}(m \cdot (d^m + q^m) \cdot \mathsf{poly}(\log q))$.

To evaluate $f(X)$ for some $X \in \mathbb{Z}_q^m$, first interpret X as a member of $\mathbb{Z}^m$ (lift X to X^{lift}). Then, look up the values $z_i = f^{\mathsf{lift}}(X^{\mathsf{lift}} \mod p_i)$ in the table T_i and use the Chinese Remainder Theorem (CRT) to obtain z where $z \equiv z_i \mod p_i$ for all i. ($X^{\mathsf{lift}} \mod p_i$ denotes coordinate-wise reduction of X^{lift} modulo p_i.) Output $z \mod q$ as the evaluation result.

According to CRT, there exists a unique $z \mod p_1 \cdots p_h$ satisfying the conditions. However, $z = f^{\mathsf{lift}}(X^{\mathsf{lift}}) \mod p_1 \cdots p_h$. Wishing z to be exactly $f^{\mathsf{lift}}(X^{\mathsf{lift}})$, Lin et al. require $p_1 \cdots p_h$ to be large enough to avoid computation wrap over. Towards this end, they set $M = \max_{X \in \mathbb{Z}_q^m} f^{\mathsf{lift}}(X)$ and let $p_1, \ldots, p_h$ to be all the primes less than or equal $16 \log M$. (Since f has individual degree less than d in all of its m variables with coefficients from $\mathbb{Z}_q$, it can have at most d^m monomials. Hence, $M = d^m \cdot q \cdot q^{m(d-1)}$.) Based on Lemma 2.4 in [9], $M < p_1 \cdots p_h$. Since there is a unique $z < p_1 \cdots p_h$ that satisfies $z \equiv z_i \mod p_i$ for all i and $f^{\mathsf{lift}}(X^{\mathsf{lift}}) < M < p_1 \cdots p_h$ satisfies $f^{\mathsf{lift}}(X^{\mathsf{lift}}) \equiv f^{\mathsf{lift}}(X^{\mathsf{lift}} \mod p_i) \mod p_i$, $z = f^{\mathsf{lift}}(X^{\mathsf{lift}})$. Finally, $z \mod q = f^{\mathsf{lift}}(X^{\mathsf{lift}}) \mod q = f(X)$.

The preprocessing algorithm runs in time $\mathcal{O}(md \log q)^m \cdot \mathsf{poly}(m, d, \log q)$, and the evaluation algorithm runs in time $\mathsf{poly}(m, d, \log q)$. Lin et al. do

not stop here and repeat the same algorithm **once** again for polynomial evaluations over $\mathbb{Z}_{p_i}$'s. Since the $\log q$ factor comes from $\log M$ upper bound for the primes p_i, applying the algorithm once again changes the running time of the preprocessing algorithm to $d^m \cdot \mathsf{poly}(m, d, \log q) \cdot \mathcal{O}(m(\log m + \log d + \log\log q))^m$. The evaluation time remains the same. Lin et al. call this technique *two-level reduction*. Refer to Appendix A.1 in [11] for the running time analysis.

5.2 Polynomials over $R_q = \mathbb{Z}_q[x]/\langle h(x)\rangle$

Lin et al. introduce an idea for moving from ring $\mathbb{Z}_q$ to $R_q = \mathbb{Z}_q[x]/\langle h(x)\rangle$ that also applies when moving from R_q to $C = R_q[y]/\langle g(y)\rangle$.

The main idea for preprocessing and evaluation is Kronecker substitution (Section 8.4, [15]). Kronecker substitution encodes a polynomial $p(x) = \Sigma_{i=0}^{d} p_i x^i$ with an integer $p(M)$ by evaluating p over a value M greater than all the coefficients p_i. To recover the polynomial, it suffices to consider digits of $p(M)$ in base M as coefficients of p.

To motivate the approach, first interpret coefficients of $f \in R_q[X_1, \ldots, X_m]$ in $\mathbb{Z}[x]$ without reducing modulo q or $h(x)$. (Lift f to $f^{\mathsf{lift}} \in \mathbb{Z}[x][X_1, \ldots, X_m]$.) Moreover, interpret $X \in R_q^m$ as a member of $(\mathbb{Z}[x])^m$. (Lift X to X^{lift}.) Observe $\beta(x) = f^{\mathsf{lift}}(X^{\mathsf{lift}}) \in \mathbb{Z}[x]$ and let $\beta(x) = \Sigma_{i=0}^{D} \beta_i x^i$ with non-negative coefficients β_i and degree at most $D = (\deg h - 1)((d-1)m + 1)$. (Notice that $\deg X_i \leq \deg h - 1$ (recall $X_i \in R_q$) and $(d-1)m + 1$ members of R_q may be multiplied by each other in a monomial of f.)

With the Kronecker substitution in mind, it is desired that $\beta(M) \in \mathbb{Z}$ is computed for a large M greater than all β_i so that digits of $\beta(M)$ in base M are β_i. Since coefficients of X_i as polynomial over x are at most $q - 1$, multiplication of two such polynomials can create coefficients at most $\deg h(q-1)$. There are at most d^m monomials added in f and at most $(d-1)m + 1$ multiplications happen in a monomial. Therefore, $\beta_i \leq d^m(\deg h(q-1))^{(d-1)m+1}$, and $M = d^m(\deg h(q-1))^{(d-1)m+1} + 1$ is a strict upper bound for β_i.

Let $X^{\mathsf{lift}} \mod (x - M)$ denote coordinate-wise reduction of X^{lift} modulo $x - M$. Observe that if $f^{\mathsf{red}} = f^{\mathsf{lift}} \mod (x - M)$ and $X^{\mathsf{red}} = X^{\mathsf{lift}} \mod (x - M)$, $\beta(M) = f^{\mathsf{red}}(X^{\mathsf{red}})$ where $\beta = f^{\mathsf{lift}}(X^{\mathsf{lift}})$. Since $X_i^{\mathsf{red}} < \beta(M) < M^{D+1}$, using the last observation and letting $r = M^{D+1}$, $\beta(M)$ can be computed by evaluating $f^{\mathsf{red}} \in \mathbb{Z}_r[X_1, \ldots, X_m]$ over $X^{\mathsf{red}} \in \mathbb{Z}_r^m$, which is exactly the case discussed in Subsection 5.1. In other words, the preprocessing and evaluation algorithm for polynomial f^{red} over $\mathbb{Z}_r$ is invoked for the

evaluation on points $X^{\mathsf{red}} \in \mathbb{Z}_r^m$. (Notice that bounding X_i^{red} by r and preprocessing the whole $\mathbb{Z}_r$ is very inefficient as the bound is far from being tight.)

The preprocessing algorithm runs in time

$$d^m \cdot \mathsf{poly}(m, d, \log |R_q|) \cdot \mathcal{O}(m(\log m + \log d + \log\log |R_q|))^m$$

and the evaluation algorithm runs in time $\mathsf{poly}(d, m, \log |R_q|)$, determined by $\log r$ and $\log\log r$.

5.3 Polynomials over $C = R_q[y]/\langle g(y)\rangle$

Moving from R_q to $C = R_q[y]/\langle g(y)\rangle$ is identical by lifting evaluations to $\mathbb{Z}[x, y]$ and reducing f modulo $(y - M')$ for large enough M' to obtain evaluation problem of $f^{\mathsf{red}} \in \mathbb{Z}[x][X_1 \ldots, X_m]$ over $\mathbb{Z}[x]$. Similar to Subsection 5.2, some r' is chosen to bound coefficients of f^{red}. Moreover, a large D' is also chosen to bound the degree of computations of f^{red} so that they do not wrap over $(x^{D'} + 1)$. This gives a polynomial evaluation problem over $\mathbb{Z}_{r'}[x]/\langle x^{D'} + 1\rangle$ such that $X^{\mathsf{red}} \in (\mathbb{Z}_{r'}[x]/\langle x^{D'} + 1\rangle)^m$, which is exactly the previous case discussed in Subsection 5.2. Refer to Appendix A.2 in [11] for more details. The running time of preprocessing and evaluation algorithms do not change except that C replaces R_q.

Again, forcefully instantiating a problem of the previous case by modding out r' and $x^{D'} + 1$ is very inefficient and causes the preprocessing to store many redundant values in the ring $(\mathbb{Z}_{r'}[x]/\langle x^{D'} + 1\rangle)^m$.

5.4 Optimizations by Okada et al. [13]

Okada et al. [13] introduce an optimization technique to avoid Kroenecker substitutions altogether by leveraging the algebraic structure of $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ when $q = p_1 \cdots p_r$, $p_i = 1 \mod 2n$, and $x^n + 1$ is a cyclotomic polynomial, i.e., n is a power of two. (Recall ciphertext space of ASHE scheme was $R_q[y]/\langle y^D + 1\rangle$ where R_q was derived from BV's SHE scheme as quotient over a cyclotomic polynomial.) In such a situation, R_q splits into prime ideals, and so $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle \cong \bigoplus_{i=1}^{r} \bigoplus_{j=1}^{n} \mathbb{F}_{p_i}$. Then, they choose prime $k > D$ (to avoid wrap over modulo D, the total degree of the encoded polynomial) and redefine the ciphertext space of ASHE scheme C as below when $p_i = 1 \mod 2nk$:

$$C' \cong \mathbb{Z}_q[y]/\langle y^k - 1\rangle \otimes \mathbb{Z}_q[x]/\langle x^n + 1\rangle \cong \mathbb{Z}_q[x]/\langle x^{nk} + 1\rangle \cong \bigoplus_{i=1}^{r} \bigoplus_{j=1}^{nk} \mathbb{F}_{p_i}$$

Refer to Fourier decomposition of ciphertexts outlined in Section 6 in work of Okada et al. [13] for more details.

6 DEPIR definition and main construction

Having all the tools ready, this Section starts with the formal definition of DEPIR (syntax, correctness, security, and efficiency) and proceeds to explain the construction and why it satisfies the definition.

6.1 Definition

Syntax
A DEPIR scheme consists of four algorithms $\mathsf{Preproc}$, Query, Resp, and Dec.

- $\widetilde{\mathsf{DB}} = \mathsf{Preproc}(\mathsf{DB}, 1^\lambda)$ is a deterministic algorithm (notice notation $=$ instaed of $\leftarrow\!\!\$$) that inputs a database $\mathsf{DB} \in \{0,1\}^N$ and security parameter λ and outputs the preprocessed database $\widetilde{\mathsf{DB}}$, which is stored by the server. The server can run this algorithm independently and only once before all queries of all clients, i.e., not limited to one client. Moreover, it does not interfere with the scheme security as the preprocessed database is information-theoretically equivalent to the initial database and only optimized for evaluation queries. In fact, for PIR schemes, $\widetilde{\mathsf{DB}}$ can be simply the encoding of DB as a polynomial as discussed in Section 3. The running time of this algorithm and its output size are known as preprocessing time and server storage size, respectively.

- $(c, s) \leftarrow\!\!\$\, \mathsf{Query}(1^\lambda, N, i)$ is a probabilistic algorithm that inputs the security parameter λ, database size N, and database index $i \in \{1, \ldots, N\}$ and outputs the ciphertext c to be sent to the server and secret key s to be stored locally by the client for decryption of server response. The secret key is only applicable for this query.

- $a = \mathsf{Resp}(\widetilde{\mathsf{DB}}, c)$ is a deterministic algorithm that inputs the preprocessed database $\widetilde{\mathsf{DB}}$ and ciphertext c sent by the client and outputs the encrypted value in the database corresponding to the index i.

- $b = \mathsf{Dec}(s, a)$ is a deterministic algorithm run by the client that inputs the secret key s and server answer a and outputs bit b at position i in

the database.

Surprisingly, query complexity is the collective running time of Query, Resp, and Dec. Communication complexity is determined by the output size of Query and Resp.

Correctness

It is expected of an honest execution of the protocol for all indices i and all databases DB to satisfy $\Pr[\mathsf{Dec}(s,a) = \mathsf{DB}[i]] = 1$ where $\widetilde{\mathsf{DB}} = \mathsf{Preproc}(\mathsf{DB}, 1^\lambda)$, $(c,s) \leftarrow\!\!\$\, \mathsf{Query}(1^\lambda, N, i)$, $a = \mathsf{Resp}(\widetilde{\mathsf{DB}}, c)$, and the probability is over the random choices of the only probabilistic algorithm Query.

Security

The security requirement is that for all probabilistic polynomial time adversary $\mathcal{A}$ and security parameters λ, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $|\Pr\left[\mathsf{Exp}^{\mathsf{DEPIR}}_{\mathcal{A}}(1^\lambda) = 1\right] - 1/2| \leq \mathsf{negl}(\lambda)$ where:

$\underline{\mathsf{Exp}^{\mathsf{DEPIR}}_{\mathcal{A}}(1^\lambda)}$

$(\mathsf{st}, i_0, i_1, 1^N) \leftarrow\!\!\$\, \mathcal{A}(1^\lambda)$

$b \leftarrow\!\!\$\, \{0,1\}$

$(c,s) \leftarrow\!\!\$\, \mathsf{Query}(1^\lambda, N, i_b)$

$b' \leftarrow\!\!\$\, \mathcal{A}(\mathsf{st}, c)$

if $b = b'$ **then**

  return 1

return 0

In other words, the adversary should not be able to distinguish between encryptions of different queried indices of the databases even if they were chosen by itself. The similarity to the IND-CPA security implies that security can be reduced to that of a possible encryption scheme used by Query. Since, in the context of DEPIR, even the server can be the adversary by observing clients' encrypted queries or making new queries itself, it is assumed the adversary has access to the database. (Notice that this makes the notion even stronger.) The state information st is common in security experiment definitions that require the adversary to interact with the challenger. Moreover, Resp is not modeled in the game as it can be simulated by the adversary internally. Finally, $\mathsf{Preproc}$ as a deterministic algorithm can be simulated by the adversary, and as mentioned before, it does not affect security at all.

Efficiency

An scheme consisting of four algorithms $\mathsf{Preproc}$, Query, Resp, and Dec is doubly efficient if $\mathsf{Preproc}$ runs in polynomial time with respect to λ and database size N, i.e., $\mathsf{poly}(\lambda, N)$ (ideally qausilinear in N, i.e., $N\mathsf{poly}(\log N)$), and Query, Resp, and Dec, collectively, runs in time sublinear in N (ideally polylogarithmic in N).

6.2 Construction

Two public parameters d and m are chosen based on database size N such that d is prime and $d^m \geq N$. Lin et al. [11] describe two options for choosing them. The reader is referred to page 23 of their work for an elaborate discussion of the choice of parameter effect on the efficiency. The ASHE scheme is set up with $\mathsf{ASHE.Setup}(1^\lambda, 1^d, 1^D = 1^{dm}, d^m)$ using the scheme security parameter λ, d as before, and dm as the upper bound for the total degree D of f_{DB} with at most d^m terms. This step defines the rings R_q and C (ciphertext ring).

The preprocessing algorithm encodes the database DB as a polynomial f_{DB} explained in Section 3. The polynomial coefficients are *lifted* from $\mathbb{F}_d$ to C, resulting in $f_{\mathsf{DB}}^{\mathsf{lift}}$. (Refer to Section 4.2 for the definition of *lifting*.) Then, the resulting lifted m-variate polynomial $f_{\mathsf{DB}}^{\mathsf{lift}}$ of individual degree less than d over R_q undergoes the preprocessing algorithm of Section 5, and the resulting evaluation tables $\widetilde{\mathsf{DB}}$ are stored by the server in random-access memory. The query algorithm finds the digits of $i = (i_1, \ldots, i_m)$ in base d, generates a secret key $s \leftarrow\!\!\$\, \mathsf{ASHE.Gen}(1^\lambda)$, obtains the encryption of each digit $c_j \leftarrow\!\!\$\, \mathsf{ASHE.Enc}(s, i_j)$, and outputs $(c_1, \ldots, c_m)$ and s. The server response generator algorithm uses the fast polynomial evaluation algorithm introduced in Section 5 and outputs $a = f_{\mathsf{DB}}^{\mathsf{lift}}(c_1, \ldots, c_m)$ using random access to tables $\widetilde{\mathsf{DB}}$. The decryption algorithm simply outputs $\mathsf{ASHE.Dec}(a, s)$.

Security, as hinted in Subsection 6.1, is reduced to that of ASHE, since the adversary (including the server) only sees m ciphertexts for each query. Lin et al. discuss how weaker security notions of ASHE can be used by adjusting the parameters in Section 4.2 of their work [11].

The preprocessing algorithm running time is bounded by the encoding and the polynomial preprocessing steps taking $\mathcal{O}(d^m \cdot m \cdot \mathsf{poly}(\log d))$ and $d^m \cdot m^m \cdot \mathsf{poly}(m, d, \log |C|) \cdot \mathcal{O}(\log m + \log d + \log\log |C|)^m$, respectively. Note that $\log |C| = \log q^{Dn} = Dn \log q = \mathsf{poly}(\lambda, D, \log d, \log N) = \mathsf{poly}(\lambda, d, m, \log d) = \mathsf{poly}(\lambda, d, m)$. (Refer to choice of parameter q in Sec-

tion 3.1 of [11].) The query complexity is bounded by the ASHE key generation, m copies of ASHE encryption, and server response, which is in turn bounded by the running time of the fast polynomial evaluation algorithm $\mathcal{O}(d, m, \log |C|)$ mentioned in Section 5. Each of these algorithms is bounded by $\mathsf{poly}(\lambda, d, m)$.

For any $\epsilon > 0$, Lin et al. [11] choose d to be the first prime greater than $\log^{2/\epsilon} N$ and achieve query complexity $\mathsf{poly}(\lambda, \log N)$ with preprocessing complexity $\mathcal{O}(N^{1+\epsilon})\mathsf{poly}(\lambda, \log N)$.

7 Conclusion and future work

This paper briefly discusses the main tools used for DEPIR construction to accelerate the literature review process for future researchers. Moreover, it compiles the related work, such as that of Brakerski and Vaikuntanathan [4], Okada et al [13], and Kedlaya and Umans [9]. More importantly, it hints at the connection between the work of Lin et al. [11] and previous research, including the relation between ASHE and BV's scheme in Section 4.2 and the use of Kronecker substitution for encoding polynomials as integers in Section 5.

However, the work of Lin et al. [11] is much more comprehensive. They outlined an algorithm to update an already preprocessed database and applied their ideas behind ASHE to construct a RAM-FHE scheme. The work of Okada et al. [13] also includes other optimization techniques including one for database encoding.

The most important observation of this paper for future development is the inefficiency of polynomial preprocessing using the Kroenecker substitution, producing large integers. Subsections 5.2 and 5.3 pinpoint several issues along the discussion. Additionally, Okada et al. [13] demonstrated the impracticality of the naive implementation of the work of Lin et al. [11]. They have also asked for alternative ASHE schemes with better noise growth and ring size. According to them, common techniques for decreasing noise and ring size, such as modulus-switching and relinearization, are not algebraic in nature, and, hence, incompatible with ASHE.

References

[1] Dmitri Asonov. *Querying Databases Privately: A New Approach to Private Information Retrieval*, volume 3128 of *Lecture Notes in Computer Science*. Springer, 2004. https://doi.org/10.1007/B98671.

[2] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 55–73. Springer, 2000. https://doi.org/10.1007/3-540-44598-6_4.

[3] P. B. Bhattacharya, S. K. Jain, and S. R. Nagpaul. *Basic Abstract Algebra*. Cambridge University Press, November 1994. https://doi.org/10.1017/CBO9781139174237.

[4] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages, 2011. https://doi.org/10.1007/978-3-642-22792-9_29.

[5] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, SFCS-95. IEEE Comput. Soc. Press, 1995. https://doi.org/10.1109/sfcs.1995.492461.

[6] Eric Fung, Georgios Kellaris, and Dimitris Papadias. Combining differential privacy and PIR for efficient strong location privacy. In Christophe Claramunt, Markus Schneider, Raymond Chi-Wing Wong, Li Xiong, Woong-Kee Loh, Cyrus Shahabi, and Ki-Joune Li, editors, *Advances in Spatial and Temporal Databases - 14th International Symposium, SSTD 2015, Hong Kong, China, August 26-28, 2015. Proceedings*, volume 9239 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2015. https://doi.org/10.1007/978-3-319-22363-6_16.

[7] Joseph Gallian. *Contemporary Abstract Algebra*. Chapman and Hall/CRC, January 2020. https://doi.org/10.1201/9781003142331.

[8] Daniel Günther, Maurice Heymann, Benny Pinkas, and Thomas Schneider. Gpu-accelerated pir with client-independent preprocessing for large-scale applications. Cryptology ePrint Archive, Paper 2021/823, 2021. https://eprint.iacr.org/2021/823.

[9] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, January 2011. https://doi.org/10.1137/08073408x.

[10] Ali Khoshgozaran, Houtan Shirani-Mehr, and Cyrus Shahabi. Spiral: A scalable private information retrieval approach to location privacy. pages 55–62, Beijing, China, 2008. IEEE. https://doi.org/10.1109/mdmw.2008.23.

[11] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic ram computation from ring lwe. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*,

STOC 2023, page 595–608, New York, NY, USA, 6 2023. Association for Computing Machinery. https://doi.org/10.1145/3564246.3585175.

[12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. page 1–35, 11 2013. https://doi.org/10.1145/2535925.

[13] Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. Towards practical doubly-efficient private information retrieval. Cryptology ePrint Archive, Paper 2023/1510, 2023. https://eprint.iacr.org/2023/1510.

[14] Ian Stewart and David Tall. *Algebraic Number Theory and Fermat's Last Theorem: Third Edition*. A K Peters/CRC Press, December 2001. https://doi.org/10.1201/9781439864081.

[15] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. https://doi.org/10.1017/CBO9781139856065.

[16] Lawrence C. Washington. *Introduction to Cyclotomic Fields*. Springer New York, NY, 2 edition, 1997. https://doi.org/10.1007/978-1-4612-1934-7.

Predicting Depression through Digital Phenotyping

Antti Kokkonen
antti.j.kokkonen@aalto.fi

Tutor: Arsi Ikäheimonen

Abstract

Depression affects a significant part of the population globally. Like other mental disorders, depression lacks reliable biomarkers and typically psychiatric diagnoses are based on clinical interviews. This adds to the already great burden of the global health-care and also the patients might not be able to report the changes in their mental state accurately. As an option for this health data could be collected from mobile devices and with these data and so called digital phenotyping we can try to create behavioral markers that are related to the patient's symptoms. And with these markers and different statistical and machine learning methods we can create predictive models to estimate current and future depressive state of the patient.

This paper reviews how depression can be predicted using digital phenotyping. The current work is a literature review and 11 scientific articles were selected for it. The paper reviews what types of data can be collected, what machine learning algorithms can be used for the models, how these models perform and what are their limitations.

From the selected articles most common data collected were related to activity, location or sleep of the participants. Most of the articles selected used machine learning models and the most common algorithm used was Random forest. The models were generally able to predict depression with good accuracy and show digital phenotyping as a viable option for pre-

dicting depression. However, there were limitations and more than half of the studies warned that there are no guarantees that their work can be generalized outside their participant groups.

KEYWORDS: depression; digital phenotyping

1 Introduction

Depression affects a significant part of the population globally. In the 2013 study by Ferrari et al. [1] the global prevalence of major depressive disorder (MDD) was estimated to be around 5 %. According to the Global Burden of Disease Consortium [2] MDD was the second leading contributor to the global disease burden in 2013. The symptoms of depression include depressed mood, diminished interests, impaired cognitive function, vegetative symptoms and it can even lead to a suicide [3]. Like other mental disorders MDD lacks reliable biomarkers [4] and typically psychiatric diagnoses are based on clinical interviews, which adds to the already great burden of the global health-care. In addition, the patients might not be able to report the changes in their mental state accurately [5].

To answer the challenges just mentioned, health data could be collected from mobile devices. It is estimated that in 2023 there were globally almost 17 billion mobile devices and over seven billion mobile device users [6]. These smartphones and wearable devices can be used to collect data through the user interaction with the device or the devices' different sensors. With so called digital phenotyping, which Torous et al. [7] define as "moment-by-moment quantification of the individual-level human phenotype in situ using data from digital devices", we can try to create behavioral markers that are related to the patient's symptoms. And with these markers and different statistical and machine learning methods we can create predictive models to estimate current and future depressive state of the patient.

This paper reviews how depression can be predicted using digital phenotyping. The current work is a literature review and 11 scientific articles were selected for this paper. The focus is on how passive data from smartphones and wearable devices can be used for predicting depression changes and which machine and deep learning strategies are used for these predictive models.

The structure of this paper is the following: First the methods for this literature review are introduced in section 2. Then, in section 3, the results of this review are presented. After that, in section 4, we discuss the results and limitations of predicting depression with digital phenotyping and also the limitations of this review. Finally, section 5 gives a conclusion of the contents of this paper.

2 Methods

For this literature review a handful of scientific articles were selected. The articles were searched from the Scopus database. The goal was to select articles that researched predicting depression changes or future depressive state through digital phenotyping using passive data collected from smartphones and wearable devices. Some of the selected articles do not exclusively discuss just depression but also other mental disorders as well such as bipolar disorder (BDD) or general anxiety disorder (GAD).

First, a simple search query '"depressi*" AND "digital phenotyp*"' was formed. Here the '*' symbol means that after the preceeding characters in the string zero or more of any character may follow. Using the filters of the Scopus search the results were limited to just articles (excluding review articles).

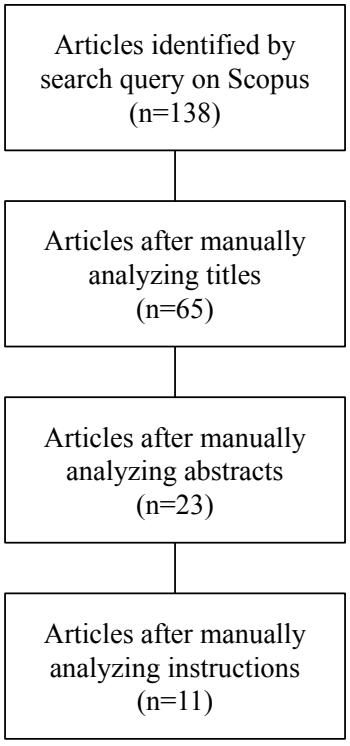


Figure 1. Article screening process

For the article screening, which figure 1 describes, the PRISMA [8] standard was applied. After the initial search the results were manually reviewed. First the titles were analyzed and unfitting articles were excluded. Then the remaining articles were again reviewed this time based on the abstract and again the unsuitable articles were excluded. Finally, the introductions to the articles were analyzed and the final selection was made. Also at this point two articles were excluded because the full texts were not accessible for free. In the end, 11 articles were selected for this review.

3 Results

This section provides the results of the review. The different types of data collected in the reviewed articles are presented in section 3.1 with focus on the more common types of data. After that, in section 3.2, the machine learning algorithms used for the predictive models of the reviewed articles are presented.

Author	Year	Passive data source	Machine learning methods	Results
Cho et al.	2019	Activity, Heart rate, Light exposure, Sleep	Random forest	Depressive episode was predicted with 85.3 % accuracy.
Jacobson and Chung	2020	Calls and text messages, Light exposure, Location	Nomoethic extreme boosting, Random forest	Depressed mood was predicted from hour to hour with an average correlation of 0.587.
Meyerhoff et al.	2021	Calls and text messages, Location, Phone usage	-	Changes in behavioral features predicted changes in depression but not vice-versa.
Zhang et al.	2021	Location	Hierarchical Bayesian linear regression	Depression severity was predicted with the prediction metrics R^2=0.526, $RMSE$=3.891.
Liu et al.	2022	Calls and text messages, Location	Histogram-based gradient boost trees, Logistic regression	Text message sentiment among other features were able to strongly predict depression status.
Bai et al.	2023	Activity, Calls and text messages, Heart rate, Location, Phone usage, Sleep	Decision trees, K-nearest neighbors, Logistic regression, Naive Bayes, Random forest, Support vector machine	Changes in mood states were predicted with an accuracy of 76.67 % at best.
Currey et al.	2023	Activity, Location, Phone usage	Logistic regression	A 50 % change in mood could be detected with 28 days of data.
Fried et al.	2023	Activity, Heart rate, Sleep, Stress	-	-
Lee H.-J. et al.	2023	Activity, Heart rate, Light exposure, Sleep	Random forest	Impending MDE for the next three days was predicted with 90.1 % accuracy.
Lee T. et al.	2023	Activity, Light exposure, Sleep	Hidden Markov model, Random forest, Recurrent neural network	Future depressive episodes were predicted with 78 % accuracy.
Ross et al.	2023	Activity, Phone usage	Deep learning neural networks, Gradient boosting, Random forest	Changes in depression predicted with 95 % accuracy.

Figure 2. Summary of the selected articles

Figure 2 shows a summary of the selected articles. More than half of the selected articles were published after 2021 and only two were pub-

lished before 2021. All of the articles were published less than five years ago.

3.1 Data collection

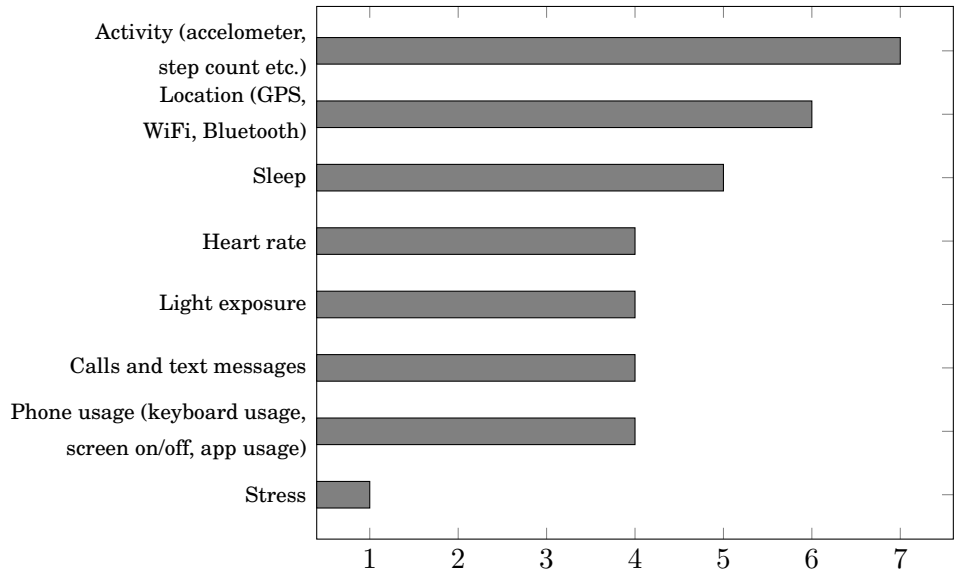


Figure 3. Types of collected data in the reviewed material

Figure 3 shows the quantities of the types of data collected in the reviewed articles. Between the different types there were some intersections of which there are some examples in this section. There were differences in how detailed the articles described the data collected.

The most common type of data was activity based data which was collected in seven of the eleven articles [9, 10, 11, 12, 13, 14, 15]. Three of these measured the step count of the users [9, 10, 13]. Curry et al. [11] and Ross et al. [15] used accelometer data, latter of which measured accelometer data only when the patient was still and typing with their smartphone. Fried et al. [12] and Lee T. et al. [14] just mention measuring activity without specifying which sensors were used.

As mentioned before there were some intersections with other types of data collected. Curry et al. [11] determined sleep duration from accelometer data and Bai et al. [9] used activity to determine whether the user was active, in light sleep, in deep sleep or not wearing the tracker.

The second most common type of data was location based data. Six of the selected articles [9, 11, 16, 17, 18, 19] used that in their studies. Almost all utilized Global Positioning System (GPS). Only Jacobson and Chung [16] used WiFi whenever GPS was not available and Zhang et al. [19] measured Bluetooth devices in the physical proximity of the

patient.

In addition to using just GPS location itself a lot of other features were extracted from the data as well such as velocity and other movement related features [16, 17, 18], which intersect with the activity based data. Almost all of the six studies also determined the type of the locations visited or some of them. Currey et al. [11] and Liu et al. [17] measured time spent at home. Meyerhoff et al. [18] measured time spent at home, work, shopping, social activites, religoius activites and excercise locations based on labels assigned manually for each location. Jacobson and Chung [16] recorded the location type based on Google Places location and local weather information of the current location.

Also Zhang et al. [19] measured number of Bluetooth connections around the user in different location types and with that estimated social connections and interactions with family, friends, co-workers and strangers. They also had features for time at home, mobility, social isolation and working status.

After location based data the most favoured type of data was sleep related data. Five of the selected articles [9, 10, 12, 13, 14] collected sleep data. All of the five studies measured sleep duration and all except Fried et al. [12] measured sleep efficiency and circadian rhythm related data. Bai et al. [9] and Fried et al. [12] collected data also about sleep phases.

3.2 Predictive models

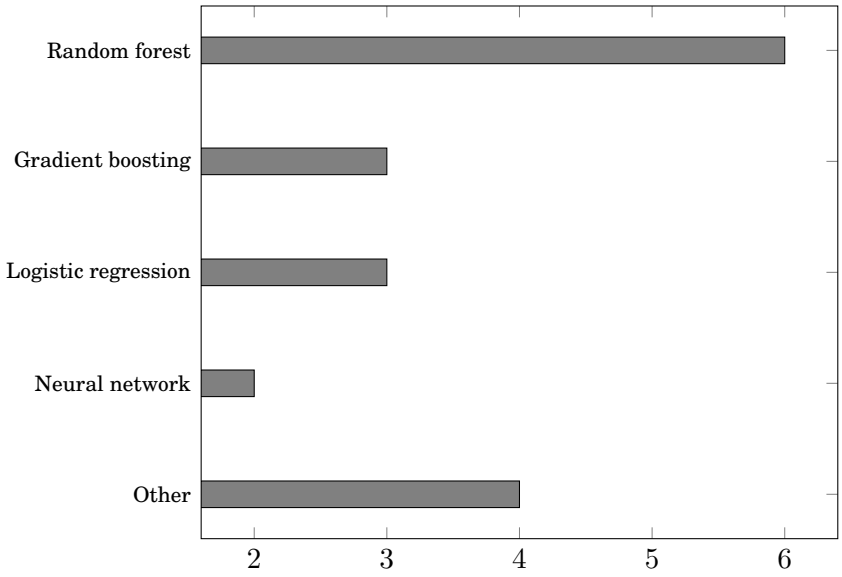


Figure 4. Machine learning algorithms in the reviewed articles

Figure 4 shows the different machine learning algorithms used to train the predictive models in the reviewed articles and their quantities. The most commonly used algorithm was Random Forest [9, 10, 13, 14, 15, 16] which was used in over half of the articles. Gradient boosting appeared in three articles [15, 16, 17] and Logistic regression as well [9, 11, 17]. Jacobson and Chung [16] utilized predictions of a nomothetic extreme gradient boosting algorithm as secondary features for a random forest model.

Some studies tried multiple algorithms [9, 14, 15, 17] and compared the results. Lee T. et al. [14], in addition to models constructed with Hidden Markov model, Recurrent neural network and Random forest, made a hybrid model using all of the previously mentioned. Fried et al. [12] and Meyerhoff et al. [18] had no mentions of machine learning algorithms in their studies.

The prediction model accuracies were fairly good. Ross et al. [15] were able to predict relevant changes in PHQ score with accuracy from 94 to 95.5 %. Lee H-J. et al. [14] got their highest accuracy of 79 % with Recurrent neural network model. Cho et al. [10] managed to predict a depressive episode (DE) with 87 % accuracy and Jacobson and Chung [16] predicted depressed mood scores with 95 % accuracy. Lee T. et al. [13] achieved the accuracy of 90.1 % in predicting impending major depressive episode (MDE). Bai et al. [9] tried also multiple different models with the highest accuracy of 85.42 %.

4 Discussion

Among the selected articles a lot of different data was collected but the most common were about activity, location or sleep of the participants. Most used machine learning method was random forest and gradient boosting with logistic regression after that. The created models were generally able to predict depression changes with good accuracy.

The reviewed articles reported many limitations to their work and many cautioned against generalizing their results without care. More than half of the studies warned that there are no guarantees that their work can be generalized outside their participant groups [10, 13, 14, 16, 17, 18]. Also at least for Ross et al. [15], Jacobson and Chung [16] and Liu et al. [17] the participants' symptom severities were determined by self-reporting and were not confirmed clinically. In addition, a common limitation was operating system related issues [9, 13, 18].

This literature review had also some limitations. Firstly, some studies here did not use only passive data but active data as well and the research was not restricted only to MDD. Also, because of the scope of this paper the number of reviewed articles was quite small. In addition, one of the articles [12] selected was a study design and had no complete results. Finally, the search query might not have been ideal and only one search portal was used which may have left articles out of the review.

5 Conclusion

This paper studied predicting depression through digital phenotyping. It was a literature review and 11 scientific articles were selected for it. In the paper some common data collection types and machine learning models were reviewed. Also, the model accuracies and some limitations were discussed.

The selected articles collected different types of passive data. Collecting data based on activity, location or sleep was the most common and this paper reviewed them in more detail. Also different machine learning algorithms were used including Random forest, Gradient boosting, Logistic regression and Neural networks. Of these Random forest was the most common one. Many of the selected articles were able to predict depression with good accuracy and show digital phenotyping as a viable option for predicting depression. However, there were limitations and more research should be done to generalize the results.

References

[1] A. J. Ferrari, A. J. Somerville, A. J. Baxter, R. Norman, S. B. Patten, T. Vos, and H. A. Whiteford, "Global variation in the prevalence and incidence of major depressive disorder: a systematic review of the epidemiological literature," *Psychological Medicine*, vol. 43, no. 3, pp. 471–481, Mar. 2013.

[2] Global Burden of Disease Study 2013 Collaborators, "Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990-2013: a systematic analysis for the Global Burden of Disease Study 2013," *Lancet (London, England)*, vol. 386, no. 9995, pp. 743–800, Aug. 2015.

[3] C. Otte, S. Gold, B. Penninx, C. Pariante, A. Etkin, M. Fava, D. Mohr, and A. Schatzberg, "Major depressive disorder," *Nature Reviews Disease Primers*, vol. 2, 2016.

[4] P. Boksa, "A way forward for research on biomarkers for psychiatric dis-

orders," *Journal of Psychiatry and Neuroscience*, vol. 38, no. 2, pp. 75–77, 2013.

[5] T. Aledavood, A. M. Triana Hoyos, T. Alakörkkö, K. Kaski, J. Saramäki, E. Isometsä, and R. K. Darst, "Data Collection for Mental Health Studies Through Digital Platforms: Requirements and Design of a Prototype," *JMIR Research Protocols*, vol. 6, no. 6, p. e110, Jun. 2017.

[6] "Mobile statistics report, 2021-2025," THE RADICATI GROUP, INC, Tech. Rep., 2021.

[7] J. Torous, M. V. Kiang, J. Lorme, and J.-P. Onnela, "New Tools for New Research in Psychiatry: A Scalable and Customizable Platform to Empower Data Driven Smartphone Research," *JMIR Mental Health*, vol. 3, no. 2, p. e5165, May 2016.

[8] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher, "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, vol. 372, p. n71, Mar. 2021.

[9] R. Bai, L. Xiao, Y. Guo, X. Zhu, N. Li, Y. Wang, Q. Chen, L. Feng, Y. Wang, X. Yu, C. Wang, Y. Hu, Z. Liu, H. Xie, and G. Wang, "Tracking and Monitoring Mood Stability of Patients With Major Depressive Disorder by Machine Learning Models Using Passive Digital Data: Prospective Naturalistic Multicenter Study," *JMIR mHealth and uHealth*, vol. 9, no. 3, p. e24365, Mar. 2021.

[10] C.-H. Cho, T. Lee, M.-G. Kim, H. P. In, L. Kim, and H.-J. Lee, "Mood Prediction of Patients With Mood Disorders by Machine Learning Using Passive Digital Phenotypes Based on the Circadian Rhythm: Prospective Observational Cohort Study," *Journal of Medical Internet Research*, vol. 21, no. 4, p. e11029, Apr. 2019.

[11] D. Currey, R. Hays, and J. Torous, "Digital Phenotyping Models of Symptom Improvement in College Mental Health: Generalizability Across Two Cohorts," *Journal of Technology in Behavioral Science*, vol. 8, no. 4, pp. 368–381, Mar. 2023.

[12] E. I. Fried, R. K. K. Proppert, and C. L. Rieble, "Building an Early Warning System for Depression: Rationale, Objectives, and Methods of the WARN-D Study," *Clinical Psychology in Europe*, vol. 5, no. 3, pp. 1–25, Sep. 2023.

[13] H.-J. Lee, C.-H. Cho, T. Lee, J. Jeong, J. W. Yeom, S. Kim, S. Jeon, J. Y. Seo, E. Moon, J. H. Baek, D. Y. Park, S. J. Kim, T. H. Ha, B. Cha, H.-J. Kang, Y.-M. Ahn, Y. Lee, J.-B. Lee, and L. Kim, "Prediction of impending mood episode recurrence using real-time digital phenotypes in major depression and bipolar disorders in South Korea: a prospective nationwide cohort study," *Psychological Medicine*, vol. 53, no. 12, pp. 5636–5644, Sep. 2023.

[14] T. Lee, H.-J. Lee, J.-B. Lee, and J.-D. Kim, "Ensemble Approach to Combining Episode Prediction Models Using Sequential Circadian Rhythm Sensor

Data from Mental Health Patients," *Sensors*, vol. 23, no. 20, p. 8544, Jan. 2023.

[15] M. K. Ross, T. Tulabandhula, C. C. Bennett, E. Baek, D. Kim, F. Hussain, A. P. Demos, E. Ning, S. A. Langenecker, O. Ajilore, and A. D. Leow, "A Novel Approach to Clustering Accelerometer Data for Application in Passive Predictions of Changes in Depression Severity," *Sensors*, vol. 23, no. 3, p. 1585, Jan. 2023.

[16] N. C. Jacobson and Y. J. Chung, "Passive Sensing of Prediction of Moment-To-Moment Depressed Mood among Undergraduates with Clinical Levels of Depression Sample Using Smartphones," *Sensors*, vol. 20, no. 12, p. 3572, Jan. 2020.

[17] T. Liu, J. Meyerhoff, J. C. Eichstaedt, C. J. Karr, S. M. Kaiser, K. P. Kording, D. C. Mohr, and L. H. Ungar, "The relationship between text message sentiment and self-reported depression," *Journal of affective disorders*, vol. 302, pp. 7–14, Apr. 2022.

[18] J. Meyerhoff, T. Liu, K. P. Kording, L. H. Ungar, S. M. Kaiser, C. J. Karr, and D. C. Mohr, "Evaluation of Changes in Depression, Anxiety, and Social Anxiety Using Smartphone Sensor Features: Longitudinal Cohort Study," *Journal of Medical Internet Research*, vol. 23, no. 9, p. e22844, Sep. 2021.

[19] Y. Zhang, A. A. Folarin, S. Sun, N. Cummins, Y. Ranjan, Z. Rashid, P. Conde, C. Stewart, P. Laiou, F. Matcham, C. Oetzmann, F. Lamers, S. Siddi, S. Simblett, A. Rintala, D. C. Mohr, I. Myin-Germeys, T. Wykes, J. M. Haro, B. W. J. H. Penninx, V. A. Narayan, P. Annas, M. Hotopf, R. J. B. Dobson, and RADAR-CNS Consortium, "Predicting Depressive Symptom Severity Through Individuals' Nearby Bluetooth Device Count Data Collected by Mobile Phones: Preliminary Longitudinal Study," *JMIR mHealth and uHealth*, vol. 9, no. 7, p. e29840, Jul. 2021.

Technical survey of ad blockers

Artem Perevedentsev
artem.perevedentsev@aalto.fi

Tutor: Tuomas Aura

Abstract

Currently, online advertising is increasingly prevalent on the internet due to the fact that modern users spend more time online. However, many users are dissatisfied with the spread of online advertising because it distracts them from the actual content and because marketing companies collect personal data to improve ad effectiveness. The use of programmatic solutions to block advertisements has become necessary.

This paper describes the various types of internet advertisements and the architecture of different ad-blocking techniques. It also analyzes recent works related to circumventing ad-blockers by marketing companies and the impact of ad-blockers on users' behavior.

KEYWORDS: Ad Blocker, Online Advertising, Privacy

1 Introduction

In the digital age, the internet has provided advertisers an effective way to spread information about their products. Currently, the revenue of online marketing is actively rising, and there are at least two main reasons for this phenomenon. First, the cost of distributing online advertisements is significantly lower compared to television, radio, and newspaper ads.

Secondly, recent studies suggest that modern individuals spend a significant amount of time on various websites, social media, and video-sharing platforms. According to Montag [6], an average person spends about 3 hours a day using a cell phone. Thus, large marketing companies shifted towards the new strategy of advertisement distribution.

However, concerns about advertising on the internet have become increasingly common in recent times. The presence of advertisements on websites distracts users from the actual content and makes it inconvenient for them to interact with the site. Another concern relates to the increase in internet traffic, which remains expensive presently. Nah [7] suggests that longer loading times for webpages can significantly decrease the number of site visits. Another major issue that arises is the violation of users' privacy. Large companies tend to collect users' data to make advertisements more relevant. For example, Google and Facebook have collected petabytes of users data, and there is no guarantee that this data will be stored safely.

To eliminate these problems, people have started to create special software which helps to decrease the number of advertisement seen online. These programs are referred to as *ad blockers* and take various forms, such as stand-alone applications, web browser extensions, and external services. The main idea of ad blockers is to filter content before it is displayed on the user's screen. By following this approach, it is possible to eliminate various advertising elements. However, marketing companies are continually enhancing their methods for delivering ads to users, and ad blockers must also continually evolve.

This paper surveys different approaches to block online advertising. The rest of the paper is structured as follows. Section 2 describes the different types of online advertisements. Then Section 3 outlines the architectures of modern ad blockers. Section 4 explains methods to counter the use of ad blockers. Section 5 discusses the impact of ad blockers on the users' privacy. And finally, Section 6 summarizes and concludes the analysis of the papers.

2 Types of online ads

There is a wide variety of online advertising. Internet marketing constantly evolves and adapts to new technologies and new types of content. This section lists the most common types of ads that can be found on the

internet today.

2.1 Web banners

Web sites are HTML documents that are downloaded from a server to the user's computer and displayed on the screen by a browser program. HTML is a markup language that instructs the browser on where to display specific information. The displayed information can take the form of text, images, or videos. One of the earliest forms of internet advertising involved adding designated areas for advertisements within a webpage source code. These areas are known as *web banners* and are usually placed next to the actual content. There are two types of web banners: static and dynamic. A static web banner contains an advertisement at the moment of loading the page from the server, while a dynamic web banner loads the advertisement from a third-party server.

2.2 Pop-up windows

During the loading of a web page, the browser may execute JavaScript code. Usually, JavaScript is used to implement interactivity that can not be achieved by using HTML and CSS. However, JavaScript is a general-purpose programming language, which means that it can be used to implement any algorithm. Thus, while the site is loading, an arbitrary algorithm could potentially be executed on the user's computer. This feature is actively used by marketing companies. For example, it is possible to open additional browser windows and display advertisements in them. This method of displaying ads is named *pop-up windows* and was popular in the 1990s but is rarely used today because modern web browsers have started to block pop-up windows by default.

2.3 Advertising in search engines

The internet is vast, making it challenging to locate the necessary information. To solve this problem, *search engines* index all the information found on the internet. The search engines process a large number of queries each day, and only large corporations are capable of maintaining the necessary infrastructure. To maintain the infrastructure, search companies had to find sources of monetization. The solution was found: displaying advertisements in the search results. Each query displays advertisements as the top results. Marketing companies pay search engine

developers for the opportunity to appear higher in search results.

2.4 Social media

Social media has revolutionized the advertising industry. Due to the large audiences of social networks and the ability for users to directly interact with each other, it is possible to work with consumers in a more targeted manner. This effect is mainly achieved by collecting and processing a large amount of personal data from users.

2.5 Email spam

Email is one of the oldest approaches to distribute ads online. It is a convenient tool for marketers since it enables sending advertisements to millions of recipients without spending too much time and money. In this case, even a small percentage of recipients who read the ad is enough to influence the audience. However, over time, organizations began installing anti-spam systems on their e-mail servers to prevent spamming. As a result, e-mail advertising is now mainly sent to established customers.

2.6 Product placement in videos

With the growing popularity of various blogging and video platforms on the internet, advertising has begun to move there as well. Advertising companies usually pay content producers to include their product in the content or to review it favorably. This type of online advertising is known as *product placement*.

3 Ad blocking techniques

The rapid increase in internet ads has led to complaints from some users, resulting in the development of ad-blocking programs. Many of these programs are open source, but they differ in their methods for blocking ads. Since there is no universal method to block all ads, each of these approaches can be useful in specific situations. The following sections examine these methods and give examples of programs that use them.

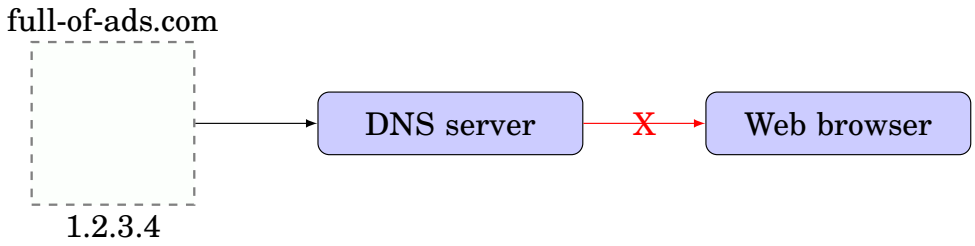


Figure 1. IP filtering

3.1 IP and DNS filtering

IP and DNS filtering is one of the earliest forms of blocking online advertisements. It involves refusing connections to addresses that distribute ads. If a certain IP address or domain is known to distribute advertisements, attempts to access these resources on the user's device can be blocked. Special lists are typically created to include addresses and domain names that are prohibited from access. Figure 1 illustrates the IP and DNS filtering algorithm. IP filtering is often implemented using a firewall in practice. The simplest example of using DNS filtering approach is the file `/etc/hosts` on Unix-like systems. User can add pairs of host names and IP addresses to this file, and later this data will be used for resolving purposes. The project *Pi-hole*[1] is a more advanced solution. In addition to DNS filtering, it can display visually appealing dashboards that provide information on the number of blocked ads. However, this methods described above are no longer effective as many marketing companies have started to dynamically change IP addresses and domains names of their servers distributing ads.

3.2 Content filtering

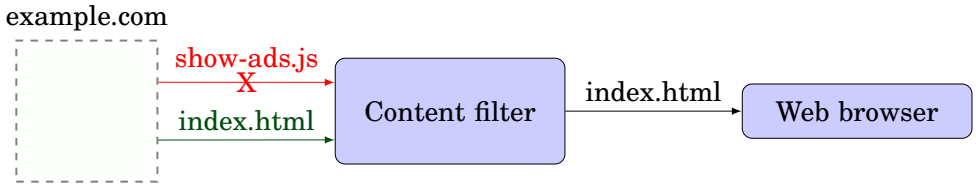


Figure 2. Content filtering

Another method to detect online ads is to find them in HTML code or DOM model of the downloaded web page. In many cases, it is possible to exclude some of HTML tags before rendering. Such an approach is refered as *content filtering*.

[1] `https://pi-hole.net/`

Content filtering programs are often distributed as web browser extensions. This architectural choice was deliberate. At present, the majority of websites use HTTPS protocol, which encrypts the content between the user and the server. However, in order for an ad blocker to perform filtering, it requires access to this data. The web browser is the only entity with the ability to decrypt the traffic, which is why most content filtering ad blockers are forced to operate as web browser extensions.

Figure 2 illustrates the content filtering algorithm. Some of such blockers are *uBlock Origin*[2], *Adblock Plus*[3], *Ghostery*[4], and *Privacy Badger*[5]. All of these are distributed as web browser extensions. The effectiveness of these blockers has been tested multiple times in practice. Specifically, studies [3] and [8] measured the percentage of blocked ad requests out of all the requests. For example, Garimella et al. [3] reported a range of 25-34%, while Pujol et al. [8] reported 18%. These figures indicate that approximately one third of the user's requests are advertisements. Although the figures differ significantly, the effectiveness of content filtering approach is undeniable.

3.3 Embedded video ads filtering

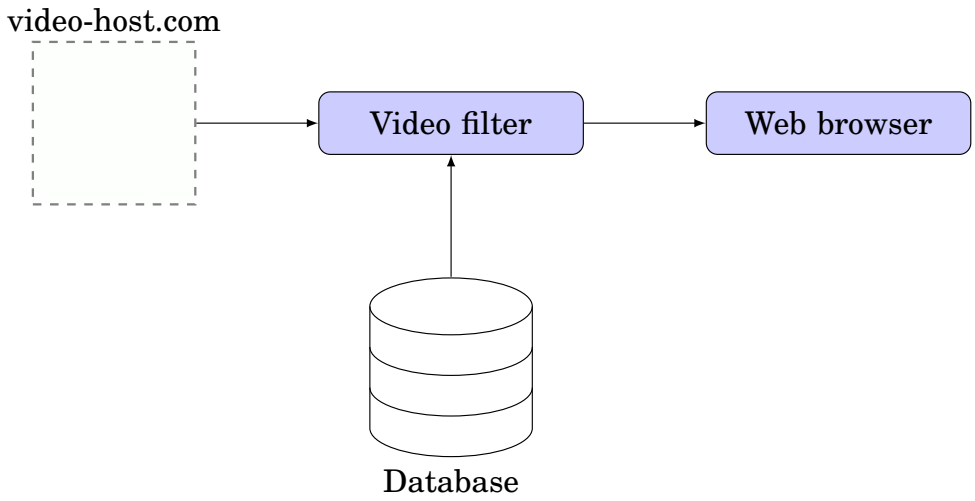


Figure 3. Embedded video ads filtering

In the case of ads embedded into video streams, the approaches described above do not work for obvious reasons. Therefore, the developers have proposed completely different methods of filtering ads from video

[2] https://ublockorigin.com/
[3] https://adblockplus.org/
[4] https://www.ghostery.com/
[5] https://privacybadger.org/

streams. The main idea is based on the use of crowdsourcing technology. This allows users who have already viewed a video and encountered an advertisement to share information with other users, enabling them to skip the ad segment of the video. The accuracy of information regarding ads must be verified to eliminate the possibility of false segments. To address this issue, other users vote for potential ad segments if they are deemed correct. The architecture of video filtering technique is represented on Figure 3. This algorithm was implemented in the program *SponsorBlock*[6], which allows to skip ads integrations in videos on YouTube. This is a new direction in the field of ad blockers that is currently under active development.

4 Counter-measures against ad blockers

As ad blockers have become more common among regular internet users, marketing companies are increasingly interested in detecting their usage. The goal for marketers is to eliminate ad blockers usage because they lose income. This section lists the most common methods for detecting and bypassing ad blockers.

4.1 Ad requests detection

One approach to detect an ad blocker is by checking access to resources that contain advertisements. This is typically done with JavaScript code in the user's browser. If the browser fails to load a request to a third-party resource that contains advertisement, it is probable that the user has an ad blocker installed. In such cases, the website may suggest that the user disable their ad blocker or block access to its content.

4.2 Bugs in implementation of web browsers

Many modern browsers are open source. Therefore, it is possible to search for bugs in the implementation of various web browser modules that will circumvent the restrictions of ad blockers. This approach was demonstrated in a paper by Bashir et al. [2]. Researchers investigated that a number of advertising and analytics companies used a bug in implementation of Chrome's API to bypass content filtering ad blockers for several years. This bug prevented web extensions from being able to filter Web-

[6] `https://sponsor.ajay.app/`

Socket connections. *WebSocket* is a connection protocol built on top of TCP which was designed to improve the performance of web applications. This protocol was standardized in 2011 by RFC 6455.

In May 2012 Chromium users reported a number of issues with the fact that Chrome's API does not allow to filter WebSocket connections. Thus web extensions were unable to block them. Then in late 2014 users reported about unblockable ads which they found in Chrome browser. Users investigated that this ad was distributing via WebSocket protocol. And finally in April 2017 the bug was fixedin a new version of Chrome.

Based on these reports Bashir et al. [2] found out that during this period of time when the bug was present a number of analytics companies used it to bypass ad blockers and show ads to users.

4.3 RAD domains

Lin et al. [5] described one more approach which some marketing companies use to be not detected by ad blockers. The idea is to use several domains which distribute the same advertisement. Researchers call this method as *RAD (replica ad domains)*. They demonstrated that domain replicas can remain unblocked for an average of 410.5 days. Often replicas have the same owner which can be found in DNS or TLS records. At the same time ad blockers can not instantly add a lot of domains to its lists since there is a probability of false-positive filtering. That is why RAD domains are very effective against ad blockers.

5 Ad blockers impact on users' privacy

Privacy is one of the main reasons why users choose to use ad blockers. Marketing companies are acquiring large massives of personal data for targeting promotions. Ad blockers are trying to block such requests as well as ad requests. Accorings to [3] uBlock Origin blocks about 100% of tracking requests. But at the same time ab blockers can negatively affect on user's privacy.

5.1 Acceptable ads

Some ad blockers have started creating lists of so-called 'acceptable ads' in order to monetize. Large companies pay ad blocker developers to be included in these lists, making their ads acceptable and preventing them

from being blocked on users' devices. Such an approach was implemented in Adblock Plus. This solution may have a number of bad consequences for a user especially in terms of privacy. Consider if some domain tracks users who loads it and if the company can pay for it to become in acceptable list then this tracker can easily bypass ad blocker. Zafar et al. [9] investigated that acceptable ads from Adblock Plus also includes domains with 1x1 pixel images which are used to track users and by default blocked.

5.2 Cookies are still avaliable

In a recent paper, Jacobsson et al. [4] demonstrated in practice that ad blockers have little or no effect on the number of cookies users have, while cookies have been and continue to be one of the easiest and most effective ways to track users' movements between different sites. Analytics companies collect this information, which they then sell to marketing companies. In this way, cookies can accurately identify visitors to websites, which undoubtedly has a negative impact on end-user security.

Ad blockers cannot afford to arbitrarily block the setting of cookies, as in many situations such blocking can negatively impact the integrity of a website. For example, cookie technology is the entire basis for user authentication and authorization on virtually every modern website. Therefore, there is currently no easy solution to this problem.

5.3 Browser fingerprinting

When a user accesses a web page, they leave behind a digital trace that includes various parameters, such as the browser and operating system version, time zone, browser extensions, and screen resolution. This combination of parameters is known as a *browser fingerprint*, which can be used to identify the user with a high degree of accuracy. The probability of two different users having the same browser fingerprint decreases as more parameters are involved in its formation. Marketing companies often exploit this feature to advertise their products online, which can compromise user privacy. A study by Gunes et al. [1] have documented cases where marketing companies have used the unique characteristics of font display in browsers to track users.

An ad blocker is typically a browser extension. Its presence in a user's list of extensions, along with its version, can potentially strengthen the browser fingerprint, leading to more accurate user identification on web-

sites.

6 Conclusion

This article reviewed the different ways of distributing advertisements on the internet, as well as the main programmatic methods of ad blocking. Some approaches used by marketing companies to bypass ad blockers have also been listed. As a result, it can be seen that ad blockers are not a universal method of countering advertising and marketers are constantly finding newer ways to deliver ads to users. Therefore, continuous improvement of ad blocking methods is required. The paper also touches on the impact of ad blockers on user privacy. The popular belief that ad blockers help to maintain user privacy is largely untrue. The article cites several possible scenarios in which the use of ad blockers does not improve user privacy or even negatively affects it.

References

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 674689, New York, NY, USA, 2014. Association for Computing Machinery. https://doi.org/10.1145/2660267.2660347.

[2] Muhammad Ahmad Bashir, Sajjad Arshad, Engin Kirda, William Robertson, and Christo Wilson. How tracking companies circumvented ad blockers using websockets. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, page 471477, New York, NY, USA, 2018. Association for Computing Machinery. https://doi.org/10.1145/3278532.3278573.

[3] Kiran Garimella, Orestis Kostakis, and Michael Mathioudakis. Ad-blocking: A study on performance, privacy and counter-measures. 05 2017. https://doi.org/10.1145/3091478.3091514.

[4] Philip Gunnarsson, Adam Jakobsson, and Niklas Carlsson. On the impact of internal webpage selection when evaluating ad blocker performance. In *2022 30th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 41–48, 2022. https://doi.org/10.1109/MASCOTS56607.2022.00014.

[5] Su-Chin Lin, Kai-Hsiang Chou, Yen Chen, Hsu-Chun Hsiao, Darion Cassel, Lujo Bauer, and Limin Jia. Investigating advertisers domain-changing behaviors and their impacts on ad-blocker filter lists. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 576587, New York, NY, USA, 2022. Association for Computing Machinery. https://doi.org/10.1145/3485447.3512218.

[6] Montag, Baszkiewicz, and Sariyska et al. Smartphone usage in the 21st century: who is active on whatsapp? *BMC Research Notes*, 2015. https://doi.org/10.1186/s13104-015-1280-z.

[7] Fiona Nah. A study on tolerable waiting time: How long are web users willing to wait? volume 23, page 285, 01 2003. https://doi.org/10.1080/01449290410001669914.

[8] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed users: Ads and ad-block usage in the wild. *Proceedings of the 2015 Internet Measurement Conference*, 2015. https://doi.org/10.1145/2815675.2815705.

[9] Ahsan Zafar, Aafaq Sabir, Dilawer Ahmed, and Anupam Das. Understanding the privacy implications of adblock plus's acceptable ads. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '21, page 644657, New York, NY, USA, 2021. Association for Computing Machinery. https://doi.org/10.1145/3433210.3437536.

How online games protect and violate player privacy

Atte Haarakangas
atte.haarakangas@aalto.fi

Tutor: Tuomas Aura

Abstract

To be added.

KEYWORDS: To be added.

1 Author's note

This paper is incomplete. It is missing parts. Some parts are early drafts. Some parts have not been proofread. Some parts have not been seen by the tutor. However, all parts that are here have been fact-checked and are reliable. This paper contains correct and factual information, it mainly lacks polish. This is due to the research and writing process taking a longer time than expected.

2 Introduction

Online games are a form of interactive media where an online connection is utilized to provide a service to a user. Online connection is often used by video games in order to communicate with other online users. This connection and things adjacent to it pose unique threats to user privacy.

Threats to privacy within the online gaming space come in many forms.

Some are related to the technology being used for networking. Some are related to the networking features provided by the online game or its service provider. Some threats are by design, either by compromise or to enable a goal of the service provider. This paper will go through some of the most common ones, as an overview of the privacy-related problems that developers and users face when engaging in online gaming.

The focus of this paper is mainly on privacy for online games on PC and console. For scope considerations, this paper does not cover privacy aspects that are exclusive to smartphone gaming or social sciences.

3 Background

To be added.

4 Types of connections

This section explains five of the most common types of connections used for online multiplayer: dedicated servers, peer-to-peer connections, hybrid models, cloud-based solutions and play-by-e-mail. The connections are called networking models. Each networking model has strengths and weaknesses when it comes to security and privacy.

4.1 Dedicated servers

In the dedicated server networking model, one participant is designated as the server and others are designated as clients. A defining feature of dedicated servers is that the server assumes the role of central authority that manages the game for all clients. An individual participant's game functionality is split between server and client. How much of the game is handled server-side or client-side depends on the game. In a typical scenario, a player inputs data that is transmitted to the dedicated server and processed by the server, after which the server can transmit an updated status to all clients.

Servers are expected to store and transmit player data through secure means. Players need to trust the server to handle their information responsibly and to keep it safe from unauthorized access. This is especially important for servers which hold sensitive data. Sensitive data is a broad term for anything that may compromise a player's privacy, such

as e-mail addresses, payment information, private communications and geolocation.

4.2 Peer-to-peer

In peer-to-peer (P2P) networking model, all participants are designated as both clients and servers. Depending on the game, one participant may be designated as a host server, which other clients rely on for handling the game status. While dedicated servers have one centralized server, P2P participants connect to each other, forming a decentralized network. P2P is generally suitable for games that do not require dedicated servers to function. However, connecting directly to other players requires sending data directly to other participants without a central authority to control the transmissions. With this, P2P networking poses unique problems when it comes to security and privacy.

4.3 Hybrid models

Hybrid networking models combine P2P networking with dedicated servers, using both at the same time for effective data processing. For example, a dedicated server could handle critical game functions that are important for all clients, while clients may also communicate with each other for less important tasks. This way, dedicated servers can avoid using resources for tasks that do not require dedicated servers. This approach is challenging, because it requires designing game systems to work with multiple kinds of connections simultaneously.

4.4 Cloud

Cloud technology is useful for games requiring dynamic and scalable server resources. They allow for reduced server costs compared to dedicated servers, but require games to be designed in ways that use cloud resources effectively. Not always having access to the same amount of computational power is a challenge that must be accounted for when developing cloud games.

Cloud networking models typically use services of cloud platforms, which introduces a third party that needs to be trusted with data. Additionally, cloud services distributing traffic means that data may be shared between multiple servers at the same time, further complicating secure data transmission and synchronization.

4.5 Play-by-e-mail

A rather archaic but effective form of multiplayer networking in which communication is done via e-mail is still present in turn-based games. These are called play-by-e-mail (PBEM) games. A recent example of a PBEM videogame is Sid Meier's Civilization VI (2016). Much like playing postal chess by sending turns in mail, players in Civilization VI have the option to send their game file to other players via e-mail. The next player would receive the file, play their turn and send the file to the next player in line. This type of networking is special in the sense that it has no weaknesses typical to other networking models. It is as secure as the method chosen for file transmission, assuming players do not find ways to make the file itself insecure. It is also a useful alternative in cases where other networking solutions are insufficient, for example, with strict networking policies, firewalls or data transmission limits.

5 Netcode considerations

Netcode is what determines what data is transmitted during online play and how it is processed. It determines how inputs are handled and how the game state is updated. Ensuring the privacy of users leads to a set of general considerations for netcode. It needs to have secure, encrypted data transmission to avoid unauthorized access. It needs user authentication and access controls to ensure that correct individuals are accessing user data. It needs to be specific about the data it collects from users to meet bandwidth and processing requirements, and to avoid transmitting user data that is unnecessary for the gameplay experience. It needs to be clear to users about how their data is being used for both trust and legal reasons. And finally, it needs to be kept up-to-date from potential vulnerabilities that could compromise privacy.

For example, Raft (2022) had a networking-related remote code execution (RCE) vulnerability that was patched 6 months after the game's release [2].

6 IP

IP addresses are used for data transmission between players. They are especially important in peer-to-peer networking, where players connect

directly to each other. IP addresses are considered sensitive data that should not be available for other players to see, because they can be used for tracking, geolocation and network attacks. Sharing another person's IP address is therefore considered to be an act of doxxing, which is a word for the act of publishing private information about a person without their permission.

A common problem in P2P games is that each player needs to know the IP addresses of other players, which exposes the IP addresses of all players to each other. Tools such as NetLimiter can be used to view all connections used by a game, which commonly includes IP addresses of other players. Players could take measures to hide their IPs themselves, for example, with the use of a proxy server or VPN. I personally believe this to be an unrealistic expectation for most players.

A network could make use of dynamic IP assignment that periodically changes the IP address of players, making it more difficult to track players based on their IPs. For example, an internet service provider (ISP) may dynamically assign new IP addresses to customers, based on network usage. It provides a layer of security that developers have no control over, and should take into account.

Developers could hide IP addresses with the use of dedicated proxy relays, which are much like dedicated servers, except all they do is relay communication between players. They act as intermediaries for IP discovery and data transmission. This way, the IP addresses of other players are only known by the relays. The relay maps real IP addresses to IP addresses used for connecting to the relay. Players communicate to the relay using the new IP addresses, and the relay forwards the communication to the correct player. This way, P2P networking works as usual, and IP addresses are hidden at minimal server upkeep cost to the developers. However, the relay server needs to stay online, which creates a potential point of failure in the system. An alternative approach to achieve the same outcome would be to make use of services designed for online game networking.

7 Services

Services are a core part of modern online gaming. They include storefronts, tools and external networking services. Examples of storefronts are Steam, Good Old Games (GOG) and PlayStation Network (PSN). Ex-

amples of tools are Steamworks, Battle.net and Xbox Live. Examples of external networking services are Discord, Twitch and TeamSpeak. Each of them add functionality to multiplayer games, with their own considerations for privacy.

7.1 Integrations

Storefronts are defined by their ability to provide players with access to games, but some of them come with tools and services to be used with the games that are on the storefronts. For example, Steamworks is designed to enable developers to integrate their games with Steam. Users on Steam may play multiplayer games using their Steam user account without a need to create a separate account for games that take advantage of Steamworks. This means that players can use the same account for multiple games on Steam, which makes it easy for players to follow each other from one game to another. This also creates a privacy concern. Players can see other players' Steam accounts and all information that the users choose to be public.

7.2 User accounts

A Steam account's public user profile can reveal things about the user. Steam allows its users to control what information is public, which is good for privacy. Visible information includes a user's game library, playtime per game, achievements, games wishlist, list of friends and Steam groups, Steam store and community activity, account age, recent activity and purchases, Steam Workshop submissions, Steam forum posts, country of residence, profile decorations, profile comments from other users and anything the user expresses in text on their profile. This information can be used to deduce a lot of things about what a user has done, is currently doing and possibly intends to do in the future. It is the user's responsibility to make sure that they are not sharing things that they do not want to share. [1]

[1] In December 2023, Steam added a feature that allows players to mark individual games as private, which hides everything about the games from others [23]. This was in addition to its existing privacy features.

7.3 Linking and social media

Not all games are accessible from the same storefront. Different publishers, such as Electronic Arts (EA) and Ubisoft, have their own storefronts. To avoid making new user accounts for each storefront, it is possible to link user accounts from one storefront with another. Linking accounts means sharing information about a user account to a third party for cross-platform features. For example, it is possible to link a Steam account with a Battle.net account in order to synchronize game progress in Overwatch 2 (2022) on Battle.net with the same game on Steam. Similarly, linking an EGS account to a PSN account enables players to use the same save files on both PCs and PlayStation systems, keeping their in-game progress regardless of the choice of platform. Account linking may also allow logging in to one service using credentials from another service provider. It is similar to using a Google account to login to other websites, but with added functionality to online games.

Linking accounts is incentivised with features beyond social networking. Twitch is a livestreaming and social media platform. Linking a Steam account to a Twitch account allows Twitch users to gain rewards for games on Steam [26].

Discord is a social media platform that is popular in gaming communities. It enables players to discuss, join and spectate games online. By default, if Discord detects that a game is running on a supported platform, such as Steam or EGS, it will update the user's status to display what game the user is currently playing. Depending on the game, platform or linked accounts, this may be accompanied by additional information or options. For example, when playing League of Legends (2009), the player's chosen character, game mode, current status and match timer are shown. Games such as Divinity: Original Sin 2 (2017) allow players to join games of other players from Discord, even when the game is played on another platform such as Steam.

Linking a Steam account to a Discord account increases the amount of information visible to other users on Discord, such as the user's Steam profile. Discord also has an option to show others what services a user has linked to their Discord. If a user is not careful with how they link services, it becomes possible for other users to see what they are doing in great detail on multiple platforms at the same time, including non-gaming platforms such as Spotify [22].

7.4 Service requirements

In order to use a service or play a game on a service, the user may be required to give the service information about themselves. For example, Battle.net requires players to verify their identity by giving them their phone number, real name and payment details before letting them play Call of Duty: Modern Warfare 2 (2022). If a user purchased the game on Steam with linking to Battle.net, they don't need to give their name of payment details to Battle.net. The reasoning for this is to deter cheaters at the expense of users' privacy, meaning that users need to trust Battle.net to protect their phone numbers from unauthorized access. [24]

7.5 Payment options

Some users may be unable or unwilling to provide private payment details such as credit card information, name and address. For this reason, there are often multiple possible privacy-oriented payment options available.

Many brick-and-mortar stores sell pre-paid cards for select games, services and platforms. For example, Fortnite V-Bucks cards are sold in stores and are redeemable through the PIN code contained within the purchase [5]. V-Bucks are a digital currency usable within Fortnite (2017).

Some games feature the option to pay by phone call or SMS. For example, JaGEx, the developers of RuneScape (1998), used to allow players to pay by calling a number to add the cost to their phone bill [10]. In certain countries, it is still possible to pay by sending an SMS message. Through these methods, the buyer would only give their phone number to the payment service provider.

An old method of paying for games privately was sending cash in traditional mail. For example, Simutronics used to have this as a payment option for GemStone IV (1988). For modern gaming, traditional mail may be too slow and impractical. Simutronics also used to provide the option to pay by arranging an electronic check over a phone call. [21]

A recent trend in privacy-oriented payment options has been cryptocurrency. Some games offer it as an alternative to credit cards, and some only accept cryptocurrency. [31]

8 User aliases

In online multiplayer, players generally protect their privacy by adopting a username that is different from their real name. This username is also known as a nickname, or a gamer tag. It is the name that is shown to other players. Players may recognize each other through their nicknames across games and platforms.

8.1 Changing nicknames

Users may have the option to protect their privacy by changing their usernames or nicknames. Changing nicknames makes it more difficult for third-party attackers to monitor individual players. Different services and games have different solutions for changing nicknames.

Some services don't allow it at all, and require creating a new account to use a different name. An example of this was PSN before 2019 [20].

Some services allow changing a nickname once for free, and charge money to change it a second time. An example of this is Xbox Live, which charges $9.99 USD every time the nickname is changed after the first time.

Some services charge money for every nickname change. An example of this is World of Warcraft (WoW) (2004) on Battle.net, which charges $10.00 USD for every time the user wants to change their nickname.

Some services allow users to change their nicknames for free, but only once within a specific amount of time. For example, Twitch lets users change their username once every 60 days.

Some services allow users to change their nicknames for free as often as they like. For example, Steam lets users change their nickname as often as they want with no extra cost to the user. There is practically no limit.

8.2 Levels of usernames

One of the reasons why a username change is easier with some services is the level at which a username is applied. For example, on Steam, the user account name cannot be changed, but is not visible to anyone. It is a unique account identifier and cannot be changed by design. The name that is visible to other users is the nickname, which can be changed anytime.

Some games feature a privacy nickname option to use a game-specific nickname that is separate from the normal nickname. For example, Tom Clancy's Rainbow Six: Siege (R6) (2015) has this feature [28]. When the player uses a privacy nickname, that nickname is shown in place of the player's normal nickname, hiding the normal nickname for other players. Due to its competitive nature, players who stream R6 to live audiences sometimes face problems in-game if other players recognize their nickname. Other players could sabotage the streamer's game on purpose. Stream viewers could also attempt to sabotage the game by sending friend invites to the streamer during pivotal moments in-game. Since R6 is an Ubisoft title on Ubisoft Connect -platform, knowing a streamer's normal nickname could also enable others to track them on other platforms through linked services.

A similar approach to privacy nicknames can be found in World of Warcraft, where players play as characters with given names. Character names are separate from usernames, and other players cannot know a player's username from the character name. The difference to privacy nicknames is that character names are unique and cannot be changed freely. This makes it easy to recognize the player in-game but protects their privacy outside the game.

8.3 Pseudonyms, unique usernames and impersonation

Privacy is especially important for streamers and other online figures who typically use pseudonyms to hide their real identity. Without pseudonyms, they may find trouble in their life outside of streaming. A real name can help attackers discover information about their target. An extreme example of this is swatting, which is the act of deceiving emergency services into sending a SWAT team to a streamer's home. It is illegal and can lead to death [29].

Pseudonyms are often associated with unique usernames. In theory, unique usernames make it more difficult to deceive others into believing that a person is someone else. For example, there is only one @elonmusk on the social media platform X, previously known as Twitter. However, unique usernames can also work against credibility and make it easier to deceive people. When usernames are unique, a credible looking username is inherently more believable.

As an example of what impersonation with unique usernames can look like, in 2022, a Twitter account named @NintendoOfUS (or @NintenDoofus)

posted a picture of a popular Nintendo character Mario giving the middle finger [3]. The picture was likely posted as critique to Twitter's controversial decision to allow any user to verify themselves through monthly payment to appear more official [14]. This payment added a blue verification badge to an account. As a result, the account was banned for impersonation. It was considered believable enough to damage Nintendo's reputation, even though its violation of Twitter's Parody Account Policy at the time [27] was open to interpretation.

The credibility problem applies to players as well. Let us assume a player whose online identity is defined by a nickname, for example SinisterStriker. If their nickname had been taken by someone else for any reason, they would have to use a different username, such as @sinisterstriker2. The person who has access to the username @sinisterstriker would have the opportunity to impersonate SinisterStriker and make the real SinisterStriker less credible. Being able to effectively impersonate a person enables attackers, for example, to use social engineering to gather private details about their targets.

It is also possible for a user to change their unique nickname. Depending on the platform, this usually frees it for others to claim. This makes it possible for attackers to take advantage of anyone who is unaware of the name change.

Until 2023, Discord had a different approach to unique usernames through the use of discriminators [30]. They were four-digit numbers attached to usernames, allowing 9,999 users to have the same username with different discriminators. These types of usernames were good for privacy, because knowing another user's username was not everything that was needed to find them on the platform. Attackers would have also needed to know their discriminator. In 2023, Discord changed to a more mainstream approach with unique usernames. Like in X, the downside is that there can only be one @sinisterstriker, and it may not be the real SinisterStriker. Nowadays, Discord allows users to use nicknames that can be changed freely. The unique username is only shown when the nickname is inspected, reducing the chance that footage of a Discord conversation would reveal the usernames of the people in the conversation. This is unlike platforms such as X that always show the unique username.

8.4 Account trading

An additional threat to privacy with unique usernames comes with the practice of selling usernames. It is possible to create accounts solely for holding rare or desirable usernames. These accounts can be sold to users looking to use the usernames. Later, the person who originally created the account can use account recovery with proof of account creation to hijack the account. This grants the account seller unrestricted access to everything on the account, including private information.

Account selling is also a problem in competitive multiplayer games with ranking systems. Players are incentivised to purchase accounts with different rankings in order to play with players of specific skill levels. Players with low skill purchase high rank accounts in order to play against more skilled players. Players with high skill purchase low rank accounts in order to play against less skilled players. This is known as smurfing or boosting, depending on intent.

8.5 Real ID

An example of how online games can purposefully work against player privacy is Battle.net Real ID, introduced in July 2010. Finding forums for Blizzard games difficult to moderate, the company began to enforce all Battle.net forum posters and repliers to use their real first and last names [15]. Players expressed concern, but Blizzard developer Micah Whipple set an example and revealed their first and last name. Within hours, the community discovered and posted private details about them, including their age, home address, phone number, family members and pictures [9]. Real ID became optional two months later [16], but it remains a part of Battle.net to this day.

In 2018, a Reddit user reported an exploit where a player could see another player's Real ID without that player's consent [11]. I did not have an opportunity to verify this exploit, so I do not know if it is possible in 2024. Although the information is not verified, it reinforces the notion that even with good intentions, a game company having access to personal and private information for unimportant reasons has the potential to be harmful for privacy.

8.6 Use of player identity

In Tribes 3: Rivals early access release version v0.2.1.7. (2024), matches were partly populated by bots. The bots appeared as if they were real players by using the likeness of real players that were not in the same match. This meant that a bot was practically indistinguishable from a real player apart from the fact that bots did not behave like real players [12]. This was not disclosed to players and was absent from the EULA [19]. It violated the player's right to choose how their identity was used. Additionally, it had the potential to cause players to misattribute actions done by bots as actions done by the real players. For example, a bot that looked like SinisterStiker could have done something counterproductive that caused their team to lose a game, making it possible for the other players to believe that the real SinisterStriker was maliciously acting against them. This is an example of a case where not respecting a player's privacy for seemingly harmless purposes could still have downsides.

A few days after the version went live, the developers stated on the official Tribes 3 Discord that this was not an intended feature. It seemed to have been caused by a bug where a server would fail to recognize that a player had left the game, giving that player's identity to bots until the server was reset. According to developer Ghost: "Prophecy absolutely does not collect player names to use them for bots." [8]

9 On-screen information in games

Online game livestreamers need to be careful not to inadvertently display private information on stream to avoid broadcasting that information to the viewers. Streamers have control over what programs their viewers see, so they can keep things like private chat off stream in most situations. Most of the time, a game streamer shows the game screen.

There are games that show private information on-screen with the assumption that only the player can see the information. Online multiplayer games sometimes show the player's own IP or location for matchmaking purposes, for example, in Age of Wonders II (2002) [25]. In Marvel's Avengers (2020), the IP became visible on-screen by mistake, due to an oversight by the game developer [18].

Some games display the local time in-game, which would reveal the time zone of the streamer. World of Warcraft gives players an option

to show server time on screen instead of local time, which is a privacy-friendly feature that also helps at scheduling events with other players.

Some games make use of private information for dramatic effect. Doki Doki Literature Club! (2017) attempts to break the fourth wall by having a character address the player in-game by their Windows username, which is often the player's real name. To protect the player's privacy, the game switches to different dialogue if it detects that streaming software is running on the PC. The alternative dialogue directly addresses the stream viewers instead of the player.

Pony Island (2016) does something similar. It tries to trick the player by sending them messages in-game as if the messages were sent by their Steam friends outside of the game. It makes it look like the player is getting messages from their Steam friends, but the message contents are actually the game's antagonist trying to get the player's attention. This reveals some of the player's Steam friends on screen.

9.1 Lobby codes

Multiplayer games with lobby codes often display the code on screen for the host to share with their friends or audience. Some games allow the host to hide the code. If an attacker had access to the lobby code, they could join the game and see, for example, Steam profiles of other players connected to the game. It is possible to design games around this problem. For example, Jackbox Games protects their players' privacy by making the host be the only person running the game. Players connect to a game by using a web browser and choose a display name. This avoids giving players information about other players in the game.

9.2 Third-party overlays

For added functionality, some games use third-party overlays that appear on top of the game. For example, Steam allows players to open a Steam overlay in-game that allows players to configure game features and browse the Steam store without exiting the game program. This overlay can display private information. For example, DotA 2 (2013) [2] handles payments through Steam's payment system. Clicking a purchase button in DotA 2's in-game shop opens the Steam overlay to allow the player to

[2]Although DotA 2 is a first-party title on Steam, it uses many of the same Steam overlay features as third-party titles on the platform.

confirm their purchase without exiting the game. This purchase confirmation screen shows the Steam account owner's full name and address. This means that the game functionally has a button that shows the player's private information on screen without warning.

There are other games on Steam, such as Battlerite (2016), which use the overlay for transactions in a different way. Clicking a purchase button in an in-game shop in Battlerite opens the Steam overlay, but unlike DotA 2, does not immediately show private information. Battlerite's purchases require multiple clicks to reach a purchase confirmation screen. This approach makes it less likely for a person to accidentally reveal information.

To avoid privacy problems with third-party overlays, streaming software such as OBS may provide an option to exclude third-party overlays from being shown. However, this does not work in all cases. Also, the inconsistencies in the ways games handle on-screen information during payment confirmation make it good practice to hide the screen from livestreams whenever handling payments.

10 Player profiling

Player profiling is the practice of gathering and analyzing information about players in order to understand them more. It is useful for predicting how the player might act in future gameplay. A common example of player profiling is adaptive difficulty. If a game notices that the player is performing below expectations, it can adjust the game difficulty to be easier to achieve an intended experience. For example, in Alan Wake II (2023), if a player is spending more resources than expected, the game will increase the amount of resources available in lootable containers. [3]

Player profiling can be done through any data points generated by the player. Online games often use player profiling to generate a unique player profile for each player. This allows games to create a unique experiences for each player. It also enables games to use the information to improve player retention and monetization strategies. For example, if a player preferred to use certain types of items, the game could place those items on a limited-time sale at a time when the player is usually online. In

[3] As a game design choice that affects the gameplay experience, it is not always suitable to use player profiling to adjust gameplay. Some games, such as Dark Souls (2011), avoid using adaptive difficulty in order to present the same challenge to all players.

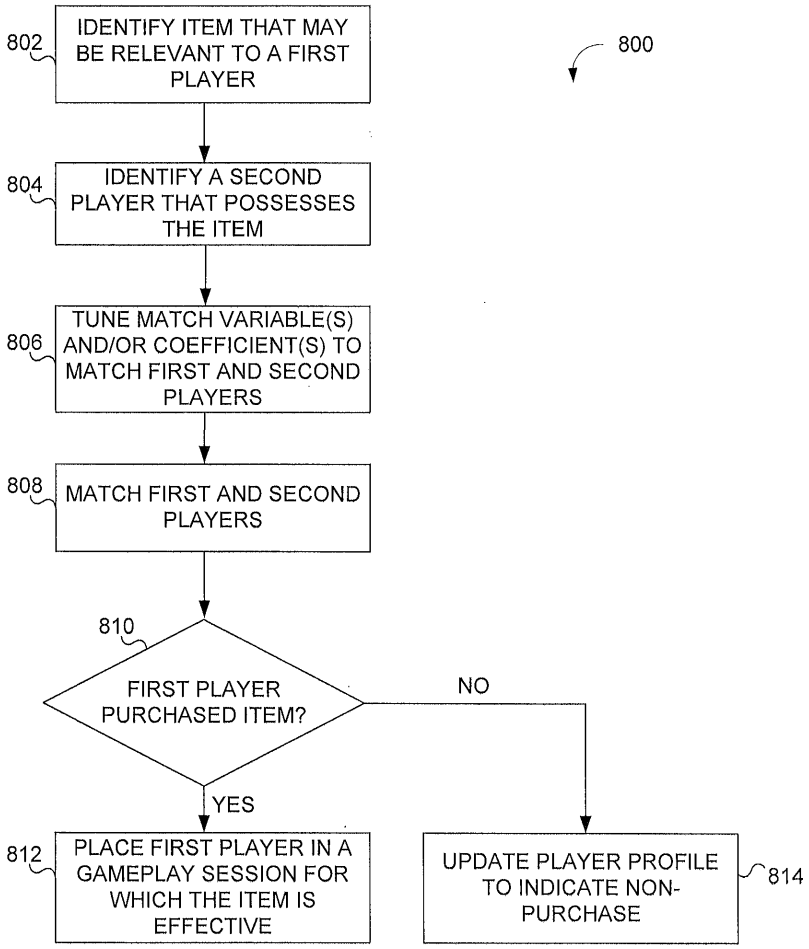


FIG. 8

Figure 1. A 2015 patent [1] by Activision demonstrates how player profiling can be used to drive microtransactions in multiplayer games.

practice, this is similar to targeted advertising but in the context of game systems. Due to how game systems are often tied to player profiling, it is usually not an option for players to decline this type of data collection. This privacy issue is exacerbated by always-online games, which require the game to be played online. [4]

Player profiling is likely going to be used more in future games. A 2022 patent for a Persona Driven Dynamic Content Framework (PDDCF) [4] describes a profiling system where a psychological profile of the player across multiple games and platforms is generated through gameplay. This "player persona" is then used to deliver personalized game content and

[4]Certain always-online games require an internet connection regardless of whether or not it has gameplay features which would require an online connection.

product recommendations. This is a privacy risk that could have serious consequences, should the player's information be revealed or used for anything outside of the games. With modern games' player profiling capabilities, it becomes increasingly important to consider players' privacy.

11 Anti-cheat

Some form of anti-cheat is used in most multiplayer games. Its purpose is to ensure a fair gameplay experience by reducing the amount of cheaters.

Anti-cheat comes in many forms. [5] Many modern anti-cheat software solutions function by monitoring activity on the computer in order to detect cheat software. Due to the nature of cheating in games, anti-cheat is commonly closed-source software. Additionally, many popular anti-cheat solutions operate at the kernel level of the operating system, which is the highest privilege level [17]. This type of anti-cheat is known as ring 0 anti-cheat. [6]

Ring 0 anti-cheat has direct access to system resources. In simple terms, it has absolute authority over the system. It can do anything without the user's knowledge and it can send any information or data on the user's computer to any third parties. In general, even on lower privilege levels, anti-cheat that is based on monitoring and reporting can be classified as spyware [6, 13] with the potential to be used for mass surveillance. Ring 0 anti-cheat is especially risky from a security and privacy perspective.

The security risk of ring 0 anti-cheat has multiple angles of exploitation. In order to be viable, it requires trust, compatible conditions and updates. The game has to come from a trusted developer and publisher. The anti-cheat chosen for the game has to come from a trusted anti-cheat developer and provider. The anti-cheat has to run on a trusted, uncompromised computer. The anti-cheat has to run on compatible hardware and software, which is why Linux users and virtual machine users often have trouble with games that use ring 0 anti-cheat. The anti-cheat software itself has to be uncompromised and resilient to backdoor access and other exploits. It also needs to be resilient to user errors and avoid confusing users with other users. Finally, it needs to be kept up to date from all

[5] For brevity, the focus is on the most severe privacy risks of anti-cheat.

[6] The term ring 0 comes from operating system protection rings designed to control access to system resources.

relevant threats.

With all its compromises to privacy and security, ring 0 anti-cheat is still not able to completely eliminate cheating [7]. It is ineffective against new and unknown cheats, and hardware-based cheating.

Ring 0 anti-cheat measures also incentivize cheaters to respond by installing cheats that run on ring 0. This escalates the situation and gives more third parties access to the players' computers.

The consensus among game developers in the industry appears to be that developing high-quality anti-cheat is a difficult and expensive task that is generally not worth the resources required for it. Most developers and publishers would rather deploy a third-party ring 0 anti-cheat solution made by anti-cheat professionals. The rationale is that cheating is seen as far worse for a game than the potential risk to players' privacy and security. [7]

The most effective way for players to stay safe from threats posed by ring 0 anti-cheat software is to play games that use it on a dedicated computer that is separate from sensitive data.

12 Conclusion

To be added.

References

[1] Activision Publishing Inc. System and method for driving microtransactions in multiplayer video games, 2015. URL https://patents.google.com/patent/US20160005270A1/en.

[2] Thomas Bouzerar. Remote code execution in raft survival game (cve-2022-47530). Technical report, Synacktiv, 2022. URL https://www.synacktiv.com/sites/default/files/2022-12/Raft_RCE.pdf.

[3] Giovanni Colantonio. Gaming companies become impersonation targets as twitter verification opens to all. *Digital Trends*, 2022. URL https://www.digitaltrends.com/gaming/twitter-blue-verification-nintendo-gaming/.

[4] Electronic Arts Inc. Persona driven dynamic content framework, 2022. URL https://patentscope.wipo.int/search/en/detail.jsf?docId=US367933587.

[7] This is based on uncited interviews and public posts of various game developers responding to concerns regarding ring 0 anti-cheat.

[5] Epic Games. How to redeem a v-bucks card. URL https://www.epicgames.com/help/en-US/c-Category_Fortnite/c-Fortnite_BillingSupport/how-to-redeem-a-v-bucks-card-a000084845.

[6] Jon Espenschied. No security reprieve from blizzard's warden. *Computerworld*, 2007. URL https://www.computerworld.com/article/1641355/no-security-reprieve-from-blizzard-s-warden.html.

[7] Lorenzo Franceschi-Bicchierai. Esports league postponed after players hacked midgame. *TechCrunch*, 2024. URL https://techcrunch.com/2024/03/18/esports-league-postponed-after-players-hacked-midgame/.

[8] Ghost. Re: Using player names for bots, 2024.

[9] Godmode. Real names on the official forums [new real id function], 2010. URL https://web.archive.org/web/20100711143455/http://wowriot.gameriot.com/blogs/Americans-are-bad-at-games/Real-Names-on-the-Official-Forums-New-REAL-ID-Function.

[10] JaGEx. New payment option - us paybysms/text, 2006. URL https://secure.runescape.com/m=news/new-payment-option---us-paybysmstext.

[11] KaraliKing. Real id privacy issue, 2018. URL https://old.reddit.com/r/Blizzard/comments/9awp29/real_id_privacy_issue/.

[12] Kenxai and Gavrok. hello gavrok, 2024. URL https://clips.twitch.tv/CloudyEnticingPanPipeHype-QtIBzg4dOsD6pdNu.

[13] Corynne McSherry. A new gaming feature: Spyware. *The Electronic Frontier Foundation*, 2005. URL https://www.eff.org/deeplinks/2005/10/new-gaming-feature-spyware.

[14] Tomás Mier. Elon musk's $8 checkmarks prove to be a twitter misinformation disaster. *Rolling Stone*, 2022. URL https://www.rollingstone.com/culture/culture-news/elon-musk-paid-twitter-checkmarks-parody-shutdown-misinformation-disaster-12346

[15] Nethaera. Battle.net update: Upcoming changes to the forums, 2010. URL https://web.archive.org/web/20100708212320/http://forums.battle.net/thread.html?topicId=25626109041.

[16] Nethaera. Regarding real names in forums, 2010. URL https://web.archive.org/web/20100712104703/http://forums.worldofwarcraft.com/thread.html?topicId=25968987278.

[17] Serif Pilipovic. Every game with kernel–level anti–cheat software. LEVVVEL, 2024. URL https://levvvel.com/games-with-kernel-level-anti-cheat-software/.

[18] @PlayAvengers, 2021. URL https://twitter.com/PlayAvengers/status/1407388018504257541.

[19] Prophecy Games. Tribes 3: Rivals - end user license agreement, 2024. URL https://store.steampowered.com//eula/2687970_eula_0.

[20] Sid Shuman. Online id change on psn: Your questions answered, 2019. URL https://blog.playstation.com/2019/04/10/online-id-change-on-psn-your-questions-answered/.

[21] Simutronics Corp. Account & billing frequently asked questions, 1999. URL https://web.archive.org/web/19990421162807/http://www.play.net/simunet_public/accfaq.asp.

[22] Spotify. Discord and spotify. URL https://support.spotify.com/us/article/discord-and-spotify/.

[23] Steam Support. Steam private games, 2023. URL https://help.steampowered.com/en/faqs/view/1150-C06F-4D62-4966.

[24] Ollie Toms. Modern warfare 2 phone number bug: How to fix the warzone 2 "mobile phone number required" steam issue. *Rock Paper Shotgun*, 2022. URL https://www.rockpapershotgun.com/modern-warfare-2-mobile-phone-number-required-fix.

[25] Triumph Studios and Daidalos. Age of wonders 2 online multiplayer guide. URL https://aow2.heavengames.com/library/online/.

[26] Twitch. How to earn drops. URL https://help.twitch.tv/s/article/mission-based-drops?language=en_US.

[27] Twitter Help Center. Parody, commentary, and fan account policy, 2022. URL https://web.archive.org/web/20221107192852/https://help.twitter.com/en/rules-and-policies/parody-account-policy.

[28] Ubisoft. Streamer mode update. *Ubisoft Dev Blog*, 2021. URL https://www.ubisoft.com/en-gb/game/rainbow-six/siege/news-updates/7Adrc6XOcNCgpKTcymSRUi/streamer-mode-update.

[29] U.S. Attorney's Office, District of Kansas. Ohio gamer pleads guilty in swatting that caused a death, 2019. URL https://www.justice.gov/usao-ks/pr/ohio-gamer-pleads-guilty-swatting-caused-death.

[30] Stanislav Vishnevskiy. Evolving usernames on discord. *Discord Blog*, 2023. URL https://discord.com/blog/usernames.

[31] Sergio Zammit. 15 best crypto games to play in 2024. *Crypto News*, 2024. URL https://cryptonews.com/news/best-crypto-games.htm.

Understanding Unique Aspects of Tailwind CSS in Comparison with Existing CSS Frameworks

Duc Vu Trong
duc.vu@aalto.fi

Tutor: Vepsäläinen Juho

Abstract

Tailwind CSS refers to a new CSS framework (2017) that follows the utility-first CSS paradigm. Recently, this framework has gained popularity and become a reliable option for styling web applications. This paper aims to discover the core concepts of Tailwind by examining the unique aspects of this framework. Tailwind provides a collection of simple utility classes to define the style of web applications without creating separate CSS files. Moreover, this CSS framework uses utility modifiers to support many modern features for styling components, including styling of component states, responsive UI, and dark mode.

This paper also presents the main features of Tailwind by showing the differences between this framework and inline styling, Bootstrap, and component libraries. Although Tailwind requires additional tooling, this CSS framework offers a more consistent approach to styling components and supports modern features that inline styling does not support. While Tailwind helps web applications improve performance and a flexible customization mechanism, Bootstrap and encapsulated components provided by libraries can reduce the time and effort to style elements with their completely styled components.

KEYWORDS: CSS frameworks, Tailwind CSS, utility-first CSS, web styling

1 Introduction

In recent years, the Cascading Style Sheets (CSS) technology has changed dramatically to improve the maintainability of code as well as the developer experience [9]. The term CSS refers to a language for defining a set of rules that determine the rendering of a structured document, whose structure as well as semantics are encoded with document languages, such as HTML [4]. Utility-first CSS refers to a new technique for using CSS that has gained popularity recently due to the flexibility and developer focus of this technique. Utility-first CSS emphasizes web design using compact, single-purpose utility classes. A prominent implementation of this approach is Tailwind CSS [7]. Tailwind CSS provides developers with a set of pre-designed utility classes to help them build modern, responsive, and efficient user interfaces.

Traditional CSS provides a simple method to style a web page; however, using only CSS poses scalability issues because changes in CSS files affect the entire project, which can lead to unexpected errors. Moreover, traditional CSS is difficult to effectively reuse, which constitutes a major problem in coding for developers. Currently, many new styling frameworks have been developed for web applications using different methodologies and approaches. Nonetheless, these styling options cannot adapt to several specific requirements due to their lack of customization ability. For example, alternative CSS frameworks do not enable developers to define a new color and use it for styling elements. Tailwind CSS was created to solve these problems using a utility-first methodology, which is new in comparison to previous styling frameworks.

This paper seeks an answer to the research question: What is Tailwind CSS, and how does it differ from other available CSS frameworks? To evaluate the feasibility of using Tailwind CSS in production, this paper reviews the main philosophy behind Tailwind CSS and the method that Tailwind uses for styling. In addition, this paper draws comparisons between Tailwind and other popular CSS frameworks.

This paper is organized as follows. Section 2 demonstrates the main ideas of the Tailwind framework. Section 3 presents the differences between Tailwind CSS and traditional inline styling. Section 4 describes the comparison of Tailwind and Bootstrap, which is also a popular framework. Section 5 discusses the advantages and disadvantages of Tailwind compared to existing component libraries. Section 6 provides a discussion

of Tailwind CSS. Finally, Section 7 provides summarizing remarks.

2 Tailwind CSS Framework

Tailwind CSS, or simply Tailwind, is a constrained set of utility classes for styling that can be directly used as the class names of HTML elements [9]. The pre-existing classes of Tailwind can comprehensively cover all styling needs, such as color, spacing, transitions, and interactions of users. With Tailwind, developers specify the style for components by applying these pre-defined classes, as shown in Listing 1, instead of traditionally writing CSS [16]. This implementation helps front-end developers reduce the time spent on developing web elements by removing the focus on managing CSS files. For example, Tailwind enables developers to implement a component in the Figure 1 using only the HTML code in the Listing 1 without additional CSS files.

```
<div className="shadow p-4 rounded bg-white flex">
  <img src="profile.png" className="w-16 mr-4" />
  <div className="font-mono">
    <h1 className="font-bold">Seminar in CS</h1>
    <p className="text-base">Tailwind CSS</p>
  </div>
</div>
```

Listing 1. Tailwind CSS Example

2.1 Tailwind constructs necessary CSS based on usage

In addition, Tailwind creates corresponding CSS files based on the class names of this framework in build time [9]. As illustrated in Figure 2, CSS is generated correspondingly to the determined utility classes of Tailwind, which are `font-bold` and `font-mono`. This CSS framework scans the source code for names of classes using regular expressions, which is a pat-

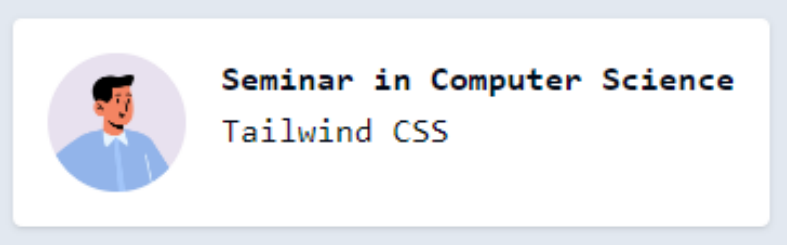


Figure 1. Result of the code in Listing 1

Figure 2. Tailwind parses class names to CSS

tern that the regular expression engine attempts to match in input text, thus extracting all possible strings of class names without considering the programming language of code. Therefore, the detection mechanism of Tailwind CSS is simple and independent of the project framework and language.

2.2 Component states

In a web application, many components have several different states to provide their functionality. For instance, a button component can have two additional states, which are clicking and hovering. The state of clicking illustrates the button when it is clicked by users, and similarly, the state of hovering represents the button that is hovered by a cursor. The style of the components in different states should be easily distinguished to ensure accessibility and make the applications more attractive. To satisfy this demand, Tailwind CSS uses predefined state modifiers, which can be used with every utility class. The modifier is added to the beginning of a utility class to ensure that the style of that class is only visible in the state of the component, which is specified by the modifier. For instance, as shown in Listing 2, a modifier `hover:` is used to assign a new color to the button when it is hovered.

```
1 <div>
2     <button className="bg-blue-500 rounded hover:bg-blue-600">
3         Click Me
4     </button>
5 </div>
```

Listing 2. Tailwind Component State Modifier

Tailwind CSS adds a new class to the element that is only responsible for styling in a specific state, whereas traditional CSS adds to an existing class a new style, which corresponds to a state [7]. In other words, Tailwind CSS separates classes that are used for default styles and styles in

different states. This separation of Tailwind CSS guarantees that a utility class can be applied conditionally, thus helping developers have complete control over the behaviors of components in different states.

The state modifier in Tailwind CSS covers all necessary states of a component. These modifiers, as shown in Table 1, can be divided into four groups: pseudo-classes, pseudo-elements, media and feature queries, and attribute selectors [7].

Tailwind Modifier	**Example**
Pseudo-classes	`hover:`, `active:`, `odd:`, `even:`
Pseudo-elements	`before:`, `after:`, `placeholder:`, `file:`
Media and feature queries	responsive breakpoint, dark mode
Attribute selectors	`[dir="rtl"]`, `[open]`

Table 1. Tailwind supports component state styling [7]

The pseudo-class modifiers focus on the states of the component that correspond to user interaction, such as hovering, focusing, and visiting. In addition, the modifiers of this group also cover states that are based on the relation of a component with others, such as parent components and sibling components, as well as the relative position of a component group, including the first and last element in a list.

The pseudo-element modifiers emphasize detail elements that compose the main components, such as elements that are added before or after a specific type of component, placeholder text for inputs, and markers for a list.

The modifiers in the group of media and feature queries cover responsive aspects and color schemes. These modifiers can be used to achieve the dark mode feature, which has become increasingly popular in web development.

The modifiers of the attribute selector group focus on the states that depend on an attribute of the component. For example, the open modifier can be used to apply style to a dialog component when it is in the opening state.

2.3 Responsive Design

Responsive design refers to an approach that enables a web application to adapt to the different layouts of devices, ranging from large devices, such as monitors and personal computers, to small devices, such as smart-

phones and smartwatches [13]. Currently, responsive design has become a major requirement of modern web applications due to the increasing demand for working on multiple devices [13]. The responsive user interface can be easily built with responsive modifiers provided by Tailwind CSS that are present for media queries in traditional CSS. Responsive utility modifiers of Tailwind CSS can be used with a similar method to other modifiers of Tailwind, which is the addition to the beginning of the utility class. Therefore, these responsive modifiers enable every utility class to be conditionally applied to different sizes of device screens.

Breakpoint Prefix	**Min. Width**	**CSS**
sm	640px	`@media (min-width: 640px) {}`
md	768px	`@media (min-width: 768px) {}`
lg	1024px	`@media (min-width: 1024px) {}`
xl	1280px	`@media (min-width: 1280px) {}`
2xl	1536px	`@media (min-width: 1536px) {}`

Table 2. Tailwind responsive modifiers [7]

By default, Tailwind provides five responsive prefixes, including `sm`, `md`, `lg`, `xl` and `2xl` [7]. As presented in Table 2, each of these modifiers corresponds to a screen width breakpoint that defines a media query CSS using the `min-width` property [7]. Tailwind responsive modifiers use the minimum width of the screen as a breakpoint to support the idea of a mobile-first breakpoint system [7]. To implement this idea, a utility class without a responsive modifier is applied by default, those utility classes accompanied with responsive prefixes can only override the design when the screen has a width larger than the breakpoint of that modifier. For instance, as presented in Listing 3, the utility class `text-base` is only applied when the screen width is greater than 768px, and similarly, class `text-xl` is applied when the screen width is greater than 1280px; otherwise, class `text-sm` is used by default.

```
1 <div>
2     <p className="text-sm md:text-base xl:text-xl">
3         Responsive design
4     </p>
5 </div>
```

Listing 3. Tailwind responsive modifiers

Tailwind responsive modifiers can be used together with the `max` mod-

ifier to improve the flexibility of using responsive breakpoints. The "max" modifier is presented by the keyword `max-` and a Tailwind responsive modifier, for example, keyword `max-lg` indicates the upper limit of breakpoint to which a utility class should be applied [7]. The max modifier is placed between a responsive modifier and a styling utility class, which only affects the screen with a width in the range of the responsive modifier to the `max` modifier. For example, as illustrated in Listing 4, class `md:max-lg:text-base` indicates that the text size is specified by the class `text-base` when the width of the screen is between the `md` and `lg` breakpoints.

```
1 <div>
2     <p className="text-sm md:max-lg:text-base xl:text-xl">
3         Responsive design
4     </p>
5 </div>
```

Listing 4. Tailwind `max` modifier

2.4 Customization

Sections	Description
Content	The paths to files in the project that include the utility class of Tailwind [7].
Theme	Visual design properties, including color, spacing, fonts, sizing, and responsive breakpoints [7].
Plugins	Registration of plugins with Tailwind for introducing new utility classes, styles, and variants [7].
Presets	Setting new configuration for the default configuration of Tailwind [7].
Prefix	Custom prefix that is added to generated utility classes of Tailwind [7].
Important	Configuration for the usage of `!important` with Tailwind utility class [7].
Separator	Customization for the character that should be used to separate modifiers and utility names [7].
Core Plugin	Configuration for disabling classes that Tailwind generates by default that are not necessary for the project [7].

Table 3. Tailwind configuration sections [7]

Tailwind CSS offers the developers extensibility and customization, apart from its set of predefined utility classes. Tailwind CSS facilitates extensive configuration through an optional configuration file at the root folder of a project, whose name is `tailwind.config.js` [7]. Therefore, customization can be achieved by applying several modifications to this configuration file, such as changing default settings, adding new utility classes, and redefining the color palette and other design elements. As shown in Table 3, this configuration file includes eight different sections: content, theme, plugins, presets, prefix, important, separator, and core plugin. Every section of the configuration file is optional and all missing sections are used with the default configuration of Tailwind [7]. Thereby, developers only have to specify necessary sections that require customization. Different customized themes can be created with Tailwind CSS to be used with different projects, which helps projects using Tailwind become unique and easy to identify. For instance, a configuration file of Tailwind is presented in the Listing 5, which defines customized colors, and text font for a project using theme section.

```
module.exports = {
  content: ['./src/**/*.{html,js}'],
  theme: {
    colors: {
      'blue': '#1fb6ff',
      'purple': '#7e5bef',
      'pink': '#ff49db',
      'orange': '#ff7849',
      'green': '#13ce66',
      'yellow': '#ffc82c',
      'gray-dark': '#273444',
      'gray': '#8492a6',
      'gray-light': '#d3dce6',
    },
    fontFamily: {
      sans: ['Graphik', 'sans-serif'],
      serif: ['Merriweather', 'serif'],
    }
  },
}
```

Listing 5. Tailwind configuration file

Tailwind CSS also allows the utilization of arbitrary styling, including arbitrary values, properties, and variants, in order to provide flexibility in crafting precise and pixel-perfect designs [7]. Arbitrary elements are

wrapped in a square bracket notation, which enables Tailwind to read and dynamically generate new CSS classes corresponding to arbitrary values. With this mechanism, completely arbitrary CSS values can be created and used together with interactive modifiers in Tailwind as other utility classes of Tailwind [7].

Moreover, Tailwind also enables developers to add customized style to the CSS file of a project using the directive `@layer` of Tailwind that contains three levels: `base`, `components` and `utilities` [7]. The base layer refers to the default style that is applied to the plain HTML elements [7]. Thereby, this layer has the lowest priority of the three layers. The components layer is used for class-based styles and the utilities layer presents the single-purpose classes that can override any other styles [7]. The allowance for breaking the constraints of Tailwind helps this CSS framework gain adaptability to any specific project requirements. For example, Listing 6 illustrates a declaration of a new utility class using the Tailwind directive `@layer` in the CSS file of a project.

```
1 @layer utilities {
2   .content-auto {
3     content-visibility: auto;
4   }
5 }
```

Listing 6. Tailwind `@layer` directive

3 Comparison between Tailwind CSS and inline styling

Inline styling CSS refers to a method for styling HTML elements that directly address CSS properties in the attribute `style` of an individual element [12]. Since being introduced, Tailwind CSS has raised several uncertainties about the differences between this new framework and inline style CSS due to their similarity in usage. Both approaches directly apply style to elements instead of creating new class names with accompanying styles and assigning these names to the elements. Tailwind and inline styling both focus on HTML elements and eliminate the necessity of CSS files.

3.1 Advantages of Tailwind over inline styling

Inline styling determines the style information for the current HTML element using the attribute `style` [1]. For example, attribute `style` of the element `P` in the Listing 7 defines the color of text and font size properties of a paragraph element [1].

```
1  <P style="font-size: 12pt; color: fuchsia">Style sheets
```

Listing 7. Inline style CSS example [1]

While pre-defined classes of the Tailwind design system provide developers with constraints, inline styling uses arbitrary values for CSS properties. Random values in inline styling make the code obscure, which leads to a lack of maintainability. Inline styling also supports the usage of CSS variables, which also refers to CSS custom properties, to help developers define custom variables for styling values. However, inline styling in the attribute `style` of an element can only specify the style information for that element [1]. Therefore, a CSS variable used in inline styling can not be reused to address the maintainability problem of inline styling. In contrast, Tailwind CSS restricts the selection of developers to a fixed set of options. For example, Tailwind offers a limited set of classes for the text size property, as shown in Figure 3. Therefore, Tailwind can build a consistent UI that is easier to maintain and continuously develop. In addition, Tailwind CSS also supports the use of arbitrary values, which must be placed inside square brackets, to fulfill the specific design requirements without losing the flexibility of inline styling.

Furthermore, Tailwind offers more efficient cache performance for web pages than inline styling. While Tailwind utility classes are converted into corresponding CSS and contained in the CSS file, inline styling is attached to the HTML content [7], [12]. As a result, Tailwind helps web applications cache the style in CSS files instead of caching HTML in inline styling. CSS files require significantly fewer changes than HTML contents; thereby, style can remain more constantly in CSS caching than in HTML caching [3]. Therefore, Tailwind supports web applications to achieve more efficiency in caching style than inline styling.

On the one hand, the inline styling approach cannot meet the requirement of responsive design because media queries cannot be used within the inline style. On the other hand, Tailwind CSS provides responsive utility variants to make applications adaptive [14]. For instance, as presented in Figure 3, Tailwind enables developers to change the size of text

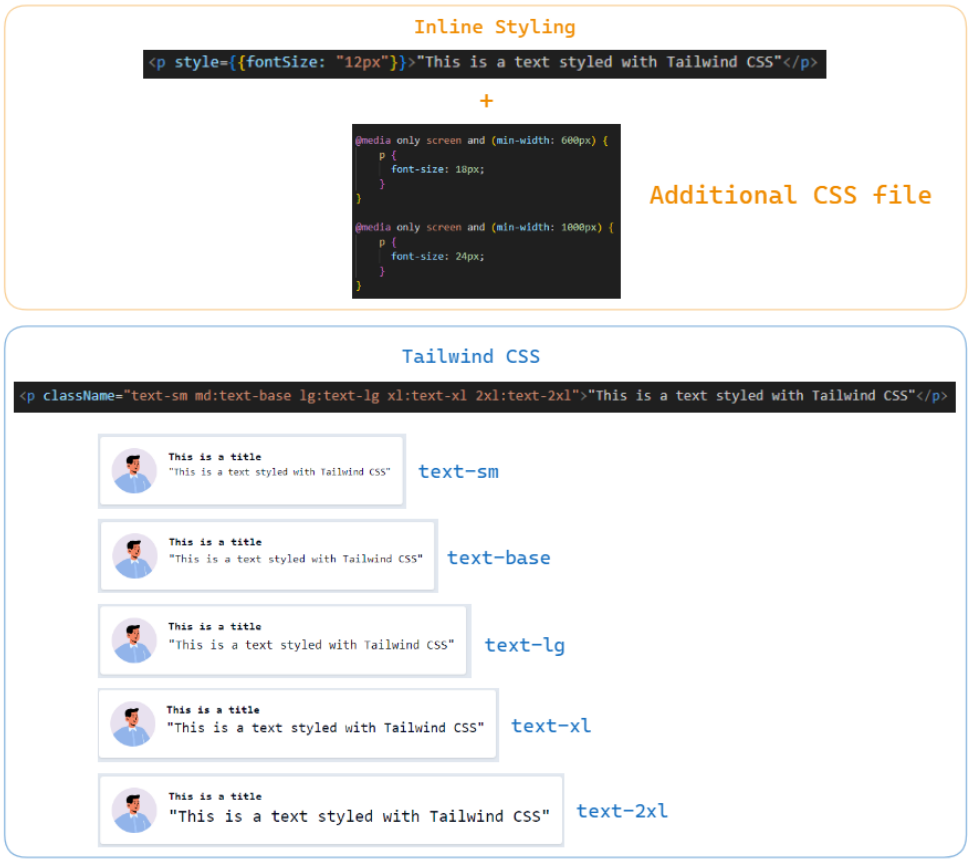


Figure 3. Advantages of Tailwind over inline styling

components to fit different screen sizes, whereas inline styling needs to use other declarations in CSS files. Each of the Tailwind utility variants, which is then transformed into CSS by Tailwind, corresponds to a breakpoint in media queries [7]. Tailwind also has a specific mechanism for using these responsive utilities to target a range of breakpoints in order to increase flexibility. Similarly, Tailwind also supports styling with pseudo-classes that can not be achieved with inline styling.

3.2 Disadvantages of Tailwind in comparison with inline styling

By default, Tailwind CSS is not included in any libraries or frameworks for building UI. Therefore, Tailwind requires several steps of installation to be used in the production, which is similar to other major CSS frameworks. Although Tailwind also can work directly in the browser, using Tailwind in this mode adds JavaScript dependencies to the resulting build. Therefore, Tailwind is recommended to use the build step with installation, especially for production, to achieve optimal performance. Tailwind can be installed using several approaches, including the Tailwind Client Interface (CLI), PostCSS, and Content Delivery Network (CDN) [7].

In addition, the installation of Tailwind depends on the framework that a project uses to develop the UI, because different development frameworks have specific methods for installing Tailwind. In contrast, inline styling can be used without any further setup because inline styling is a basic approach to using CSS inside HTML files that is accompanied by HTML [1].

4 Comparison between Tailwind and Bootstrap

Bootstrap (2011) is a comprehensive CSS framework for front-end development that has gained its reputation for many years [10]. This framework offers developers design templates for many UI components, including typography, forms, buttons, tables, and navigation [10]. Moreover, Bootstrap also facilitates styling components by applying pre-existing classes to HTML elements, which is similar to the approach of Tailwind.

4.1 Advantages of Tailwind over Bootstrap

In comparison with Bootstrap, Tailwind has several advantages. Tailwind CSS can help a web application achieve better performance than Bootstrap from a scalability perspective since the file size of Tailwind is smaller than that of the Bootstrap library. Tailwind CSS only includes the CSS styles that correspond to the utility classes used by developers, therefore decreasing the loading time of pages. These styles are converted from used utility classes and included in the main CSS file of a project. Whereas, the Bootstrap framework not only loads its entire CSS file but also includes Javascript components that considerably increase the size of the file, especially on page initialization. For instance, Figure 4 presents network requests in a web browser for loading two similar web pages, one of which uses Tailwind CSS and the other one uses Bootstrap. The page using Tailwind CSS only needs to load a little CSS rules contained in the file `index.css` which has a size of 14.9 kilobytes. In contrast, the page using Bootstrap has to load complete CSS and JavaScript files of Bootstrap with a total size of 352 kilobytes.

Furthermore, Tailwind CSS offers better customization ability and flexibility compared to Bootstrap. While Tailwind provides separate utility classes that can be used to make a large number of different style combinations, Bootstrap uses built-in themes and templates. Tailwind CSS

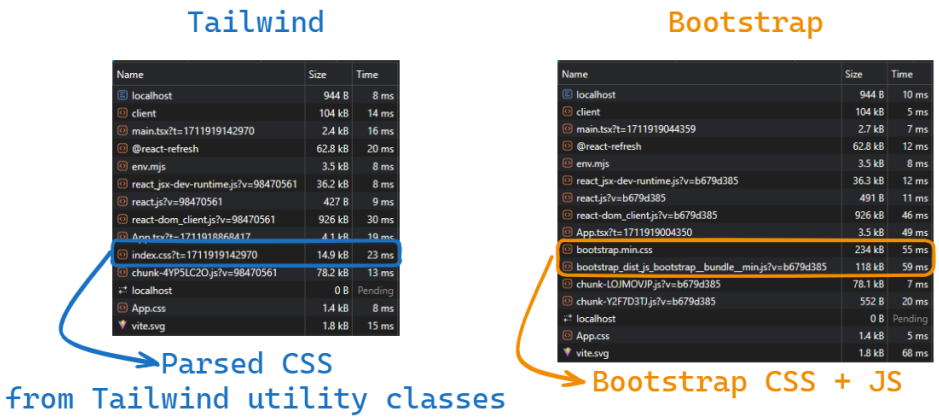


Figure 4. Comparison of file loading between Tailwind and Bootstrap

also allows developers to create their own customized style options based on the utility system of Tailwind. Therefore, using Tailwind CSS can help develop applications with unique UI, which is laborious to achieve with Bootstrap.

4.2 Disadvantages of Tailwind in comparison with Bootstrap

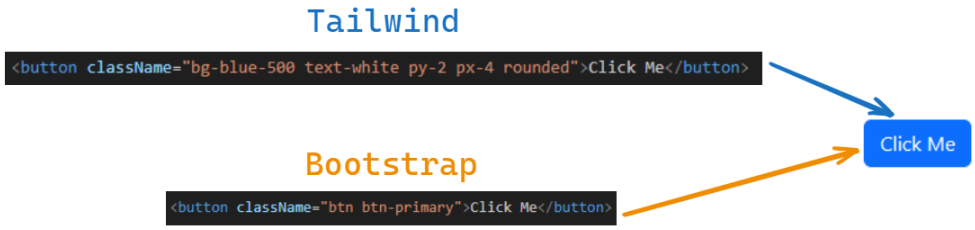


Figure 5. Comparison of coding between Tailwind and Bootstrap

Tailwind CSS still poses several drawbacks compared to Bootstrap. The newer framework is less efficient than Bootstrap from a development performance perspective. Since a class of Bootstrap framework includes multiple built-in CSS rules to specify a complete component that can be used immediately for production, this framework can significantly accelerate the development process. In contrast, a utility class of Tailwind refers to a single CSS property, such as color and size. Therefore, Tailwind CSS requires developers to spend more time choosing suitable utility classes that it provides and testing to ensure that the styled component fits correctly with the design. Figure 5 illustrates the difference in code for presenting a button component using Tailwind and Bootstrap. While Tailwind needs to use five different utility classes to specify the button, Bootstrap only requires two simple classes that contain many different

CSS rules to determine a pre-designed button. Furthermore, as shown in Figure 5, Tailwind forces developers to select classes with detail properties, such as color of text and background, which is not necessary with Bootstrap. As a result, Tailwind requires more time and effort to complete a design than Bootstrap, especially with a project that does not have a unique design. In addition, Tailwind also requires more time for beginners to get acquainted with utility classes and understand the approach that Tailwind uses.

5 Comparison between Tailwind and component libraries

Component library refers to a set of pre-built UI components that can be used easily in front-end development to help developers save their time and effort [15]. The library offers many complete components that can satisfy common functionalities of modern web applications, including table, form, and menu [15]. In addition, the component library also provides several components that are already provided by plain HTML, including buttons and input, because the default style of these components in HTML can not fulfill the requirement of modern web applications, which is having a distinct and user-friendly interface [15]. Recently, several component libraries have been developed using Tailwind CSS from scratch for different front-end development frameworks, such as Tailwind UI (2020), Flowbite (2021), and DaisyUI (2022) [5], [8], [6].

5.1 Advantage of Tailwind over component library

Using Tailwind CSS offers a higher level of customization in comparison with the component library. While each utility class of Tailwind is used to determine a single property of the element, components provided by the library are completely encapsulated. These components refer to the black-box components that developers can directly use to implement the design without acknowledgment of the compositions inside them or the method that the library uses to create them. Therefore, pre-built components from the library are difficult to customize with specific details. In certain circumstances, the properties of these components can be overridden by additional CSS, however, this approach can lead to conflicts of styling, thus reducing the maintainability of the code. Although many modern component libraries enable developers to configure several prop-

erties, this level of customization is not sufficient to implement a design that is highly different from the design system of the library.

5.2 Disadvantages of Tailwind compared to component library

In comparison with component libraries, Tailwind CSS can complicate the code. Tailwind CSS has to use many different utility classes, which can occupy several lines of codes for class names, to visualize a component, especially with complex and detailed designs. In contrast, component libraries provide developers with comprehensively wrapped components that can be used directly in the format of an HTML element. For example, as shown in Listing 9, Tailwind requires the use of many different utility classes with different HTML elements, including `div`, `img` and `p`, to make a styled card component in Figure 6 that has UI effect on hovering. Whereas, as presented in Listing 8, a card component from the Ant Design library can significantly simplify the code and reduce the time spent on implementation. Consequently, Tailwind CSS requires more effort from developers in designing components to guarantee the reusability of the code. Otherwise, a large number of Tailwind utility class names can reduce the maintainability of the project.

```
<Card
    hoverable
    style={{ width: 240 }}
    cover={<img alt="example" src="https://os.alipayobjects.com/
        rmsportal/QBnOOoLaAfKPirc.png" />}
>
    <Meta title="Europe Street beat" description="
        www.instagram.com" />
</Card>
```

Listing 8. Card component from Ant Design library [2]

```
<div className='bg-white rounded-lg shadow w-60
    hover:cursor-pointer hover:shadow-xl'>
    <img alt='example' src='https://os.alipayobjects.com/
        rmsportal/QBnOOoLaAfKPirc.png'
        className='rounded-t-lg'/>
    <div className='p-6'>
        <p className='font-medium text-base mb-3'>
            Europe Street beat
        </p>
        <p className='font-light text-slate-500 text-sm'>
            www.instagram.com
        </p>
```

```
11      </div>
12  </div>
```

Listing 9. Card styled by Tailwind

Figure 6. An example for a card component [2]

6 Discussion

Tailwind CSS refers to a utility-based CSS framework that can fulfill many requirements of modern web applications without the need to write and manage separate CSS or style sheet files. However, Tailwind requires time and several techniques to be used efficiently in a project. In addition, although Tailwind CSS can be applied to different kinds of projects, this framework can address several difficult requirements of specific projects.

6.1 Using Tailwind CSS in practical projects

Tailwind provides utility classes for styling instead of pre-defined UI components [7], [17]. Therefore, using Tailwind in a real-world context is accompanied by a strong understanding of its utility classes as well as its prefix modifiers. Although the name of these utility classes has many similarities with that of traditional CSS properties, Tailwind CSS requires developers to spend much time gaining a complete understanding to use this CSS framework effectively. In addition, Tailwind CSS can slow down developers who were attached to arbitrary values in traditional CSS.

Utility classes provided by Tailwind are placed directly in the attribute `className` of HTML elements [7]. Therefore, these classes can complicate the implementation of complex components because these components require many different Tailwind classes. To update the style of a component, developers have to read through all the utility classes to find a specific class that needs to be changed, which requires much effort and time. As a result, Tailwind utility classes can reduce the readability of code and perplex developers who have to work on code from others. To address this problem, developers can determine components that need to be used multiple times in the project and separate them into reusable components to minimize the duplicate codes of utility classes.

6.2 CSS frameworks are suitable for different kinds of project

This paper presents the basic differences between Tailwind CSS and three other available options, which are inline styling, Bootstrap, and the component library. These alternatives are chosen because they are popular methods for styling web applications and have several similarities with Tailwind that can confuse developers when selecting a technology to use in projects. However, recently, many new CSS frameworks have been developed and have become increasingly popular for their modern features. Therefore, Tailwind can also be compared with other frameworks to gain more information.

This paper can not indicate that Tailwind can provide a more efficient approach to using CSS than other CSS frameworks. The most effective approach depends on the project because each project has a different scale and requirements for styling. Therefore, the most effective framework for a project refers to the framework that can adapt to the specific requirements of that project. Tailwind becomes the most suitable CSS framework for a project that requires a unique UI and does not need to be completed in a significantly short period.

7 Conclusion

This paper answers the research question: what is Tailwind CSS, and how does it differ from other available CSS frameworks? The answer is provided by examining the styling principles of Tailwind CSS and comparing Tailwind with other popular CSS frameworks. Tailwind CSS refers to a

modern CSS framework that uses utility-first CSS methodology. This CSS framework offers developers a set of primitive utility classes that can be used to determine the style of HTML components without writing separate CSS files [11]. In addition, Tailwind provides utility modifiers to fulfill the requirements of modern web applications, including styling of component state, dark mode, and responsive design.

Although both Tailwind CSS and inline styling apply directly the style to the HTML elements, utility classes of Tailwind provide a more consistent method for styling than inline styling. Moreover, Tailwind also supports responsive design and pseudo-class that can not be implemented with inline styling.

While Tailwind helps applications reduce load time and offers significant customization, the Bootstrap framework can be used more easily, thus decreasing the effort of developing. Similarly, Tailwind offers a higher level of customization than component libraries, this CSS framework can complicate the code and require more time for implementation.

While this paper presents the core concept of Tailwind CSS and illustrates the differences between Tailwind and other approaches, at least the following open research questions remain:

- How can Tailwind CSS be improved or extended to optimize the styling process without compromising its core philosophy of utility-first CSS?

- How does Tailwind CSS affect the performance of modern web pages?

References

[1] Adding style to html. https://www.w3.org/TR/html401/present/styles.html. Accessed on March 25th, 2024.

[2] Ant design. https://ant.design. Accessed on March 25th, 2024.

[3] Comparation between utility class and inline style. Accessed on March 25th, 2024.

[4] Css snapshot 2023. https://www.w3.org/TR/CSS/. Accessed on March 25th, 2024.

[5] Daisyui. https://daisyui.com/. Accessed on March 25th, 2024.

[6] Flowbite. https://flowbite.com. Accessed on March 25th, 2024.

[7] Tailwind css. https://tailwindcss.com. Accessed on January 25th, 2024.

[8] Tailwind ui. https://tailwindui.com. Accessed on March 25th, 2024.

[9] *Modern CSS with Tailwind, 2nd Edition*. Pragmatic Bookshelf, 2022.

[10] S Shahu Gaikwad and PRATIBHA Adkar. A review paper on bootstrap framework. *IRE Journals*, 2(10):349–351, 2019.

[11] Ivaylo Gerchev. *Tailwind CSS : craft beautiful, flexible, and responsive designs*. SitePoint Pty. Ltd., Australia, [first edition]. edition, 2022.

[12] Eric A. Meyer. *CSS pocket reference*. O'Reilly, Sebastopol, California, 3rd ed. edition, 2008.

[13] Kailashkumar V Natda. Responsive web design. *Eduvantage*, 1(1), 2013.

[14] Fadli Rifandi, Tri Adriansyah, and Rina Kurniawati. Website gallery development using tailwind css framework. *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, 6:205–214, 12 2022.

[15] Sanna Salonen. Evaluation of ui component libraries in react development. 2023.

[16] Maryam Shokrinejad Shirazifard. *Tailwind CSS (Software framework)*. 08 2023.

[17] Marzieh Somi. *User Interface Development of a Modern Web Application*. PhD thesis, Politecnico di Torino, 2023.

Encrypted DNS and Privacy

Emnet Mehari Tekeste
emnetmehari.tekeste@aalto.fi

Tutor: Tuomas Aura

Abstract

As internet use has become evermore prevalent, considerable work has been undertaken to improve the privacy of the communications that happen over it. However, for many years, one of its most important building blocks, the Domain Name System (DNS), had remained a blind spot in these efforts. Recent years have seen significant efforts to remedy this. These efforts have primarily been directed towards encrypting DNS traffic on the wire. While the results of these efforts have served to make DNS traffic more secure in several aspects, there remain notable risks related to side-channel attacks and logged DNS request data.

KEYWORDS: DNS, DoH, DoT, DoQ, Privacy

1 Introduction

The Domain Name System (DNS) is a central component of internet infrastructure. In simple terms, DNS is the distributed directory service for resolving human-readable addresses into numerical addresses that are used for routing on the internet. A device that seeks to access a domain on the internet will connect to one or many DNS servers to retrieve the numerical IP address of the domain and then initiate a connection to the

IP address [12, 20].

The original DNS design grew out of a need to accommodate more varied usage of the internet [20], and while these needs were met, the design overlooked the many privacy implications of such a system. Over the years, much work has been done to both study the privacy implications [25] and to try and design security improvements [13, 11, 15]. While the most popular DNS resolvers now support these additions to DNS [8], most DNS traffic remains unencrypted [18].

This paper examines different approaches to achieve a more private DNS, their potential shortcomings, and their current prevalence.

The rest of the paper is organized as follows. Section 2 will provide an overview of DNS and define some terms that will be used throughout. Section 3 will outline the potential threats to privacy at various components of the system. Section 4 will discuss proposed solutions and the degree to which they are currently in use, while section 5 looks at possible problems in those solutions. Section 6 discusses the findings. Lastly, section 7 concludes the paper. While an overview of DNS and its privacy extensions is provided, an understanding of the fundamentals of networking is assumed.

2 Background

The DNS consists of three main components: resolvers, name servers, and the domain name space and resource records. In the simplest mode of operation, a resolver makes a query to a name server that responds with the relevant resource record. However, on the internet scale, domain name resolution involves multiple queries following the initial request with caching at several points along the path [20]. Below is a broad overview of the main components of DNS.

i **Resolver:** The resolver is included in the client-side operating system or in an application such as a web browser. It sends a query to a name server on another host in order to resolve a domain, although it might cache the information locally. Its most common function is to resolve host names to host addresses, but the reverse is also possible [20].

ii **Name server (NS):** The name servers store the database of mappings divided into *zones*. Zones are subtrees within the tree-like structure of

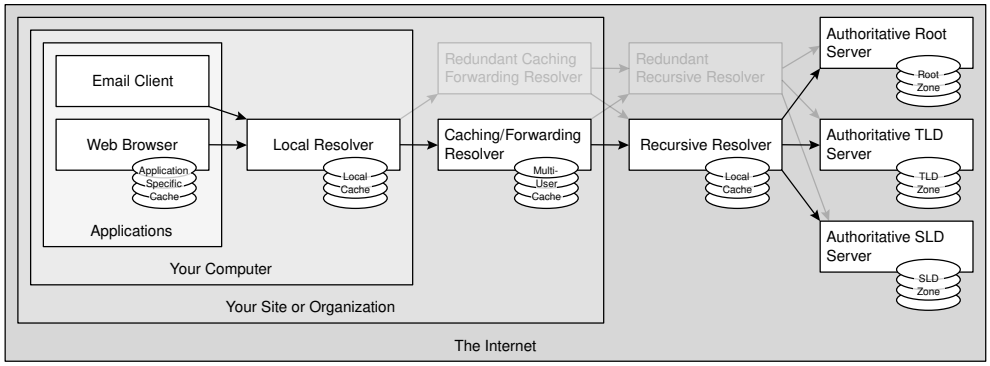


Figure 1. Possible DNS resolution sequence [Aaron Filbert, CC BY-SA 4.0, via Wikimedia Commons]

the domain name space. Upon receiving a query, a name server may respond with the requested data if it is available in its zone or a referral to a name server that could have it. The zones will generally be redundantly available on multiple name servers, and mechanisms exist to propagate changes [19]. Recursive name servers are those that can perform the entire name resolution on behalf of a user, sending requests to other servers as required. Recursive servers are maintained by the Internet Service Providers (ISP), or they can be open services, such as Google Public DNS.

iii **Resource Record (RR):** A resource record [20, 21] is the information contained within a node of the domain name space tree, which may also be empty. A RR may be of different types based on the information it contains, such mail exchange (MX) or IPv4 host address (A).

All communication over DNS is conducted via a single, extensible message format. The message includes header information, such as whether it is a query or response, or a normal, reverse or other type of query [21]. While these messages are most commonly transferred over UDP, TCP support has been a requirement for DNS since 2016 [9].

Subsequent sections will assess what privacy threats exist at the components mentioned above and interactions between them, along with what new threats may be introduced with proposed alternatives. Following previous similar works [25], this paper uses privacy-related terminology as defined in RFC6973 [5].

3 Privacy Considerations for Traditional DNS

A discussion of the privacy issues of traditional DNS remains relevant as most DNS traffic still remains unencrypted. Two scenarios to consider are the case where there exists an eavesdropper somewhere on the path and the case where the resolvers themselves are spying on their users. In both cases, the unencrypted nature of DNS undermines security.

Recursive resolvers are more prone to both of these issues compared to other servers in the hierarchy. This is not only because the heavily cached nature of DNS means that they see the most traffic, but also because they receive the user's IP address as part of the request, whereas other servers might only have access to the domain the user is trying to visit. Thus, the path from the user to the recursive resolver is the most ideal target for an eavesdropper. Malicious recursive resolvers also have the most information available to them.

Previous work has demonstrated the possibility to identify different users given sufficient logs of their DNS requests, even with changing IP addresses. For instance, research [17] demonstrates the possibility to distinguish individual users with considerable accuracy and track their activity across sessions, with this accuracy increasing further if the users are highly active or known to visit highly distinguishable domains. In addition to correlating user browsing sessions, work [1] has been done to show that DNS traffic analysis can be used to correlate multiple requests of the same session, such that, for instance, different secondary requests generated from a single visit to a webpage can be identified as part of a single activity.

Furthermore, the increasing adoption of public recursive resolvers has necessitated the inclusion of all or a portion of the user's IP address in the request from the recursive to the authoritative NS [4] for performance reasons. This can reveal a user's location even behind a VPN. The added information, along with introducing another point of weakness to the hierarchy, can be used to perform cache poisoning attacks on select subsets of users [16] by a malicious actor.

4 Encrypted DNS

Several encrypted alternatives to traditional DNS have been proposed and have seen different degrees of adoption. A quick overview of the most

prominent ones is given below.

4.1 DNS over TLS

DNS over TLS was published in 2016 as RFC7858 [13]. DoT operates in the same manner as traditional DNS, but over a TCP connection and encrypted with TLS. While there exists a proposal for DoT that utilizes UDP, it remains experimental and is considered out of the scope of this paper. DoT uses a dedicated port, 853 by default, and requires that the port not be 53 — the designated port for regular DNS traffic.

DoT aims to service two use cases [13]. The first is an opportunistic mode that favors availability over complete privacy. It allows a client to use a DNS server discovered via a means such as DHCP without first validating it. The second guarantees full privacy but requires that the clients and servers have an established trust relationship. While this is feasible in settings such as those in enterprises, it may pose some discoverability problems in others.

4.2 DNS over HTTPS

RFC8484 [11], published in 2018, defines DNS over HTTPS (DoH). DoH delivers its DNS messages via HTTP GET or POST requests over HTTPS. The utilization of HTTP implies that, by design and unlike DoT, DoH does not have a dedicated port and transports its messages intermingled with normal HTTP traffic. Similar to the more strict mode of DoT, discovery and configuration of a DoH server for use is conducted out-of-band.

4.3 DNS over QUIC

DNS over QUIC (DoQ) [14] was standardized in 2022 and is the most recent addition to the list of DNS standards. DoQ aims to provide the same protections as DoT on top of QUIC, which additionally grants it decreased latency in the initial request. It similarly uses port 853 and offers opportunistic or strict usage profiles. In contrast to DoT and DoH, DoQ also explicitly addresses privacy considerations for traffic between the recursive and authoritative servers.

5 Privacy considerations for Encrypted DNS

Each of the encrypted DNS protocols mentioned above primarily intend to address problems associated with DNS traffic on-the-wire. Encryption of said traffic renders passive eavesdropping attacks infeasible. However, certain active attacks might still be possible.

The most distinguishing aspect of DoH is its use of HTTP and the subsequent intermingling of DNS traffic with regular HTTPS traffic. This complicates passive analysis of DNS traffic for the passive observer. It also renders identifying and blocking the DNS traffic more difficult although not impossible. As Csikor et al [6] have demonstrated, a machine learning based approach could identify DoH traffic with a negligible false positive rate. HTTP will also necessarily entail the inclusion of additional information with every request that would not otherwise be present [11] in a DNS message, possibly to the benefit of a rogue recursive resolver.

DoT and DoQ can largely be discussed together as they are similar in many ways and share the same goals. Unlike DoH, DoQ and DoT do not require server authentication in their opportunistic modes and may provide reduced protection as a result. Additionally, the use of a dedicated port might trivialize efforts to restrict or monitor their use.

These opportunistic modes of operation of DoQ and DoT also hint at a problem of discovery and negotiation. The Adaptive DNS Discovery Working Group works with the mandate to draft unified methods for discovering encrypted DNS servers and determining their capabilities and this issue of discovery remains an active area of work.

Despite their encryption, all three protocols may be prone to some side channel attacks in their basic mode of operation. Encrypted DNS traffic is susceptible to fingerprinting attacks. In addition to the different variations possible in regular internet traffic, DNS has added factors that could aid in fingerprinting, such as the state of the cache at the recursive resolver or the effects of load balancing [23].

For instance, Siby et al. [23] have demonstrated that it is significantly easier to perform DNS fingerprinting to surveil browsing activity than similar attacks on web traffic due to factors such as the smaller sizes of DNS messages. They found that despite some variations based on the resolvers in use and host location, DNS fingerprinting offers considerable precision.

Mitigation strategies for fingerprinting attacks exist and offer varying

levels of efficacy. Padding the payload to equalize sizes is recommended by all three protocol specifications. This is also a default for popular DoH and DoT providers such as Cloudflare and Google. While the padding does mitigate fingerprinting to some degree, it does not entirely prevent it. Siby et al. [23] found the usefulness of the padding to prevent fingerprinting to be below expectation, but also noted that DoT appears to be less susceptible to such attacks, possibly as a result of having a lower average number of requests per domain. Another study [3] similarly noted that padding alone is insufficient and recommended its use along with some mechanism to ensure uniform timing of packets.

6 Further discussion

While the methods discussed in the previous section improve DNS privacy on-the-wire, little has been said about ensuring privacy of the queries at the name servers. Different attempts have been made over the years to remedy this. Oblivious DNS over HTTPS [15, 24] is a notable recent development. It uses two servers, a DNS server and a proxy, such that neither one is aware of both the client's source IP and the contents of the DNS message. Cooperative DNS servers can also implement their own ways to ensure added protection. For instance, Mullvad recently announced that all of its encrypted DNS servers would run entirely on the RAM to ensure that no data will persist [2].

Another aspect of DNS worth mentioning is the link between the recursive and the authoritative resolver. As discussed in section 2, modern internet infrastructure had necessitated that this link carry a considerable amount of information. However, recent ventures have been increasingly successful in diminishing the information being transmitted. A notable example is QNAME Minimization which is a technique that attempts to reduce the amount of information authoritative servers are privy to by ensuring that a recursive server will only send them requests consisting of information they are likely to have. This technique, while having some impact on performance, could also improve privacy [7].

Furthermore, it ought to be emphasized that all of the efforts mentioned here fall far short of deterring any adversary with sufficient means and commitment. Even if DNS data is completely private, internet traffic will necessarily leak information elsewhere, such as in the Server Name Indication (SNI) of the HTTPS handshake. While plenty of work is on-

going to reduce all kinds of privacy risks, for instance, Encrypted Client Hello to hide SNI [22], it will perhaps never be enough. Regardless, these efforts are worth undertaking with the intent to *"increase the cost of attacking, force what was covert to be overt, or make the attack more likely to be detected"* [10].

There also exist several non-technical questions that warrant mention in any discussion of privacy. Whether there are legitimate reasons, such as safety, to prevent the adoption such encrypted technologies will likely remain an open debate.

7 Conclusion

Internet privacy has been a trending issue over the past couple of years and DNS, being the bedrock of the modern internet, has been no exception. As such, there is a plethora of work being done to address the many issues present in its design. This work has primarily come in the form of efforts to encrypt DNS traffic on-the-wire that had remained in plaintext for most of its existence. While considerable leaps have been made in this regard, much work remains to be done to completely secure all the traffic. Moreover, securing DNS traffic alone is no panacea for internet privacy, and it should be remembered that it is impossibly difficult to conceive technical solutions to what are, perhaps, social and political problems.

References

[1] Ignacio N. Bermudez, Marco Mellia, Maurizio M. Munafo, Ram Keralapura, and Antonio Nucci. DNS to the rescue: discerning content and services in a tangled web. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 413–426, New York, NY, USA, 2012. Association for Computing Machinery. https://doi.org/10.1145/2398776.2398819.

[2] Martin Brinkmann. Mullvad's public encrypted DNS Servers run in RAM now - gHacks Tech News — ghacks.net, 2023. [Accessed 26-03-2024].

[3] Jonas Bushart and Christian Rossow. Padding ain't enough: Assessing the privacy guarantees of encrypted DNS. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association, August 2020.

[4] Carlo Contavalli, Wilmer van der Gaast, David C Lawrence, and Warren "Ace" Kumari. Client Subnet in DNS Queries. RFC 7871, May 2016. https://doi.org/10.17487/RFC6891.

[5] Alissa Cooper, Hannes Tschofenig, Dr. Bernard D. Aboba, Jon

Peterson, John Morris, Marit Hansen, and Rhys Smith. Privacy Considerations for Internet Protocols. RFC 6973, July 2013. https://doi.org/10.17487/RFC6973.

[6] Levente Csikor, Himanshu Singh, Min Suk Kang, and Dinil Mon Divakaran. Privacy of DNS-over-HTTPS: Requiem for a dream? In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 252–271, 2021. https://doi.org/10.1109/EuroSP51992.2021.00026.

[7] Wouter B. de Vries, Quirin Scheitle, Moritz Müller, Willem Toorop, Ralph Dolmans, and Roland van Rijswijk-Deij. A first look at qname minimization in the domain name system. In David Choffnes and Marinho Barcellos, editors, *Passive and Active Measurement*, pages 147–160, Cham, 2019. Springer International Publishing. https://doi.org/10.1007/978-3-030-15986-3.

[8] Casey Deccio and Jacob Davis. DNS privacy in practice and preparation. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, CoNEXT '19, pages 138–143, New York, NY, USA, 2019. Association for Computing Machinery. https://doi.org/10.1145/3359989.3365435.

[9] John Dickinson, Sara Dickinson, Ray Bellis, Allison Mankin, and Duane Wessels. DNS Transport over TCP - Implementation Requirements. RFC 7766, March 2016. https://doi.org/10.17487/RFC7766.

[10] Stephen Farrell and Hannes Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258, May 2014.

[11] Paul E. Hoffman and Patrick McManus. DNS Queries over HTTPS (DoH). RFC 8484, October 2018. https://doi.org/10.17487/RFC8484.

[12] Paul E. Hoffman, Andrew Sullivan, and Kazunori Fujiwara. DNS Terminology. RFC 8499, January 2019. https://doi.org/10.17487/RFC8499.

[13] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016. https://doi.org/10.17487/RFC7858.

[14] Christian Huitema, Sara Dickinson, and Allison Mankin. DNS over Dedicated QUIC Connections. RFC 9250, May 2022. https://doi.org/10.17487/RFC9250.

[15] Eric Kinnear, Patrick McManus, Tommy Pauly, Tanya Verma, and Christopher A. Wood. Oblivious DNS over HTTPS. RFC 9230, June 2022. https://doi.org/10.17487/RFC9230.

[16] Panagiotis Kintis, Yacin Nadji, David Dagon, Michael Farrell, and Manos Antonakakis. Understanding the privacy implications of ecs. In Juan Caballero, Urko Zurutuza, and Ricardo J. Rodríguez, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 343–353, Cham, 2016. Springer International Publishing.

[17] Matthias Kirchler, Dominik Herrmann, Jens Lindemann, and Marius Kloft. Tracked without a trace: Linking sessions of users by unsupervised learning of patterns in their DNS traffic. In *Proceedings of the 2016 ACM*

Workshop on Artificial Intelligence and Security, AISec '16, pages 23–34, New York, NY, USA, 2016. Association for Computing Machinery. https://doi.org/10.1145/2996758.2996770.

[18] APNIC Labs. ORR Encrypted DNS Use by Country.

[19] Edward P. Lewis and Alfred Hoenes. DNS Zone Transfer Protocol (AXFR). RFC 5936, June 2010. https://doi.org/10.17487/RFC5936.

[20] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, November 1987. https://doi.org/10.17487/RFC1034.

[21] P. Mockapetris. Domain names - implementation and specification. RFC 1035, November 1987. https://doi.org/10.17487/RFC1035.

[22] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-18, Internet Engineering Task Force, March 2024. Work in Progress.

[23] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS => Privacy? A Traffic Analysis Perspective. In *Proceedings of the 2020 Network Distributed System Security Symposium (NDSS)*. The Internet Society, February 2020. https://doi.org/10.14722/ndss.2020.24301.

[24] Sudheesh Singanamalla, Suphanat Chunhapanya, Jonathan Hoyland, Marek Vavruša, Tanya Verma, Peter Wu, Marwan Fayed, Kurtis Heimerl, Nick Sullivan, , and Christopher Wood. Oblivious DNS over HTTPS (ODoH): A Practical Privacy Enhancement to DNS. In *Proceedings on Privacy Enhancing Technologies Symposium*, 2021. https://doi.org/10.2478/popets-2021-0085.

[25] Tim Wicinski. DNS Privacy Considerations. RFC 9076, July 2021. https://doi.org/10.17487/RFC9076.

Physics-Inspired Deep Learning for Climate Forecasting

Fikri Şan Köktaş
san.koktas@aalto.fi

Tutor: Yogesh Verma

Abstract

This paper reviews state-of-the-art climate forecasting models that leverage physics-inspired deep learning. Working mechanisms of GenCast, NeuralGCM, ClimaX, and ClimODE are analyzed, thereby evaluating their strengths and weaknesses. Models integrating physics-inspired deep learning offer promising alternatives to traditional simulation-based approaches.

KEYWORDS: climate forecasting, climate models, deep-learning, physics

1 Introduction

In the past, computer simulations based on atmospheric physics were used to predict climate. However, recent progress in deep learning has changed this approach. The newer models can make weather forecasts without relying fully on simulations. Instead, they leverage the power of machine learning algorithms. However, it is often hard to understand how these models make predictions. For example, determining the precise influence of each parameter on the outcome poses a significant challenge. Moreover, these models don't consider all of the important factors that affect the climate. This paper presents a literature review of the state-of-the-art models and explains how they tackle these issues.

2 Literature Review

Climate modeling extends beyond daily weather forecasts, covering long-term climate patterns and trends over decades [1]. This section focuses on 4 state-of-the-art climate forecasting models: GenCast, SRGAN, NeuralGCM, ClimaX, and ClimODE, respectively.

2.1 GenCast

GenCast is a machine learning-based generative model for ensemble weather forecasting. It yields faster, more reliable, and more accurate results than conventional ensemble forecasting systems [2].

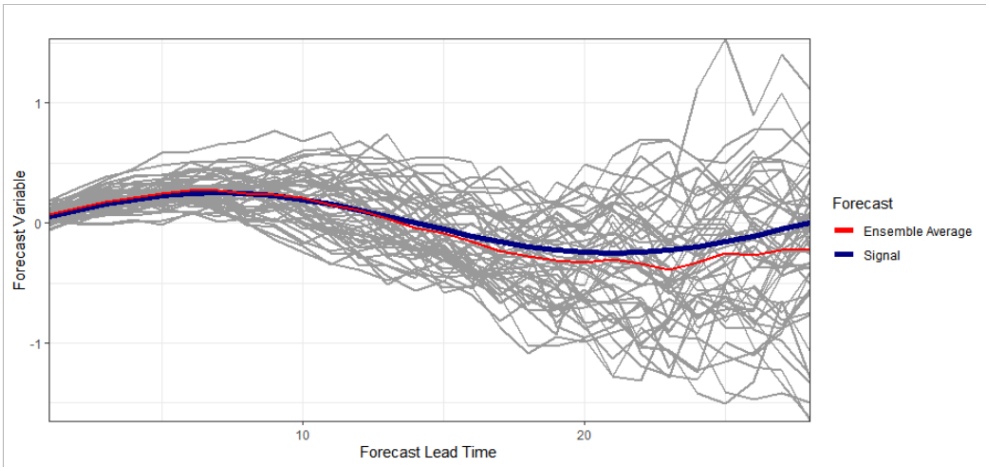


Figure 1. Figure illustrating the principle of ensemble weather forecasting [3]

In ensemble weather forecasting, the model is run multiple times with different initial conditions to generate multiple possible outcomes [4]. This approach helps to understand the uncertainty of predictions and increases the reliability of the predictions by yielding multiple possible outcomes.

Table 1. Underlying Mathematical Equation for Ensemble Learning Models used within GenCast

Ensemble Learning Model	**Mathematical Expression**
General Form	$\hat{f}(x) = \frac{1}{M} \sum_{m=1}^{M} h_m(x)$

GenCast leverages a diffusion model. Diffusion models utilize equations that describe the physics of diffusion (mixing through random molecular motion [5]) and advection (bulk movement of a fluid [6]) .

GenCast can forecast ensembles of trajectories for 84 different weather variables, up to 15 days ahead [2]. GenCast is known for its skillfulness, fast performance, and reliability. It achieved a higher accuracy than

other ensemble forecast systems like European Centre for Medium-Range Weather Forecasts's (ECMWF) Ensemble Prediction System (ENS), while also reducing the computation time [2].

Table 2. Mathematical expressions for diffusion and advection used within GenCast

Equation Type	Equation
Diffusion Equation	Fick's first law: $J = -D\nabla c$ Diffusion equation: $\frac{\partial c}{\partial t} = D\nabla^2 c$
Advection Equation	Advection equation: $\frac{\partial c}{\partial t} + (\mathbf{v} \cdot \nabla)c = 0$

2.2 NeuralGCM

Neural General Circulation Model (NeuralGCM), combines neural networks with traditional general circulation models (GCMs) [7]. Traditional GCMs utilize physics-based models to simulate Earth's climate system [8]. NeuralGCMs introduce neural networks to traditional GCMs, thereby decreasing computational costs, improving the quality of simulations, and enhancing the accuracy of predictions.

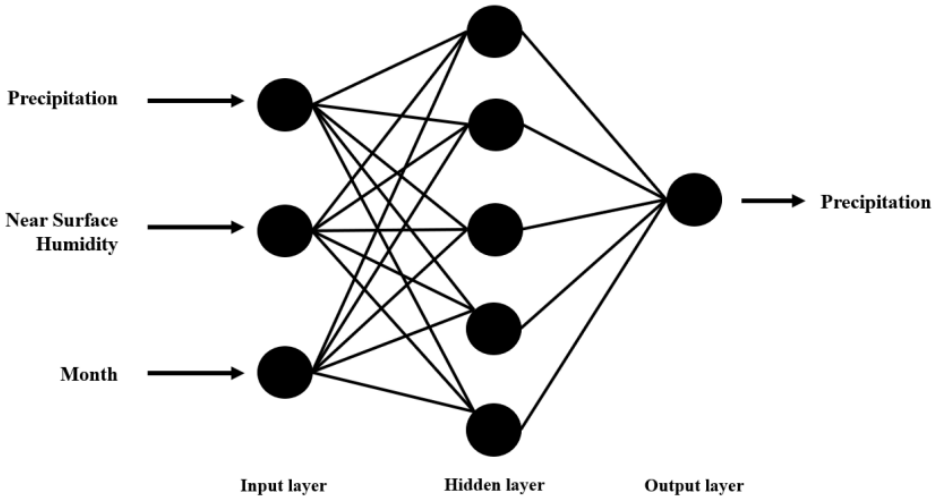


Figure 2. Diagram showing how neural network structure predicts precipitation [9]

NeuralGCM combines a differentiable solver for atmospheric dynamics with machine-learning components. It uses Fifth Generation ECMWF Reanalysis data (ERA5) for training, and it aims for up to 5-day accurate weather forecasts [7].

Table 3. Fundamental Components of Neural Networks used in NeuralGCM

Component	Equation / Description
Linear Transformation	$z = Wx + b$
Activation Function	$a = \sigma(z)$
Loss Function	$L = f(y, \hat{y})$
Optimization	$\theta = \theta - \alpha \nabla_\theta L$
Backpropagation	$\frac{\partial L}{\partial w_{ij}^l} = \delta_i^l a_j^{l-1}$

Table 4. Fundamental Equations of GCMs used in NeuralGCM

Equation	Equation
Navier-Stokes	$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{g}$
Continuity	$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$
Thermodynamic Equation of State	$p = \rho R T$
Energy Conservation	$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \frac{1}{c_p \rho} \nabla \cdot (\kappa \nabla \theta) + \frac{1}{c_p \rho} Q$
Radiative Transfer	$\frac{dI_\lambda}{ds} = -(\kappa_\lambda \rho + \sigma_\lambda) I_\lambda + \epsilon_\lambda B_\lambda(T)$

2.3 ClimaX

ClimaX is a deep-learning-based model that is specifically designed for weather and climate science [10]. It uses a type of neural network architecture called the Transformer architecture. Transformer architecture has an advanced attention mechanism that allows the model to focus on the most important parts of the input sequence [11]. ClimaX can be fine-tuned for multiple climate-related tasks such as predicting weather and climate changes. It is more flexible, generalizable, and efficient than most other data-driven models [10].

ClimaX is pre-trained on large, unsupervised datasets [10]. It should be fine-tuned for specific tasks. However, this can be effortlessly done as ClimaX supports distributed training through PyTorch Lightning [10].

ClimaX's architecture uses Vision Transformers (ViT) [10], which are a type of neural network architecture designed for processing images. For feature extraction, ViTs use self-attention mechanisms, unlike traditional CNNs that use convolutions [12]. ViTs of ClimaX are adapted for the different types of weather and climate data requirements. This is achieved through variable tokenization and aggregation. These strategies are used to handle the complexity and irregularity of climate datasets. ClimaX achieved promising results even for variables not seen during pre-training [10].

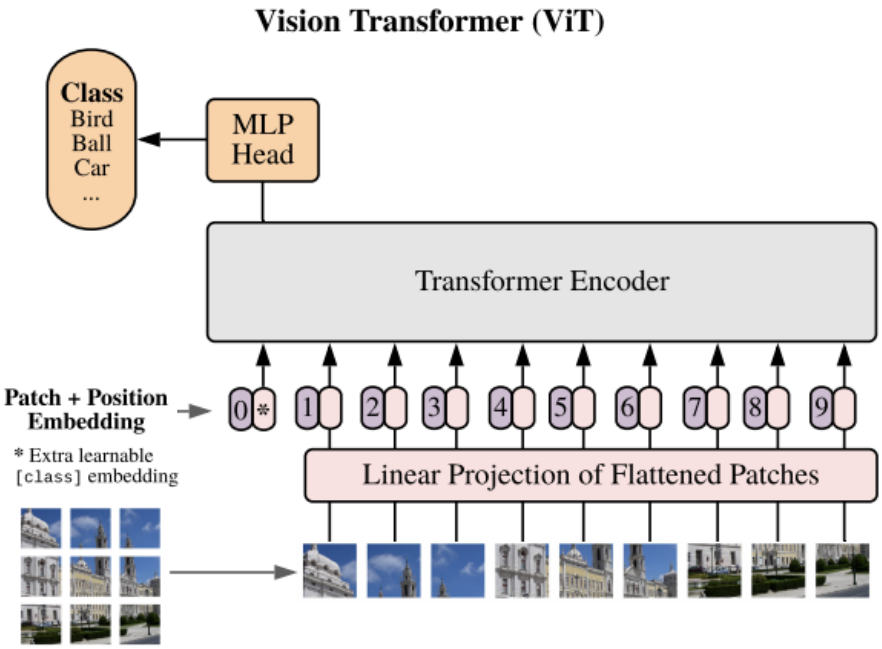


Figure 3. Diagram explaning the working mechanism of vision transformers [13]

Table 5. Underlying math equations of Vision Transformers

Equation	Description
Self-Attention Mechanism	$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$
Feedforward Neural Networks	$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$
Positional Encodings	$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$ $PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$

2.4 ClimODE

ClimODE utilizes physics-informed Neural Ordinary Differential Equations (ODEs) for climate forecasting [14]. Unlike data-driven models like transformers, ClimODE considers the underlying physics of climate change. ClimODE also shows better performance than data-driven models by achieving a higher level of accuracy and fewer parameters [14].

ClimODE considers the movement of quantities (e.g. water, air) over time (referred to as advection in statistical mechanics [6]) which can affect weather changes. It uses a neural network to understand the movement of weather-related elements (e.g. heat, moisture, etc.) and how they interact with each other [14]. Moreover, ClimODE considers the fact that quantities like energy or momentum are conserved based on the laws of physics.

ClimODE uses a Continuous-time PDE model [14], which helps to model climate processes without being limited to specific time intervals. This flexibility can result in more precise and smooth simulations of climate events.

Table 6. Some simplified ODEs that could be used within ClimODE.

Name of the Equation	**Mathematical Expression**
Radiative Transfer Equations	$\frac{dI}{dx} = -\alpha I + \beta$
Chemical Kinetics	$\frac{d[A]}{dt} = -k[A]$
Simple Carbon Cycle Models	$\frac{dC}{dt} = I - O$
Sea Ice Models	$\frac{dS}{dt} = G - M$
Ecosystem Models	$\frac{dP}{dt} = \alpha P - \beta PR$, $\frac{dR}{dt} = \gamma\beta PR - \delta R$
Ocean Circulation Models	$\frac{dT}{dt} = F(T,S)$, $\frac{dS}{dt} = G(T,S)$
Glacial Dynamics Models	$\frac{dh}{dt} = I - O$

Table 7. General Form of Continuous-time PDEs used within ClimODE

Continuous-time PDE	**Mathematical Expression**
General Form	$\frac{\partial u}{\partial t} = F\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \dots\right)$

3 Discussion

Model Name	Year	Type	Stand-out Feature
GenCast [2]	2023	GM	Computational Efficiency
NeuralGCM [7]	2023	GM	Computational Efficiency
ClimaX [10]	2023	TS	Flexibility
ClimODE [14]	2024	TS	Flexibility

Table 8. Comparison of models. "GM" stands for generative modeling whereas "TS" stands for time series forecasting

Model	**Computational Resource Requirements**
GenCast	Low
NeuralGCM	Low
ClimaX	High
ClimODE	Medium

Table 9. Computational Resource Requirements of Different Models

GenCast and NeuralGCM stand out with their computational efficiency while ClimaX and ClimODE stand out with their flexibility. ClimODE uses similar deep-learning techniques to NeuralGCM or GenCast, but it incorporates more computationally intensive algorithms. Therefore, its computational resource requirements were classified as "Medium". On the other hand, ClimaX can require high computational resources especially if the dataset is large.

4 Conclusion

This section summarizes this paper, presents the main findings, and gives recommendations for future research.

4.1 Summary

This paper provides an overview of state-of-the-art climate forecasting models, focusing on their unique attributes, methodologies, and applications.

GenCast is a machine learning-based model for ensemble weather forecasting that leverages diffusion models to predict weather variables up to 15 days ahead. It achieves high accuracy and reduced computation time.

NeuralGCM combines neural networks with traditional GCMs to simulate Earth's climate system. It achieves improvements in computational efficiency, simulation quality, and prediction accuracy.

Climax is a flexible deep-learning model based on the Transformer architecture. It uses Vision Transformers adapted for the irregular nature of climate data. It is pre-trained on large datasets for weather and climate science tasks, but it needs to be fine-tuned for specific objectives.

ClimODE uses physics-informed Neural Ordinary Differential Equations (ODEs) to forecast climate, focusing on the physics of climate change. It models climate processes continuously over time without being limited to specific time intervals.

4.2 Main Findings

- GenCast and NeuralGCM stand out with their computational efficiency while ClimaX and ClimODE stand out with their flexibility.

- The choice of model should depend on individual priorities, whether

they prioritize accuracy, computational efficiency, flexibility, or other factors.

- Models that leverage physics-inspired deep learning can be a faster and more accurate alternative to traditional models that rely fully on simulations.

4.3 Future Research

- Future research should focus on decreasing the computational costs and training time of the models.

- Hybrid models that combine physics inspired deep learning models with traditional models should be explored further.

References

[1] R. McSweeney, "Q&a: How do climate models work?." https://www.carbonbrief.org/qa-how-do-climate-models-work/. [Accessed: Mar 15, 2024].

[2] I. Price, A. Sanchez-Gonzalez, F. Alet, T. Ewalds, A. El-Kadi, J. Stott, S. Mohamed, P. Battaglia, R. Lam, and M. Willson, "Gencast: Diffusion-based ensemble forecasting for medium-range weather," 2023. [Accessed: Apr 3, 2024].

[3] J. Dutton, "The difference between deterministic and ensemble forecasts," Jun 2022. [Accessed: Mar 30, 2024].

[4] D. Simecek-Beatty, "Chapter 11 - oil spill trajectory forecasting uncertainty and emergency response," in *Oil Spill Science and Technology* (M. Fingas, ed.), pp. 275–299, Boston: Gulf Professional Publishing, 2011. [Accessed: Apr 2, 2024].

[5] Libretexts, "12.9: Diffusion," Nov 2020. [Accessed: Mar 29, 2024].

[6] F. Phillips and M. Castro, "5.15 - groundwater dating and residence-time measurements," in *Treatise on Geochemistry* (H. D. Holland and K. K. Turekian, eds.), pp. 451–497, Oxford: Pergamon, 2003. [Accessed: Apr 3, 2024].

[7] D. Kochkov, J. Yuval, I. Langmore, P. Norgaard, J. Smith, G. Mooers, J. Lottes, S. Rasp, P. Düben, M. Klöwer, S. Hatfield, P. Battaglia, A. Sanchez-Gonzalez, M. Willson, M. P. Brenner, and S. Hoyer, "Neural general circulation models," 2023. [Accessed: Apr 1, 2024].

[8] A. Broccoli, "Paleoclimate modeling of last glacial maximum gcms," in *Reference Module in Earth Systems and Environmental Sciences*, Elsevier, 2014. [Accessed: Apr 3, 2024].

[9] C. Prathom and P. Champrasert, "General circulation model downscaling using interpolation—machine learning model combination—case study: Thailand," *Sustainability*, vol. 15, no. 12, 2023. [Accessed: Mar 30, 2024].

[10] T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, and A. Grover, "Climax: A foundation model for weather and climate," 2023. [Accessed: Apr 3, 2024].

[11] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, 2022. [Accessed: Mar 30, 2024].

[12] F. C. Morabito, R. Kozma, C. Alippi, and Y. Choe, "1 - advances in ai, neural networks, and brain computing: An introduction," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing (Second Edition)* (R. Kozma, C. Alippi, Y. Choe, and F. C. Morabito, eds.), pp. 1–8, Academic Press, second edition ed., 2024. [Accessed: Apr 3, 2024].

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Accessed: Apr 3, 2024].

[14] Anonymous, “ClimODE: Climate forecasting with physics-informed neural ODEs,” in *The Twelfth International Conference on Learning Representations*, 2024. [Accessed: Mar 31, 2024].

State Management Solutions in React Web Development Library

Furkan Ün
furkan.un@aalto.fi

Tutor: Juho Vepsäläinen

Abstract

This paper explores various state management strategies and libraries within the React ecosystem, highlighting the challenges they address and the options available to developers.

KEYWORDS: react, state management, prop drilling, redux, zustand,mobx, jotai, recoil, valtio,

1 Introduction

In the early 1990s, the landscape of the World Wide Web was different from what we experience today. Web pages were static, with fixed content that remained the same for all users, regardless of their preferences or interactions. Recently, modern web pages have transitioned towards a more dynamic and personalized experience. As web applications evolve towards dynamic user interfaces, the need to efficiently handle and manipulate the business logic on the client side becomes increasingly important [25].

The emergence of single-page applications (SPAs) represented a significant advancement toward achieving dynamic and interactive web experiences. React library emerged as a key player in the development of SPAs. It enables developers to build more scalable and maintainable web

applications with a component-based architecture [13].

In React, state refers to a plain javascript object and acts as a memory for components. [21]. As the number of components, and interactions between these components grows, effective state management can become a crucial challenge for application performance and user experience [24]. To mitigate these issues, the React ecosystem has developed several libraries and solutions dedicated to state management.

This paper reviews the different state management strategies and libraries, with a particular focus on the React ecosystem. It aims to explore the current main state management options available to developers using the React library for web development.

This paper is organized as follows. Section 2 describes the state and necessity of state management solutions, and the problems they are addressing in complex React applications. Section 3 describes the most widely used state management libraries in React ecosystem. Finally, Section 4 provides concluding remarks.

2 State Management

In recent years, single-page web applications (SPAs) have emerged as a prominent solution to increasing demand for highly reactive client-side applications. Unlike traditional multi-page websites, SPAs load a single HTML page and dynamically update content as the user interacts with the app, minimizing page reloads and improving user experience [4]. This approach requires robust state management to track changes in the application state in response to user actions.

2.1 Components

Components are building blocks of SPAs and help create reusable views. They are often classified into two main categories: stateful and stateless. While stateful components actively keep track of changes and update themselves accordingly(dynamic), stateless components are solely responsible for rendering the properties that are passed to them without maintaining any internal state (static). Figure 1 depicts how state changes are reflected in view. In Figure 1,

State refers to a plain javascript object and acts as a “memory” for components to keep track of changes.

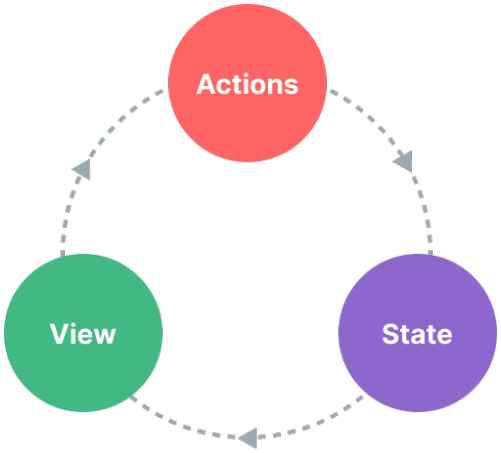


Figure 1. One-way data flow [https://redux.js.org/tutorials/essentials/part-1-overview-concepts]

View stands for declarative representation of user interface in different states

Action refers to a user input or any event that causes state changes.

2.2 Component State

In a broader context, "state" refers to any meaningful data within a running application. However, when focusing on the component level, this definition can be narrowed down into two distinct categories: local and global state [20].

Local state, also called internal state, is local by how it is used and not intended to be accessed by other components. An example of a local state could be an object responsible for tracking the expanded status of a menu in a navigation bar. On the contrary, global state refers to data that needs to be accessible and shared across multiple components, such as user preferences, login information, or the contents of a shopping cart in an e-commerce application.

While management of the local state is easier and more predictable, global state management could lead to performance and maintenance issues depending on the complexity of an app and methods to manage the state.

2.3 State Management Problem

Modular architecture of SPA, comprises hundreds, if not thousands, of components depending on the size and complexity of the application. As components nested each other, they form a hierarchical tree-like struc-

ture known as the component tree. Understanding the component tree is essential for how data is shared and flows between components. [22]

In React, data flows uni-directionally from parent components to their child components. Child components are unable to directly alter the original data provided by their parent components or send data back to them. While this uni-directional data flow improves predictability of state, it can restrict data exchange between components. [20].

Consider a scenario where two components on the same level always require synchronized updates. Since data flow is uni-directional, these components cannot directly access each other's state. One possible solution to this problem is moving the state to the closest common ancestor of the components, and then passing it down as a property. This approach is known as a "lifting state up" [20]. Overusing this approach may result in an accumulation of state near the root, potentially leading to increased component re-renders.

2.4 Component Rerenders

React framework re-renders the child component whenever its parent re-renders. As a result, whenever the state changes, it causes the entire subtree to be re-rendered, and it could lead to poor user experience and performance issues.

Rendering performance is influenced by the number of components rendered. To decrease the number of rendered components, memoization techniques, such as useMemo and memo, can be used. useMemo and memo enable caching the result of a calculation between re-renders and increase the render performance by skipping unnecessary renderings [8]. Even though these techniques work well in theory, they introduce another concern about where these memoizations should be placed in the component tree. [17].

In the scenario above lifting the state one level up solves the problem but in a different scenario, in which synchronized components were in a different levels of a component tree and the nearest common ancestor was in several levels above, this could lead to another issue known as "prop drilling".

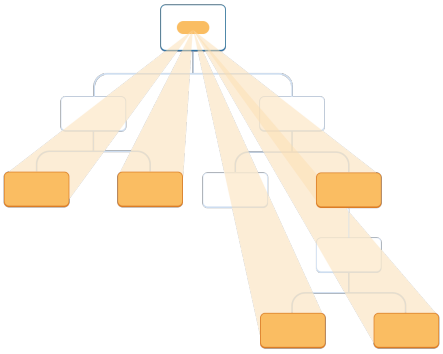


Figure 2. Using context in distant children [https://react.dev/learn/passing-data-deeply-with-context]

2.5 Prop Drilling

Prop Drilling refers to passing data deeply through several child components. The name "drilling" refers to forcing intermediate components to pass a property, that intermediate components are not interested in until it reaches to intended component.

As the application evolves and the component tree grows, prop drilling makes it difficult to reason about the application state. It causes the state to be passed through multiple layers, and, more importantly, results in state mutations scattered throughout the component hierarchy. This not only complicates the understanding of the application's state but also makes tasks, such as refactoring the state structure and debugging, significantly more difficult for application developers.[10].

While prop drilling may suffice for smaller applications or simpler use cases, developers tend to seek alternative solutions, particularly in more complex applications. One such solution is the usage of React's built-in Context API [12].

2.6 Context API: Passing Data Deeply In The Component Tree

Context API enables parent components to provide data to the entire tree below it without passing it through multiple layers of components.[19]. This approach allows only the consumers of context to be re-rendered when the parent component, also known as the provider, is re-rendered.

However, despite its convenience, there are certain considerations associated with the Context approach. Eventually, developers tend to put more value into the context [11]. Since it is not possible to perform granular updates on context-provided values by default, it can also lead to non-optimal re-renders [18]. Single field change in provided value, re-renders

all of the consumers even if they are not dependent on it. One solution for this issue could be splitting value into multiple "micro-contexts" and placing consumers under their dependent micro-contexts. Even though splitting context could solve the rendering problems, it introduces a lot of layers in the component tree and forces specific positions for future components in the tree depending on the business logic [17].

To address these challenges, developers often turn to state management solutions. These solutions provide structured methodologies for managing application states, offering centralized stores for data, and facilitating optimized re-renders. State management solutions promote cleaner, more maintainable codebases and streamline the development process by decoupling state management from the component tree.

3 State Management Solutions

Over the years, various approaches to state management have emerged, each offering distinct methodologies and solutions to address the complexities of managing application state. This section introduces the most popular and common state management solutions in the React ecosystem.

Understanding the principles and trade-offs of each approach enables developers to make informed decisions when selecting the most appropriate state management solution for their projects, ultimately leading to more scalable, maintainable, and responsive applications.

3.1 Redux

Redux is one of the most widely used [1] implementations of the Flux architecture, offering a predictable state container for JavaScript applications, particularly within the React ecosystem.

The core idea behind Redux is to introduce a single centralized location for managing the global state within an application and to follow specific patterns when updating the state. Redux is structured around these four fundamental concepts: [3]

Action refers to an event that describes something that has happened in the application. Each action includes a mandatory string "type" property that provides a descriptive name for the action.

Reducer behaves as an event listener and handles events based on the action type. It determines how state transitions should occur by re-

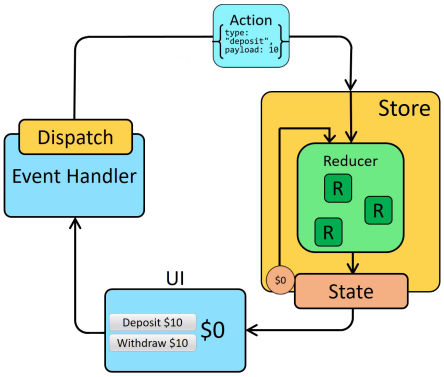


Figure 3. Using context in distant children [https://react.dev/learn/passing-data-deeply-with-context]

ceiving the current state and the action object as arguments and determines whether to update the state based on the action's type field.

Store is an object where the application state lives. The store is created by passing in a reducer function.

Dispatching an action is the only way to update the state in Redux. The store contains a method called **dispatch** which takes action object as an argument. Dispatching action could be thought of as "triggering an event" to inform the store about a specific action. Reducers within the store then listen for these events and update the state accordingly based on the action type.

Figure 3 shows typical data flow in the Redux library. Redux uses a uni-directional data flow in the given order. First, the UI dispatches an action. Then, the store runs the reducers, and the state is updated based on action. Finally, the store notifies the UI that the state has changed, and the UI re-renders based on the new state.

3.2 Zustand

Zustand is a small, scalable hook based state management solution. It is designed to be free from boilerplate codes [9].

The store in Zustand represents the central source of truth for the application state. It encapsulates the current state data and provides functions for updating the state. Creating a store typically involves using Zustand's **create()** function, which initializes the store with an initial state and returns functions for reading and updating the state. Zustand store is a hook and can be used anywhere in the application to manage the global state without any provider or wrapper.

Zustand uses an immutable state model where modifying state object

properties is prohibited. Instead, state updates must be achieved by creating new objects. State update causes re-render on consumer components. Since Zustand does not automatically track which fields are being used in the component, render optimization is achieved manually. [5]

3.3 Recoil

Recoil is an experimental, atom-based state management library developed by a team at Facebook. Recoil has two core concepts: atoms and selectors [23].

An **atom** refers to a piece of state that can be read and updated from any component in the application. Each atom requires an application-wide unique key (a string), and an optional default value. This key property is passed as an argument to a special function called **useRecoilState()**, to read from and write to the atom. When an atom is updated in one component, each subscribed component is re-rendered with the updated value.

A **selector** accepts other atoms and selectors as inputs, and whenever the value of a passed atom or selector changes, the selector's value is recomputed. Components can subscribe to selectors just as they do to atoms, and changes in selectors trigger re-renders in subscribed components.

3.4 Jotai

Jotai is an atom based state management library inspired by Recoil. It offers global state management with a simplicity similar to React's local state management hook, useState [14]. Jotai has two types of atoms: primitive atoms and derived atoms

Primitive atom refers to small, isolated piece of state. Serve as a building block for application state

Creating small atoms can lead too many atoms to organize. To avoid this, Jotai enables creation of **derived atoms** from existing atoms.

Jotai takes a bottom-up approach[16], in contrast to the monolithic big store model. Developers gain precise control over their application's state structure by creating small atoms and gradually combining them to form larger atoms. This method allows re-render optimization by adding only necessary atoms that will be used in components.

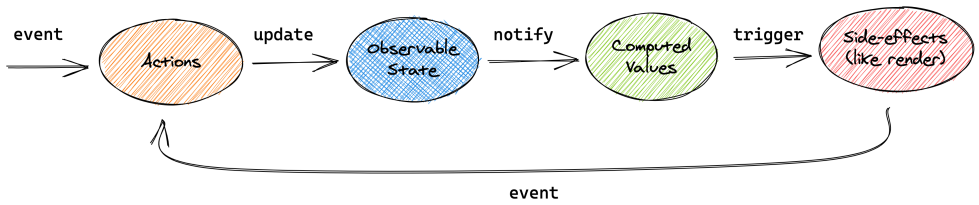


Figure 4. MobX State Flow[https://mobx.js.org]

3.5 Valtio

Valtio is a proxy-based, minimal state management solution. It uses JavaScript's native concept of proxies to create an observable state. Valtio's proxy transforms the object passed to it into a self-aware proxy, enabling automatic tracking of changes to the state [2]. Valtio uses an immutable state model and does not force any structural behavior, such as action or dispatch to update the state. This approach allows developers to update the state in a straightforward manner, similar to working with regular JavaScript objects [15].

3.6 MobX

MobX is a standalone state management library. The core idea behind MobX is to prevent producing an inconsistent state. This is accomplished through a straightforward strategy: "Make sure that everything that can be derived from the source state, will be derived. Automatically." [7]. MobX is structured around four fundamental concepts:

State refers to any value that can be mutated and serves as a source to derivations

Action represents functions that mutate the state

Derivations are automatically computed values from the source state. It could be primitive data, for example, the count of items in an array or more complex visual elements such as an HTML view. [26]

Reactions execute automatic task at the right time after state changes. For instance, updating the React component tree in response to a state change.

MobX uses a uni-directional data flow, where actions mutate the source state, which in turn causes all derivations to be automatically and synchronously updated. As a result, derivations never become stale, and the side effects are immediately visible to any observer. [6]

Table 1 provides key characteristics of various state management solutions discussed in this section.

Library	Architecture	State Model	Learning Curve	Github Stars (Apr 3,2024)
Redux	Flux Based	Immutable	Challenging	60401
Zustand	Hook Based	Immutable	Easy	41815
Recoil	Atom Based	Immutable	Moderate	19427
Jotai	Atom Based	Immutable	Easy	17066
Valtio	Proxy Based	Mutable	Easy	8352
MobX	Observable Based	Mutable	Moderate	27159

Table 1. Comparison of State Management Solutions in React Applications

4 Conclusion

As applications become more complex, effective state management becomes paramount for ensuring optimal performance and user experience. The React ecosystem offers various state management solutions to address these challenges.

This paper examines various state management strategies and libraries within the React ecosystem. Each state management library or framework comes with its unique set of characteristics, advantages, and limitations. From the simplicity and ease of use offered by React's built-in useState and useContext hooks, to the more elaborate and feature-rich solutions such as Redux,Zustand,Jotai,Recoil,Valtio and MobX, developers are presented with a spectrum of choices. The decision to adopt a particular state management strategy should, therefore, be guided by a thorough understanding of the application's specific requirements, scalability needs, and the development team's proficiency.

References

[1] Compare npm packages. https://npm-compare.com/@reduxjs/toolkit,zustand,recoil,jotai,valtio/#timeRange=SIX_MONTH. [Accessed 26-03-2024].

[2] Getting started - valtio, makes proxy-state simple for react and vanilla. https://valtio.pmnd.rs/docs/introduction/getting-started. [Accessed 01-04-2024].

[3] Redux fundamentals, part 2: Concepts and data flow. redux. https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow. [Accessed 26-03-2024].

[4] What are single page applications? what is their impact on users' experience and development process. https://www.netguru.com/blog/what-are-single-page-applications. [Accessed 01-04-2024].

[5] Working with zustand, tkdodos blog. https://tkdodo.eu/blog/working-with-zustand. [Accessed 30-03-2024].

[6] MobX Docs. The gist of mobx. https://mobx.js.org/the-gist-of-mobx.html. [Accessed 24-03-2024].

[7] MobX Docs. Mobx: Getting started. https://mobx.js.org/getting-started. [Accessed 24-03-2024].

[8] React Docs. memo. https://react.dev/reference/react/memo#skipping-re-rendering-when-props-are-unchanged. [Accessed 01-04-2024].

[9] Zustand Documentation. Introduction. https://docs.pmnd.rs/zustand/getting-started/introduction. [Accessed 30-03-2024].

[10] Kent C. Dodds. Prop drilling. https://kentcdodds.com/blog/prop-drilling, 2024. [Accessed 03-03-2024].

[11] Mark Erikson. Blogged answers: Why react context is not a "state management" tool (and why it doesn't replace redux). https://blog.isquaredsoftware.com/2021/01/context-redux-differences/#why-context-is-not-state-management. [Accessed 01-04-2024].

[12] David Herbert. A better way of solving prop drilling in react apps. https://blog.logrocket.com/solving-prop-drilling-react-apps/. [Accessed 24-03-2024].

[13] iBrandStudio. React js: The impact of react on web development. https://react.dev/learn/state-a-components-memory, 2023. [Accessed 24-03-2024].

[14] Jotai. Introudction: Jotai. https://jotai.org. [Accessed 30-03-2024].

[15] Daishi Kato. How valtio proxy state works (react part). https://blog.axlight.com/posts/how-valtio-proxy-state-works-react-part/. [Accessed 01-04-2024].

[16] Daishi Kato. Micro state management with react hooks. https://learning.oreilly.com/library/view/micro-state-management/9781801812375/, February 2022.

[17] Jason Miller Marvin Hagemeister. Introducing signals. https://preactjs.com/blog/introducing-signals/, 2024. [Accessed 26-03-2024].

[18] Ibrahima Ndaw. Pitfalls of overusing react context. https://blog.logrocket.com/pitfalls-of-overusing-react-context/. [Accessed 24-03-2024].

[19] React. Passing data deeply with context — react.dev. https://react.dev/learn/passing-data-deeply-with-context, 2024. [Accessed 26-03-2024].

[20] React. Sharing state between components. https://react.dev/learn/sharing-state-between-components, 2024. [Accessed 04-03-2024].

[21] React. State: A Component's Memory – React — react.dev. https://react.dev/learn/state-a-components-memory, 2024. [Accessed 07-02-2024].

[22] React. Understanding your ui as a tree. https://react.dev/learn/understanding-your-ui-as-a-tree, 2024. [Accessed 04-03-2024].

[23] Recoil RSS. Core concepts: Recoil. https://recoiljs.org/docs/introduction/core-concepts. [Accessed 30-03-2024].

[24] Technopalette Solutions. The power of state: A deep dive into state management. https://www.linkedin.com/pulse/power-state-deep-dive-management-technopalette-solutions/. [Accessed 07-02-2024].

[25] Lorenzo Ventura. Analysis of Redux, MobX and BLoC and how they solve the state management problem. Master's thesis, Politecnico Milano 1863, 2021.

[26] Michel Weststrate. The fundamental principles behind mobx. https://hackernoon.com/the-fundamental-principles-behind-mobx-7a725f71f3e8. [Accessed 24-03-2024].

Mental Health Disclosures on Social Media Platforms

Ghazal Shenavar
ghazal.shenavar@aalto.fi

Tutor: Yunhao Yuan

Abstract

Social media platforms have become hubs for individuals to openly discuss their mental health experiences and seek support. This paper provides an analysis of the evolving dynamics of mental health disclosures on social media, focusing on individual behavior implications. By reviewing recent literature spanning from 2021 to 2024, this paper categorizes studies based on their focus areas, motivations, and analytical approaches. This paper found that language analysis, sentiment analysis, and machine learning techniques are commonly employed to extract insights from social media data, with advanced models like Bidirectional Encoder Representations from Transformers (BERT) gaining prominence. However, researchers must struggle with privacy concerns and biases inherent in data collection methods. By addressing these challenges, we can better understand and address mental health issues in the digital age.

KEYWORDS: Literature Review, Mental Health, Social Media

1 Introduction

The number of social media users has had a steady increase over the past decade. As per Statista records, Facebook alone had more than 3 bil-

lion users in January of 2024, followed by almost 2.5 million users on YouTube [17]. These platforms have given voice to every person to discuss their narrative and share their own stories. Over time, these stories have evolved to include personal details [15]. Furthermore, these platforms have become a medium to discuss and raise awareness regarding issues related to mental health.

In these platforms as well, the stigma surrounding mental health issues persists, deterring many from seeking necessary treatment [5]. However, the anonymity afforded by social media platforms offers a potential remedy to this barrier, providing individuals with a platform to seek support and share experiences without fear of judgment [2]. Moreover, a variety of advocacy strategies (e.g, awareness-raising, promotion of diversity and inclusively) are employed by content creators and users to facilitate conversations on mental health issues [16].

Consequently, a wealth of information on mental health has emerged on social networking sites, offering valuable insights for various mental health studies and interventions. Previous research has utilized social media data for multiple purposes e.g. counselling methods [1] and social media redesign [18, 4]. The data is used to discern indicators of mental health issues, and how people in dire situations react to different conversations [1]. Previous research have used the available data in developing practical tools in clinical or Social Media contexts to help individuals in need [6, 1].

There have been notable studies [3, 14] exploring and classifying the work done regarding mental health surveillance in social media. This work attempts to update the aforementioned survey [3] by looking at papers published between 2021 to 2024, focusing on the implications of individual behaviour on social media for mental health.

The structure of the article is as follows. In section 2, the corpus is introduced and categorized based on year, mental health concern, motivation for study and the level it was conducted on. In section 3, the methods and results of the papers are discussed. Following in section 4, some privacy concerns and possible reasons for data bias are discussed. Finally, the paper concludes in section 5.

2 Methodology and Corpus overview

The process of gathering relevant papers involved two primary methods. Firstly, a targeted approach was employed, where one key paper served as an entry point to the subject [18], and additional papers were selected from its references. Secondly, to ensure comprehensive coverage, the keywords "Social Media" and "Mental Health" were utilized to retrieve pertinent literature from Google Scholar. Each search result was evaluated based on its title and abstract, with a focus on relevance to the study's objectives. While numerous papers addressing the impact of social media on mental health were identified, they were excluded from consideration, as they fell outside the scope of this research. Subsequently, the selected papers were categorized based on their year of publication, the specific mental health issue under investigation, and the underlying motivations driving the respective studies. The outcomes of these categorizations are elaborated upon in the ensuing sections.

2.1 Year of Publication

This study contains two papers from each of the years 2021 [8, 12], 2022 [13, 7], and 2023 [9, 10]. As the papers were collected primarily in the first two months of 2024, there weren't many papers available for the year 2024.

2.2 Mental Health Issue in Focus

The explored studies predominantly center around depression and suicide. Moreover, there was a notable exploration of how depression might lead to thoughts of suicide [8]. The exact issues are divided and can be viewed in figure 1.

Additionally, while depression took center stage, other mental health issues were indirectly addressed. For example, while examining depression severity in [10] and [7], there were indications of possible connections with eating disorders, although not explicitly stated as the focus of research. Similarly, the instances highlighted in various studies also touched upon stress, loneliness, and addiction, widening the array of mental health concerns investigated.

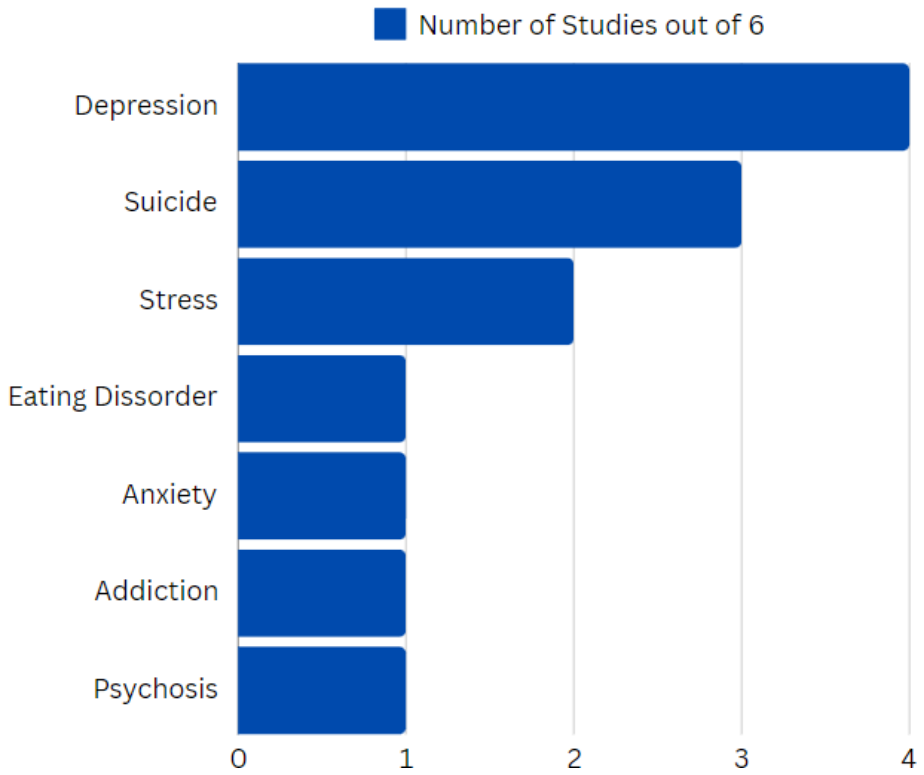


Figure 1. Number of studies per issue. In this graph, studies are divided by what issues they focus on. The most common concern is depression, followed by suicide.

2.3 Motivation for Study

This section aims to uncover the driving forces behind each study. Some studies, like [10], sought to build upon existing research by establishing a foundation and providing a dataset. Conversely, others such as [8, 13, 9], were interested in harnessing social media as a passive mental health sensor, offering potential aid to those in need. Another objective across studies like [8, 12, 7] was to leverage the vast pool of social media data to deepen our understanding of mental health issues and potentially apply findings in clinical settings.

For instance, [13] explored social media's role in predicting the necessity for quantity of clinical assistance in a college environment, proposing adjustments to resource allocation accordingly. Meanwhile, the study outlined in [12] discussed the ramifications of widespread media coverage on suicide-related content, highlighting the potential influences of particular posts and categorizing them based on the observed Papageno [1] and Werther effects [2].

[1] "The Papageno efect concerns how media can play a positive role in preventing and mitigating suicidal ideation and behaviors" [18]

[2] "The Werther effect is a phenomenon in which cases of suicide increase after

2.4 Level of Study

Studies can be categorized into three levels based on their scope of investigation: post-level, user-level, and population-level studies [3].

- Post-Level Studies: Post-level studies center around analyzing individual social media posts. Researchers examine the content, language, and sentiment of individual posts to identify indicators of mental health issues. While post-level studies offer granularity and specificity, they may overlook broader trends and patterns present across multiple users or posts.

- User-Level Studies: These studies focus on individual users within a social media platform. Researchers analyze the behaviors, interactions, and content shared by specific users to gain insights into their mental health status and patterns of engagement. User-level studies provide detailed, personalized data but may lack broader generalizability.

- Population-Level Studies: Population-level studies encompass broader analyses that consider the collective behavior and trends across a larger population of social media users. Researchers may employ statistical methods to identify patterns, correlations, and trends at the population level. These studies offer insights into the prevalence and distribution of mental health issues within a population but may lack the depth of understanding provided by user- or post-level analyses.

Among the reviewed studies, only [13] adopted a population-level approach. The remaining studies focused on developing datasets or methods, with a primary emphasis on analyzing individual posts rather than considering broader population trends.

3 Results

In this section, this paper discusses the methodologies used across various studies, including aspects such as data sourcing, ground truth establishment, variable selection, algorithm choice, and performance evaluation.

the publication of suicide news due to imitation." [11]

3.1 Source of Data

Given the primary emphasis of this investigation on language processing techniques for identifying mental health indicators, the majority of the data were sourced from two prominent social media platforms that focus on text-based communication: Reddit [8, 13, 9, 7] and Twitter (renamed to X in 2023) [12, 10].

For Reddit, posts were generally retrieved based on their association with relevant subreddits, while on Twitter, researchers employed predefined keywords to identify pertinent posts.

3.2 Ground truth

In [8], a mix of community associations and human annotation was used. Alternatively, In [9], an already labeled dataset was used. In other cases [13, 12, 7, 10], in addition to labeling depression severities in [8], human annotators were the ones establishing the ground truth. A common method was using multiple annotators to label one post and then deciding the confidence based on that (e.g. [10]). In some cases, posts which were annotated differently by more than one annotator were removed from the dataset; potentially simplifying the classification and removing confusing cases resulting in better results than what could be expected from unfiltered data.

All papers had no mention of obtaining control data. Rather, it seemed that they used the scraped data and posts labeled with not the issue by annotators as the control group.

As per twitter [12, 10], URLs and usernames were either removed from posts or posts mentioning them were removed overall. In [12], the intention was to avoid confusion because of words used in the URLs and usernames. For both, another consensus was to preserve the privacy of users and avoid traceability in the resulting database.

In [10], tweets with less than 8 words were also omitted as they were considered to contain too little information.

3.3 Variable selection

In [9], sentiment analysis was a component of the methodology, complementing other analytical techniques. Conversely, the remaining studies primarily centered around language analysis. Notably, [8] developed a

domain-specific lexicon tailored to the study's focus. On the contrary, [13] and [9] utilized a widely available general-purpose lexicon, namely LIWC (Linguistic Inquiry and Word Count), for their analyses. Additionally, [9, 12, 7, 10] adopted the BERT (Bidirectional Encoder Representations from Transformers) model, trained on general data, in their methodologies. Furthermore, topical analysis emerged as another pertinent methodology in [9].

3.4 Algorithm Selection

Many of the reviewed studies relied on the BERT (Bidirectional Encoder Representations from Transformers) model [9, 12, 7, 10] or variants derived from BERT, such as XLNet [12] and DistilBERT [10]. BERT is a powerful language representation model that has shown effectiveness in various natural language processing tasks due to its bidirectional architecture and large-scale pre-training on diverse datasets.

In addition to BERT-based models, Support Vector Machines (SVM) [8, 13, 12, 7, 10] and Long Short-Term Memory (LSTM) networks [8, 7] were commonly employed for evaluation alongside the aforementioned transformer techniques. BiLSTM (Bidirectional LSTM) architectures were specifically utilized in [7, 10] to enhance the learning capabilities of LSTM networks.

Furthermore, [12] employed a majority classifier as a baseline for comparison with other methods. Random Forest [8] and Logistic Regression [8, 7] were also explored as alternative classification approaches. Additionally, the use of Gated Recurrent Units (GRU) and Bidirectional GRU (BiGRU) was investigated in [7].

3.5 Performance Parameters and Validation Methods

The primary evaluation metrics utilized across the studies included F1 score and Accuracy, which were commonly employed for performance assessment [8, 9, 12, 7]. Additionally, Precision and Recall metrics were frequently utilized to gauge the model's ability to correctly identify positive instances while minimizing false positives [8, 9, 12]. Moreover, the Area Under the Curve (AUC) metric was utilized by [8, 10] to evaluate the model's ability to discriminate between positive and negative instances.

While F1 score, Accuracy, Precision, Recall, and AUC are widely recognized metrics for evaluating classification performance, it's essential to

note that other metrics were also explored by certain studies. For instance, [9] employed Expected Calibration Error and Adaptive Calibration Error to quantify the impact of calibration efforts on their models. Moreover, [13] utilized metrics such as Pearson correlation coefficient, mean absolute error, and the symmetric mean in their evaluation process.

Furthermore, cross-validation emerged as a prevalent validation method across the reviewed studies, with both 5-fold and 10-fold cross-validation being employed [8, 13, 9, 12, 10]. This methodological approach aids in assessing the model's performance across multiple subsets of the dataset, thereby enhancing the robustness and generalizability of the results.

4 Privacy Concerns and Data Bias

While the utilization of social media data for mental health studies presents promising opportunities, several privacy concerns and potential biases must be addressed. In the reviewed literature, measures were taken to mitigate these risks, albeit with varying degrees of thoroughness (e.g. no privacy concern was mentioned in [12]). Privacy measures used in the studied papers are listed below.

- Username Omission: Several studies [8, 12, 10] removed usernames or URLs from social media posts to safeguard user privacy. For instance, in [8, 12], usernames were systematically scrubbed from the data. Similarly, in [10], tweets mentioning usernames were entirely excluded from the dataset. Anonymization was a recurrent theme across multiple studies, reflecting a conscientious effort to protect user identities and maintain confidentiality.

- Selective Data Sharing: In [7], authors exercised caution by sharing the resultant database selectively and with the approval of an Institutional Review Board (IRB). This approach ensures that sensitive information is not indiscriminately disseminated and is consistent with ethical guidelines for research involving human subjects.

- Anonymity Preservation: In [13], anonymity was leveraged as a privacy-preserving mechanism. By allowing users to participate in discussions without revealing their identities, this study prioritized privacy concerns inherent in mental health-related discourse on social media plat-

forms.

Collection methods can introduce biases into the research findings. For instance, the reliance on social media platforms like Reddit and Twitter for data collection may inadvertently introduce selection bias. First, users who actively engage on these platforms may not be representative of the broader population, potentially skewing the data towards certain demographics or viewpoints. In addition, the use of specific keywords or subreddit associations to retrieve posts may further exacerbate this bias by favoring content that aligns with predetermined criteria.

Furthermore, the absence of a specific control group poses another potential source of bias in mental health research utilizing social media data. While some studies utilize posts labeled as not indicative of mental health issues as a control group, the lack of a well-defined control group introduces uncertainty regarding the comparability of the analyzed data. This absence may obscure the true impact of mental health indicators identified through social media analysis, as the absence of a control group hinders the ability to differentiate between normal behavior and behavior indicative of mental health concerns.

5 Conclusion

In conclusion, this study provides a comprehensive overview of the evolving landscape of mental health disclosures on social media platforms, focusing on individual behavior implications. The rise of social media usage has facilitated a significant shift in how individuals discuss and share their mental health experiences, offering both opportunities and challenges for mental health research and interventions.

Our analysis reveals a growing body of literature dedicated to exploring mental health dynamics on social media, with studies spanning various mental health issues such as depression, suicide, stress, loneliness, and addiction. These studies employ a range of methodologies, including language analysis, sentiment analysis, and machine learning techniques, to extract insights from social media data. Notably, the adoption of advanced machine learning models like BERT underscores the increasing sophistication of data analysis methods in this domain.

While the utilization of social media data offers valuable insights into mental health dynamics, researchers must remain vigilant regarding po-

tential biases introduced by collection methods and the absence of a specific control group. Addressing these concerns is crucial for ensuring the validity and generalizability of research findings in this domain. Moreover, efforts to address privacy concerns should remain a priority to uphold the integrity and ethicality of mental health research in this digital age.

In summary, this study contributes to the ongoing dialogue surrounding mental health on social media platforms, discussing emerging trends, challenges, and opportunities. By advancing our understanding of individual behavior implications, we can better inform mental health interventions, policies, and support systems in the digital era.

References

[1] Tim Althoff, Kevin Clark, and Jure Leskovec. Large-scale analysis of counseling conversations: An application of natural language processing to mental health. *Transactions of the Association for Computational Linguistics*, 4:463–476, 2016. doi: 10.1162/tacl_a_00111.

[2] Scott E. Caplan and Jacob S. Turner. Bringing theory to research on computer-mediated comforting communication. *Computers in Human Behavior*, 23(2):985–998, 2007. doi: 10.1016/j.chb.2005.08.003.

[3] Stevie Chancellor and Munmun De Choudhury. Methods in predictive techniques for mental health status on social media: a critical review. *npj Digital Medicine*, 3:43, 2020. doi: 10.1038/s41746-020-0233-7.

[4] Munmun De Choudhury, Sanket S. Sharma, Tomaz Logar, Wouter Eekhout, and René Clausen Nielsen. Gender and cross-cultural differences in social media disclosures of mental illness. *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2017. https://api.semanticscholar.org/CorpusID:19000482.

[5] Patrick Corrigan. How stigma interferes with mental health care. *American Psychologist*, 59(7):614–625, 2004. doi: 10.1037/0003-066X.59.7.614.

[6] Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 2098–2110, New York, NY, USA, 2016. Association for Computing Machinery. doi: 10.1145/2858036.2858207.

[7] Muskan Garg, Chandni Saxena, Veena Krishnan, Ruchi Joshi, Sriparna Saha, Vijay Mago, and Bonnie J. Dorr. CAMS: An annotated corpus for causal analysis of mental health issues in social media posts. 2022.

[8] Sanjana Garg, Jordan Taylor, Mai El Sherief, Erin Kasson, Talayeh Aledavood, Raven Riordan, Nina Kaiser, Patricia Cavazos-Rehg, and Munmun

De Choudhury. Detecting risk level in individuals misusing fentanyl utilizing posts from an online community on reddit. *Internet Interventions*, 26, December 2021. doi: 10.1016/j.invent.2021.100467.

[9] L. Ilias, S. Mouzakitis, and D. Askounis. Calibration of transformer-based models for identifying stress and depression in social media. *IEEE Transactions on Computational Social Systems*, 2023. doi: 10.1109/TCSS.2023.3283009.

[10] Mohsinul Kabir, Tasnim Ahmed, Md. Bakhtiar Hasan, Md Tahmid Rahman Laskar, Tarun Kumar Joarder, Hasan Mahmud, and Kamrul Hasan. DEPTWEET: A typology for social media texts to detect depression severities. *Computers in Human Behavior*, 139:107503, 2023. doi: 10.1016/j.chb.2022.107503.

[11] LH Kim, GM Lee, WR Lee, and et al. The werther effect following the suicides of three korean celebrities (2017–2018): an ecological time-series study. *BMC Public Health*, 23:1173, 2023. doi: 10.1186/s12889-023-16080-1.

[12] Hannah Metzler, Hubert Baginski, Thomas Niederkrotenthaler, and David Garcia. Detecting potentially harmful and protective suicide-related content on twitter: A machine learning approach. 2022.

[13] K. Saha, A. Yousuf, and R.L. et al. Boyd. Social media discussions predict mental health consultations on college campuses. *Sci Rep*, 12:123, 2022. doi: 10.1038/s41598-021-03423-4.

[14] Ruba Skaik and Diana Inkpen. Using social media for mental health surveillance: A review. *ACM Computing Surveys*, 53:1–31, 12 2020. doi: 10.1145/3422824.

[15] B. K. Smith, W. Bender, I. Endter, J. Driscoll, M. Turpeinen, and D. Quan. Silver stringers and junior journalists: Active information producers. *IBM Systems Journal*, 39(3.4):730–748, 2000. doi: 10.1147/sj.393.0730.

[16] Sarah Smith-Frigerio. Grassroots mental health groups' use of advocacy strategies in social media messaging. *Qualitative Health Research*, 30(14):2205–2216, 2020. doi: 10.1177/1049732320951532.

[17] Statista. Global social networks ranked by number of users. https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. Accessed: February 17, 2024.

[18] Yunhao Yuan, Koustuv Saha, Barbara Keller, Erkki Tapio Isometsä, and Talayeh Aledavood. Mental health coping stories on social media: A causal-inference study of papageno effect. In *ACM Web Conference 2023 - Proceedings of the World Wide Web Conference, WWW 2023*, page 2677–2685, United States, April 2023. ACM. doi: 10.1145/3543507.3583350.

Modelling cloud applications to achieve energy efficient scheduling

Gianni Canavero
gianni.canavero@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

Energy consumption regarding cloud computing is a popular topic given the impressive growth of data centre diffusion, and the industry focusing on sustainability. In this paper we analyse state-of-the-art research based on virtual machine consolidation, comparing the modelling of data centres and the scheduling algorithms adopted. We observe that the proposed papers base their models on CPU and memory analysis and focus on virtual machine consolidation.

KEYWORDS: cloud computing, VM consolidation, energy efficiency, energy modelling

1 Introduction

In 2022, the electrical demand for the global network of data centres was estimated to range from 240 to 340 TWh, corresponding to 1-1.3% of the global demand [12]. In the last two decades, data centres evolved from in-house solutions to cloud technologies, which surpassed 600 billion dollars in market share in 2024 [8]. These technologies implement the Infrastructure as a Service (IaaS) paradigm where companies can have access to storage and computing power without the need to manage or control the

underlying infrastructure [15]. To best utilize the computing infrastructure, powerful machines are split in parts reserved for different customers by virtualization technology.

These parts have variable computing power usage and are independent between themselves, cloud providers can distribute them on different machines according to their needs. To achieve a power efficient schedule for these parts, several modelling approaches have been built.

Until recently, cloud computing power consumption was not particularly relevant in research. Initially, priority was given to the quality of service by choosing the closest data centre or the fastest for the requested job to reduce latency. Then, the interest of cloud providers has moved towards the control of operating costs, which are mainly divided into computing power and cooling systems [5]; thus, research towards energy efficient solutions is rising in popularity.

This paper reviews recent research in predicting energy consumption of single applications, and how their scheduling on different machines can lead to the most efficient solution.

The paper is organized as follows. Section 2 contains the background knowledge needed to analyse the proposed solutions, section 3 explains the methods of the papers presented in the following section. Section 5 inspects the presented works and section 6 concludes the paper.

2 Background

2.1 Virtualization

Virtualization permits the sharing of a computing machine between different isolated environments. This can be achieved by the hypervisor, which interposes between the actual hardware and one or more operating systems guaranteeing them access to the computing power, as they were the only OS being executed on the machine. Another method is using a single operating system with a kernel that supports virtualization features through containers.

IBM [11] states "In a virtualized environment, computing environments can be dynamically created, expanded, shrunk or moved as demand varies", which has become the key to the success of cloud computing. Thanks to cloud technologies, the providers have the ability to divide

large machines into multiple entities, which can be rented to different customers and scaled in real-time following the demand.

Virtualization is used in the cloud environment mostly for the deployment of virtual machines (VMs) and containers. These two implementations differ from the presence of an operating system kernel in each application for the former, and the usage of a single shared kernel for all the applications in the latter. This difference affects access to the hardware, but it is not relevant in the modelling approaches considered in this paper, therefore, the following sections for modelling and scheduling applications consider both virtual machines and containers in a general case. The term VM will be used for both of them.

2.2 Data centres CPUs power consumption

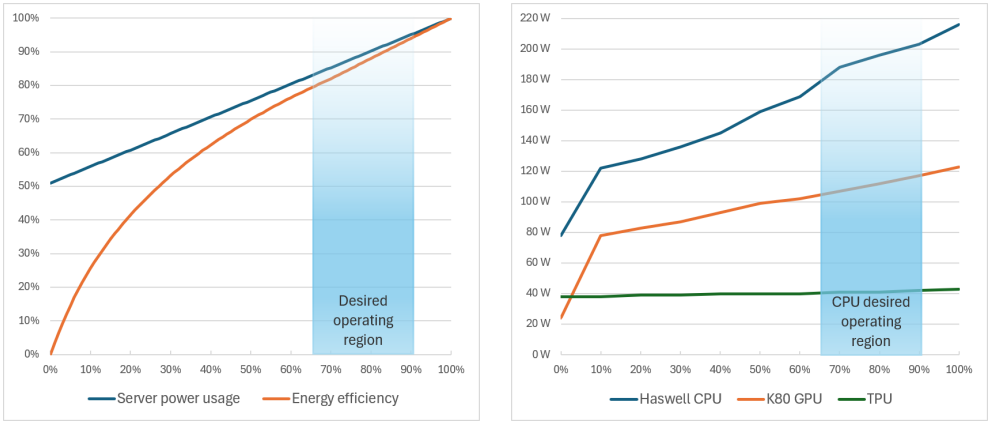


Figure 1. Barroso and Hölzle [2] on the left and Jouppi et al. [6] on the right.

The power consumption of a CPU depends greatly on its usage, generally in a data centres environment is modelled as follows:

$$P_{CPU} = P_{idle} + \sum P_{VM} \tag{1}$$

Where P_{idle} is the power consumption when the machine has only the operating system loaded, and P_{VM} is the power consumption of each loaded virtual machine. Barroso and Hölzle [2] have shown that CPU consumption in idle is close to half the consumption at maximum utilization. More recent studies have shown that Intel's Haswell CPU architecture consumes 56% of the maximum consumption at only 10% of the total load. For other components, such as GPUs (NVIDIA Tesla K80) and tensor processing units (TPU), energy consumption is even less reliant on system utilization [6]. Figure 1 illustrates their reports.

This data clarifies that the goal of VM scheduling in a power-oriented approach is to maximize machine utilization concentrating it on fewer machines and turning off the remaining ones to avoid idle consumption, this process is called virtual machines consolidation.

Consolidation leads to two major disadvantages: a highly utilized machine does not provide space to scale to VMs in case a spike in demand occurs; this compromises performance and high-priority processes will oblige the provider to move other applications to different machines. The second disadvantage is connected to the first, and it is the moving of VMs: this operation requires the application to be stopped for several seconds (even more if the destination machine is currently switched off) and may break the service level agreement (SLA) between provider and customer.

3 Methods

The analysis considers a data centre composed of a highly variable number of host machines. For a better representation of real data centres, host machines are different in their hardware and grouped into several generations: older machines are less efficient in terms of computing power for consumed electricity.

The following sections present three different scheduling solutions all based on VM consolidation, the objective is to concentrate the VMs in the minor number of host machines possible to switch off a major number of unused host machines. Therefore the algorithm has to model host machines resource usage to identify on which group of them is more efficient to consolidate the VMs. Also, individual virtual machines are modelled providing information to the algorithm for choosing which VM has to be moved, creating the final schedule.

The analysis focuses on the modelling and scheduling algorithms, comparing assumptions and different choices made to balance the benefits in terms of the saved energy and the drawbacks regarding computing power and quality of service.

4 Proposed papers

This section presents three approaches to modelling and scheduling cloud applications focusing on energy efficiency.

4.1 Utilizing power consumption and SLA violations using dynamic VM consolidation in cloud data centres

Arshad et al. [1] propose a classification of host machines in over-loaded (HOL), medium-loaded (HML) and under-loaded (HUL). The classification is done by measuring CPU and memory utilization and comparing them with two thresholds (T_{high} and T_{low}).

$$\text{If } CU \geq T_{high} \vee MU \geq T_{high} \rightarrow \textbf{HOL} \tag{2}$$

$$\text{If } T_{low} \leq CU \leq T_{high} \vee T_{low} \leq MU \leq T_{high} \rightarrow \textbf{HML} \tag{3}$$

Otherwise, the machine is reported as **HUL**, and VMs are consolidated by moving them from HOL and HUL machines towards HML ones.

In the case of a HOL machine the VM with the higher CPU utilization over memory utilization is moved, this helps both freeing more computing capacity and having a faster migration. In addition on overloaded hosts with high CPU usage, the algorithm will prefer moving a CPU-intensive VM, whereas for hosts with high memory usage memory-intensive VMs are selected. For underloaded hosts, every VM is moved.

Every time the solution migrates a VM, the destination host is the one using fewer resources at the moment (reported as less resource utilization LRU_h in the paper).

4.2 Enhancing Energy-Efficient and QoS Dynamic Virtual Machine Consolidation Method in Cloud Environment

Liu et al. [14] follow the idea of the precedent paper of merging VMs from overloaded and underloaded machines in the remaining ones. The paper uses a host overload detection algorithm based on the ARIMA model [3], which uses the historical CPU utilization of each host machine.

Host overload is defined as a workload close to the maximum capacity because a part of the computing power is reserved to prevent SLA violations. For each overloaded host, their solution assumes that a loss of computing power is guaranteed both in the case of moving the VM elsewhere (due to the moving) and keeping it on the same machine (due to the lack of computing power). Therefore, by analysing which of the two options is more convenient for every VM, a group of them is moved.

Finally, the algorithm computes the destination host machines. In order to prevent the overload of other machines, for each of them the program calculates the median absolute deviation of the median (MAD) of the

CPU usage to have a robust index of it [10]. Then starting from the most demanding VM to be moved, the algorithm selects the destination host machines by checking two conditions: the machine is not overloaded, the machine can host the VM and still has a defined amount of free capacity to support fluctuations of demand.

The underloaded machines are defined by two conditions: not hosting any migrated VM and has the highest h^{ea} defined below:

$$h^{ea} = \frac{P(h)}{h^u} \tag{4}$$

Where $P(h)$ is the total power consumption of the machine, and h^u is the CPU usage, thus the host with the highest ratio is the least power efficient. The VMs from the underloaded machine are moved according to the same algorithm used for the overloaded hosts.

4.3 Security supportive energy-aware scheduling and energy policies for cloud environments

Fernández-Cerero et al. [7] propose a scheduling approach which takes in consideration both power efficiency and security demands.

The workload is divided into batch tasks and service tasks, the former are independent between themselves and with a defined execution time, the latter are long-lasting services that compose the base load of each host machine in use and are not considered in the scheduling algorithm in order to avoid disruption of their service.

Each batch task is categorized by its security demands (sd), whereas the host machine is categorized by its trust level (tl) based on hardware security and software updates. Using this information, plus the task workload (wl) and the machine compute capacity (cc), the algorithm builds the $SBETC$ (Security Biased Expected Time to Compute) matrix. This matrix has an entry for each *(host machine, task)*-tuple and is built by the following:

$$\boldsymbol{SBETC}[j][i](SD, TL) = \frac{wl_j}{cc_i} + b(sd_j, wl_j, tl_i, cc_i) \tag{5}$$

where $j \in [\#\text{ of tasks}]$, $i \in [\#\text{ of host machines}]$, SD is the security demand vector, TL is the trust level vector, b is the security bias function that predicts the computing time needed by the security part of each task given the host machine it is running upon.

A genetic algorithm computes the scheduling, using the optimization formula described as follows:

$$C_{max} = \min_{S \in Schedules} \{\max_{j \in Tasks} C_j\} \tag{6}$$

where Schedules is the set of all the combinations of groups of tasks j, and C_j is the makespan of j-th task in the specific S schedule. The makespan takes into consideration which machine is running upon and the execution order.

The aim is to minimize the makespan of all the tasks.

A similar consideration is done for energy consumption, where the optimization is defined as:

$$\text{argmin}_{s \in Schedules} \sum_{i \in 1,\dots,m} \left(\sum_{j=1,\delta_{i,j}(s)=1}^{n} P^i_{busy} \left(\frac{wl_j}{cc_i} + b^i_j \right) + \sum_{j=1,\delta_{i,j}(s)=0}^{n} P^i_{idle} t^i_{idle} \right) \tag{7}$$

again for each possible schedule s we consider each machine $i \in [m]$ and each batch service $j \in [n]$, the function $\delta_{i,j}(s)$ returns 1 if the service j is scheduled to the machine i in s. Each machine has its P^i_{busy}, which denotes the power consumption when executing a task, and P^i_{idle}, which is the power consumption in idle. The time passed executing is given by the workload of the task over the computing capacity of the machine, the security needs of the task related to the trust level of the machine denoted by b are also considered. On the other hand, t^i_{idle} is given by the following formula:

$$t^i_{idle} = C_{max} - t^i_{busy} \tag{8}$$

C_{max} is the total computing time indicated by the chosen schedule s.

The paper shows four different approaches to manage the result of the algorithm run on the two optimization formulas: *makespan centric*, *energy centric*, *makespan centric until given threshold*, and *energy centric until given threshold*. The first two consider the best result in the centric feature and consider the other feature only in case of the same score. The last two consider all the schedules that pass the centric feature threshold and then analyse them by the other feature.

5 Analysis

The presented solutions aim all three towards the same solution: consolidate the VMs in a restricted number of host machines avoiding overloading and switching off the unused ones. The first two papers model

the data centre with a similar approach, by dividing machines into over-loaded, medium-loaded and under-loaded ones. The third paper has a different perspective on the modelling and adds an interesting view on the security side of the scheduling.

The cited works differ in the VM computation usage analysis, in detail: Liu et al. present the ARIMA model and review its robustness, on the other hand, Arshad et al. work with an easier model based on real-time CPU consumption. Fernández-Cerero et al. use a previous work without investigating its statistical properties [13].

In the scheduling part, various stratagems from the presented papers might be combined: Liu et al. approach of considering the most efficient option between moving a VM and keeping the host overloaded is an interesting feature to reduce the VM migrations. Arshad et al. formula for selecting VMs with a high ratio of CPU to memory usage helps to reduce the migration duration, while Fernández-Cerero et al. approach to security is relevant in categorizing host machines according to their trust level.

In addition, while working with overloaded machines, Arshad et al. differs from the other two papers. Their solution assumes the existence of VMs with high CPU or memory utilization in any case. Therefore, if an overloaded machine has a large number of VMs without any one of them exceeding the imposed thresholds, the algorithm will not change the state of the machine, leading to possible SLA violations.

The presented papers base themselves on the assumption that CPU and memory are the main contributors to the energy consumption of a cloud computing data centre. This strong assumption leads them to ignore other high-consuming hardware, such as tensor processors and GPUs highly used in artificial intelligence applications and HPC [4]. Another factor useful for the modelling might be I/O usage, especially in cloud applications working on a big data platform, where tools like Apache Hadoop or Apache Spark are I/O intensive.

Finally, the papers compare their scheduling approaches based on virtual machine consolidation with other methods, such as dynamic voltage and frequency scaling (DVFS). A heterogeneous algorithm that combines both methods may help in the case of an under-loaded machine running heavy VMs, where moving the VMs is not efficient. Recent literature has proposed similar algorithms [16, 9].

6 Conclusion

This paper compares three different approaches to model and schedule virtual machines in data centres to increase cloud computing energy efficiency.

We found out that several state-of-the-art procedures base themselves on VM consolidation, which guarantees higher energy efficiency but has the major drawback of VM migrations that may cause SLA violations. Therefore, the different presented works model the state of the data centre and manage migrations with different techniques and considering varying factors.

We also noted lacking features that may be considered in further research, such as not considering other hardware outside of CPU and memory, and focusing only on VM consolidation whereas other approaches could produce better results in selected cases.

References

[1] Umer Arshad, Muhammad Aleem, Gautam Srivastava, and Jerry Chun-Wei Lin. Utilizing power consumption and sla violations using dynamic vm consolidation in cloud data centers. *Renewable and Sustainable Energy Reviews*, 167:112782, 2022. https://doi.org/10.1016/j.rser.2022.112782.

[2] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40:33–37, 2007. https://doi.org/10.1109/MC.2007.443.

[3] Rodrigo N. Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE Transactions on Cloud Computing*, 3(4):449–458, 2015. https://doi.org/10.1109/TCC.2014.2350475.

[4] Pawel Czarnul, Jerzy Proficz, Adam Krzywaniak, and Jan Weglarz. Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments. *Sci. Program.*, 2019, jan 2019. https://doi.org/10.1155/2019/8348791.

[5] Khosrow Ebrahimi, Gerard F. Jones, and Amy S. Fleischer. A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities. *Renewable and Sustainable Energy Reviews*, 31:622–638, 2014. https://doi.org/10.1016/j.rser.2013.12.007.

[6] Norman P. Jouppi et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. 2017. https://doi.org/10.48550/arXiv.1704.04760.

[7] Damián Fernández-Cerero, Agnieszka Jakóbik, Daniel Grzonka, Joanna Kołodziej, and Alejandro Fernández-Montes. Security supportive energy-aware scheduling and energy policies for cloud environments.

Journal of Parallel and Distributed Computing, 119:191–202, 2018. https://doi.org/10.1016/j.jpdc.2018.04.015.

[8] Gartner. Public cloud services end-user spending worldwide from 2017 to 2024 (in billion U.S. dollars), 2024. https://www.statista.com/statistics/273818/global-revenue-generated-with-cloud-computing-since-2009/.

[9] Mirsaeid Hosseini Shirvani, Amir Masoud Rahmani, and Amir Sahafi. A survey study on virtual machine migration and server consolidation techniques in dvfs-enabled cloud datacenter: Taxonomy and challenges. *Journal of King Saud University - Computer and Information Sciences*, 32(3):267–286, 2020. https://doi.org/10.1016/j.jksuci.2018.07.001.

[10] P.J. Huber. *Robust Statistics*. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley, 2004. https://doi.org/10.1002/9780470434697.

[11] IBM. IBM white paper (2009) Seeding the Clouds: Key Infrastructure Elements for Cloud Computing. https://public.dhe.ibm.com/software/sg/cioleadershipexchange/seeding_the_cloud.pdf.

[12] IEA. Tracking Clean Energy Progress 2023. Technical report, International Energy Agency, 2023. https://www.iea.org/reports/tracking-clean-energy-progress-2023.

[13] Joanna Kołodziej. *Evolutionary hierarchical multi-criteria metaheuristics for scheduling in large-scale grid systems*, volume 419. Springer, 2012.

[14] Yaqiu Liu, Xinyue Sun, Wei Wei, and Weipeng Jing. Enhancing energy-efficient and qos dynamic virtual machine consolidation method in cloud environment. *IEEE Access*, 6:31224–31235, 2018. https://doi.org/10.1109/ACCESS.2018.2835670.

[15] Choi E. Lumb I. Rimal, B.P. *A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems*. Springer, 2010. https://doi.org/10.1007/978-1-84996-241-4_2.

[16] Boris Teabe, Alain Tchana, and Daniel Hagimont. Enforcing cpu allocation in a heterogeneous iaas. *Future Generation Computer Systems*, 53:1–12, 2015. https://doi.org/10.1016/j.future.2015.05.013.

Signals - new standard for state management in the web?

Guting Huang
guting.huang@aalto.fi

Tutor: Juho Vepsäläinen

Abstract

As web applications grow in complexity, effective state management becomes crucial for providing seamless user experiences. The reactive programming paradigm triggered the emergence of various JavaScript reactive libraries that react to changes efficiently. Among them, signals presents an efficient way to model reactivity that is distinct from previous more event-based solutions. This paper investigate the reactive abstractions and techniques of stream-based libraries that uses observables and signal-like libraries that implements signals and derived computations. The comparative analysis discusses design considerations, evaluation models, and methods for dependency tracking and data propagation. From this, we recognize that both stream-based and signal-like libraries serves differing purposes and each is useful for various aspects to state management.

KEYWORDS: signals, reactive JavaScript, state management, fine-grain reactivity, observables

1 Introduction

As modern web applications continue to grow in complexity and interactiveness, the demand for a seamless user experience and robust state management techniques has grown. Interactive web applications are often driven by events that the program must react to and coordinate asynchronously. Traditional methods handle events through callbacks, which are methods registered to the events and emit side effects. This approach is highly imperative and error-prone, placing heavy loads on the programmer to coordinate the interdependencies between the side effects. On the other hand, the increasing popular reactive programming (RP) paradigm is declarative and focuses more on "what" to do, leaving the "how" problems to the compiler and the framework [5].

Concepts of RP can be dated back to Fran [6], a language designed for constructing functional reactive animation and interactive media applications. In [6], *Behaviors* are time-varying values that continuously change, and *Events* are values that are connected to discrete points in time. The concept of continuous, time-varying values was carried onward to subsequent works and stimulated a number of JavaScript reactive programming solutions. These solutions provide either a dedicated language that later compiles to JavaScript or an extension with additional JavaScript constructs to represent the continuously changing events and behaviors [11].

Notably, *signals* has resurged since its initial introduction by KnockoutJS [14] and captured the interests of many recent reactive frameworks. Consequently, different variations have emerged revolving around this reactivity-primitive-based paradigm.

This paper summarizes the reactive primitives and state management techniques employed by modern signal-based libraries, aiming to provide insights into their approaches to achieving reactivity in comparison with previous reactive solutions. Section 2 introduces common problems and design considerations for reactive systems. Sections 4 and 3 describe two identified approaches to reactivity in the web. Section 5 compares the solutions, discussing each of their benefits and drawbacks. Section 6 concludes.

2 Design considerations

The survey consists of representative libraries selected based on their contribution in shaping JavaScript's reactive landscape and the availability of documentations and discussions within the development community involving the library authors. While these libraries share fundamental similarities, each exhibits nuanced differences in how they model reactivity. The following sections discuss two ways to modeling reactive behaviors and events: one utilizes signals and computations, while the other involves representing and operating on data streams. Before delving deeper into the specifics, first we explore their common characteristics and issues considered.

2.1 The observer pattern

The reviewed libraries implement variations of the observer pattern [8], which is a common design pattern in software development useful for managing one-to-many dependencies between entities where a state change affects the dependencies. The pattern is also known as the publish-subscribe pattern. This pattern was commonly implemented with abstractions to subjects and observers. Observers subscribe themselves to subjects, who notifies the observers in case of a change. The observers then request the updated state from the subject to update its own state.

2.2 Evaluation models

In general, there are two approaches adopted by reactive frameworks: push-based (event-driven) or pull-based (data-driven) [7]. In push-based evaluation model, when the parent is updated, new values are pushed to all its dependent decedents who will react to the changes. In other words, propagation of data is driven by events. On the other hand, a node in pull-based evaluation models pulls data from the source(s) in response to incoming requests. As an example, consider the following expressions:

```
x = 3
y = x * 2
z = x + 1
sum = y + z
```

Listing 1. Dependency example

In push-based model, `sum` recomputes whenever `x`, `y` or `z` gets updated. In pull-based model, `sum` recomputed when it is accessed (lazy evaluation).

A key challenge in the push-based model is the avoidance of wasteful computations in order to achieve more efficient state updates. In the example above, when `x` changed, `sum` might be recomputed twice as a response to change notifications from both `y` and `z`. In contrast, pull-based approaches update the values only when they actually change. However, extensive traversal and rerunning of the dependencies may be needed to stabilize the sources of change [4]. Thus, there may be a delay before results of an event are visible after the event's occurrence .

2.3 Glitch Avoidance

Glitch avoidance refers to the problem of observing an inconsistent state when the state is partially evaluated after a change. It may only occur to push-based models [1]. In Listing 1, a glitch may occur when `x` updates and `sum` responds to a change in `y` before `z` gets updated. The resulting `sum` does not reflect the expected outcome.

3 Signal-like libraries

Library	Signal	Derivation	Effect	Fine-grain
KnockoutJS [14]	observables	computed	-	yes
SolidJS [15]	Signals	Memo	Effects	yes
VueJS [18]	ref	computed	watchers	yes
Preact [12]	signal	computed signals	effect	yes
MobX [17]	observables	derivations	reactions / autoruns	yes
Svelte [10]	-	-	-	no
Qwik [2]	signals	computed	tasks	yes
Angular Signals [9]	signals	computed	effects	yes

Table 1. Terminology variations used by signal-like libraries for describing signal, derivations, and effects

Signals [14, 15, 18, 12, 17, 2, 9] are first-class reactive values that represent a piece of state or computation. Libraries designed around signals offer primitives to store and update reactive values, to derive and

cache computed values, and to create side-effects. The remaining of this article commonly refer to these fundamentals as signals, derivations, and effects, respectively. Table 1 summarizes the terminology variations for the primitives. Signals work by wrapping around values and providing a set of getter and setter functions that enables reading and writing to the signals. Values are passed to other dependencies by reference to the signal, and each signal holds a list of subscribers who will be notified of any change.

Libraries such as Vue [18], Solid [15], Preact [12], and Qwik [2], implement deep reactivity where elements inside nested objects and arrays are also made reactive and tracked automatically when creating a signal. On the other hand, shallow reactivity is when only the assigned value or property will be made reactive. While deep reactivity may be desired most of the time for its convenience, shallow reactivity can be useful for avoiding the observation cost of large objects [18].

Common characteristics to signal-like solutions include automatic dependency tracking, lazy evaluation, and synchronized change propagation. Next, we look into each of these characteristics in detail.

3.1 Dependency Tracking

Libraries such as Vue, Solid, Qwik make values reactive by intercepting property access using JavaScript's native Proxy API. As a result, reads and writes of signals will be noticed and used to create subscription tracking and effect triggering functionalities. On the other hand, Svelte [10] takes a different approach by tracking the subscriptions at compile time, ignoring any JavaScript syntax limitations.

Dependency tracking in the signal-like solutions facilitates a dynamic management of state dependencies that constructs the dependency graph during runtime. A global context stack is maintained to track any running derivations or reactions [3]. Derivations and computations are monitored by being pushed to the stack, and on execution, each accessed signal links itself as a dependency of the uppermost effect. When a signal updates, its subscribers recompute. In the end, we have a dynamic dependency graph that reconstructs each time during execution, possibly adding new nodes or removing inactive ones [3]. This mechanism leads to what is commonly referred to as *fine-grain reactivity*. Since the signals are binded to specific DOM elements, only those elements are updated on change.

A refined approach used by compiled libraries such as Svelte is to stat-

ically analyze the dependencies during build. This implies that the reactive behavior becomes inherently integrated into the final output, reducing potential runtime overheads.

3.2 Data Propagation

Data propagation is the process of updating the state in response to changes in the underlying data or dependencies. MobX [17] uses a push-based approach and defines a topological ordering of the nodes to be run synchronously, preventing stale derivations to be observed as described in Section 2.2. Since only relevant dependencies are tracked,

Other fine-grain reactive libraries such as Preact combine both pull and push models by pushing the change notifications but delaying the computation until the values are pulled. Preact signals keep a version number that updates on value change and is compared during effect runs to check if the sources have changed since their last update [16]. If the resources have not changed, then it avoids walking up the dependency graph. Solid follows a similarly approach with Preact except that it uses graph coloring instead of version numbers.

4 Stream-based libraries

Stream-based libraries follow the concepts of *Events* and *Behaviors* more closely through representations of data streams. For instance, RxJS [13] model the data streams as observable streams. It implements a variation of the observer pattern and has built-in primitives to represent subjects and observers.

In RxJS, observables serve as sources of asynchronous data streams, encapsulating sequences of values or events that may occur over time. These streams emit values consumed by subscribed observers and can be composed using operators including `map`, `filter`, and `reduce` [13]. Observables are push-based and lazy, meaning values are pushed to consumers only upon subscription, and both synchronous and asynchronous events may be emitted.

Observables [13] are push-based and lazy. Values emitted by the observable gets pushed to the consumers who react to the values. When multiple values are emitted, the values are not pushed until the consumer subscribes. Both synchronous and and asynchronous events may be emit-

ted.

Tracking dependencies between the subjects and observers is facilitated by the Subscription object, which monitors when an observer subscribes or unsubscribes from the observable. When an observable emits a value, this value is propagated downstream to all subscribers through the observable chain. Data is propagated synchronously, resembling the behavior of regular function calls. [13]

5 Discussion

Reactive libraries employ signals and observables for distinct purpose, each serving unique roles in managing data and facilitating reactivity. Observables primarily focus on modeling change propagation over time and can handle asynchronous operations. In contrast signals are designed for data storage and deriving data calculations.

One notable difference lies in the way they handle dependencies. Observables lack the dependency resolution required for glitch avoidance. On the other hand, signals are inherently aware of other signals, enabling dependency resolution through synchronous data propagation. This design facilitates optimizations not achievable with observables. By recognizing that the reactive dependency graph is oftentimes more shallow than the component graph, separate management of the dependency graph allows more performant updates in reaction to changes [16]. Updates are made only to relevant components dependent of the signal. Lastly, signal's synchronism simplifies stack tracing and the debugging process.

Despite their advantages, signals also present certain drawbacks. They emphasize data flow over control flow, which may pose challenges in JavaScript who is not a data flow language. Update only through value access means that it's important where you access the values, thus requiring a learning curve for programmers new to the concepts. Furthermore, overusing deep reactivity can lead the difficulties in knowing which values are reactive or not.

6 Conclusion

The resurgence of signals as an alternative paradigm to reactive state management underscores a shift from traditional event-based control flow to a data-centric flow. Signals, representing first-class reactive values, offer a comprehensive approach to managing state and derived computations. This paper examined the available solutions from four axes: the reactive abstractions, the evaluation model, the dependency tracking mechanism, and data propagation. We compared two common approaches to reactivity, discussing their nuances and trade-offs.

By adopting signals, developers can achieve fine-grained reactivity while mitigating the complexities associated with traditional event-driven models, achieving easier management of the data flow. Finally, we recognize that both stream-based and signal-like libraries serves differing purposes and each is useful for various aspects to state management. While signals allows us to express and more easily manage complex UI behaviors, observables will continue to be useful for user interaction and other asynchronous events.

References

[1] BAINOMUGISHA, E., CARRETON, A. L., CUTSEM, T. V., MOSTINCKX, S., AND MEUTER, W. D. A survey on reactive programming. *ACM Comput. Surv. 45*, 4 (aug 2013).

[2] BUILDER.IO. Qwik, Sept 2022. https://qwik.dev/.

[3] CARNIATO, R. Building a reactive library from scratch, Jan 2022.

[4] CARNIATO, R. Derivations in reactivity, Jan 2024. https://dev.to/this-is-learning/derivations-in-reactivity-4fo1.

[5] CZAPLICKI, E. Elm : Concurrent frp for functional guis. https://api.semanticscholar.org/CorpusID:11022636.

[6] ELLIOTT, C., AND HUDAK, P. Functional reactive animation. In *Proceedings of the Second ACM SIGPLAN International Conference on Functional Programming* (New York, NY, USA, 1997), ICFP '97, Association for Computing Machinery, p. 263–273. https://doi.og/10.1145/258948.258973.

[7] ELLIOTT, C. M. Push-pull functional reactive programming. In *Proceedings of the 2nd ACM SIGPLAN Symposium on Haskell* (New York, NY, USA, 2009), Haskell '09, Association for Computing Machinery, p. 25–36. https://doi.org/10.1145/1596638.1596643.

[8] GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Boston, 1994.

[9] GOOGLE INC. Angular signals, Jun 2023. https://angular.io/guide/signals.

[10] HARRIS, R. Svelte, Nov 2016. https://svelte.dev/.

[11] KAMBONA, K., BOIX, E. G., AND DE MEUTER, W. An evaluation of reactive programming and promises for structuring collaborative web applications. In *Proceedings of the 7th Workshop on Dynamic Languages and Applications* (New York, NY, USA, 2013), DYLA '13, Association for Computing Machinery.

[12] MILLER, J. Preact, Sept 2022. https://preactjs.com/.

[13] RXJS TEAM. Reactive extensions library for javascript, May 2013. https://rxjs.dev/guide/overview.

[14] SANDERSON, S. Knockoutjs, Jul 2010. https://knockoutjs.com/documentation/introduction.html.

[15] SOLIDJS TEAM. Solidjs, Apr 2018. https://www.solidjs.com/.

[16] VIIDE, J. Signal boosting, Sept 2022. https://preactjs.com/blog/signal-boosting/.

[17] WESTSTRATE, M. Mobx, Mar 2015. https://mobx.js.org/README.html.

[18] YOU, E. Reactivity in depth, Feb 2014. https://vuejs.org/.

Exploring contemporary user tracking methods in web services

Gyeha Lim
gyeha.lim@aalto.fi

Tutor: Tuomas Aura

Abstract

Since their inception in 1994, HTTP cookies have been pivotal in shaping the functionality of modern websites, providing convenience for users while enabling website owners and advertisers to track online behavior for targeted advertising. However, privacy concerns have influenced the tracking methods, particularly highlighted by the European Union (EU)'s General Data Protection Regulation (GDPR). This paper investigates user tracking methods without third-party cookies, focusing on exploiting first-party cookies and browser fingerprinting. Analyzing existing research reveals that first-party cookies are susceptible to leakage to third-party domains, enabling them to track user behavior. Additionally, browser fingerprint can be used to uniquely identify the user based on their browser and device characteristics. The browser fingerprint can be used both independently and in conjunction with cookies.

KEYWORDS: first-party cookies, third-party cookies, external cookies, fingerprinting

1 Introduction

The concept of cookies emerged in 1994, developed by Lou Montulli, a web browser programmer at Netscape Communications. Their initial purpose was to enable users of e-commerce platforms to keep a persistent shopping cart [5]. Since then, cookies have become integral to the features of modern websites, enhancing user experience, enabling personalized interactions, facilitating e-commerce transactions, and other functionalities such as authentication. While convenient, the cookies have also allowed website owners and advertising agencies to track online user behavior and exploit users' data for invasive advertising campaigns. This led to privacy concerns, especially in the EU countries. The GDPR, adopted in May 2018, requires organizations to gain consent from users to collect and process personal data via cookies. As a result, users in the EU have been asked to accept or reject cookies on websites. Some browsers have enforced limitations on the usage of third-party cookies due to these privacy concerns and regulations. However, advertisers have found new ways to track users through first-party cookies and browser fingerprinting. Henceforth, this paper will explore user tracking methods without third-party cookies. Section 2 provides a short overview of cookies and the user tracking network. Section 3 delves into using first-party cookies and browser fingerprint for user tracking.

2 Background

This section provides a concise overview of HTTP cookies. Additionally, it explores the intricacies of third-party tracking networks, elucidating how ad exchange services or ad networks orchestrate the placement of advertisements across domains by tracking user activities.

2.1 HTTP cookies

When a user visits a website, it stores cookies on the user's browser. These cookies are called *first-party* cookies. They have been used to enhance the user experience by storing preferences and maintaining user sessions. Modern websites embed third-party resources, such as images or scripts, from other domains to provide useful features to improve the user experience. When a user visits a website that embeds third-party resources,

the browser sends a request to the third-party domain, allowing them to set cookies on the browser while retrieving the required resources. These cookies are called *third-party* cookies. Whereas first-party cookies can track the user behavior only on the website the user is currently visiting, third-party cookies enable data collection beyond the scope of the visited website. Suppose the user continues to browse different websites that contain the resources from the same third-party domain. In that case, the browser will continue to send the same third-party cookies with each request, allowing the third-party domain to track the user's online activities and gather data across multiple websites. Thus, online advertisers and tracking applications such as Google Analytics have used third-party cookies to track users extensively across the internet [6].

2.2 Third-party tracking network

An ad exchange service is a digital marketplace where a website sells its ad space and an advertiser purchases it via a real-time auction [9]. Websites typically connect their ad inventory to ad exchange services through one or more ad networks. Advertisers, on the other hand, often leverage advertising agencies to participate in the bidding process for ad inventories. When a user visits a website that subscribes to an ad exchange service, third-party cookies from ad networks are set on the user's browser [9]. As the user browses other websites connected to the same ad networks, these networks can track the user across different websites. This information about the user is shared with the ad exchange service and the advertising agency so that the advertising agency would bid higher for a user more likely to be interested in its advertised product or service. Gomer et al. [9] referred to this network of websites, advertising agencies, and tracking agencies as *third-party tracking network*, as shown in Figure 1.

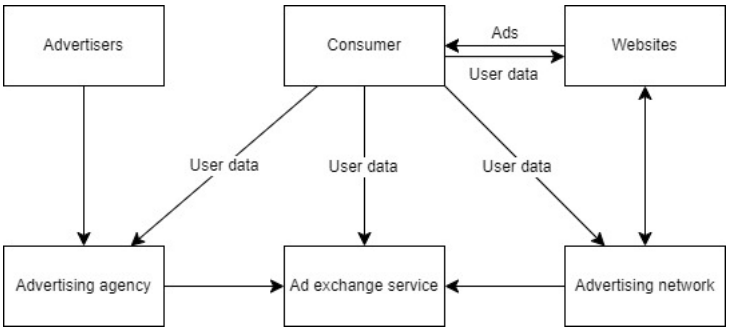


Figure 1. Third-party tracking network

2.3 Cookie synchronization

Advertising agencies and ad exchange services also leverage third-party cookies to identify the same user in the future by assigning distinct IDs within their domains [9]. Consequently, each domain knows the same user with a different ID. Cookie synchronization tackles this challenge by enabling different domains to synchronize their cookies and share the user's data [12]. Figure 2 illustrates how cookie synchronization works. Consider that third-party domains *tracker1.com* and *tracker2.com* set their cookies on the user's browser and assign user IDs *user123* and *userABC* to the same individual. When this user visits *website3.com* that only embeds resources from *tracker1.com*, *tracker2.com* remains unaware of the user's visit to *website3.com*. However, *tracker1.com* responds with an HTTP redirect that makes the user's browser send a GET request to *tracker2.com* specifying the ID of the user from *tracker1.com* so that *tracker2.com* can map this ID to its own, effectively linking the previously separated user profiles. Thus, cookie synchronization allows users to share their data with third parties that do not have their resources embedded in the website. This enriches their knowledge of the user from several data sources and helps them make better decisions during the real-time auction [12].

3 Tracking users without third-party cookies

In 2005, Apple Safari became the first browser to block third-party cookies by default with a privacy protection feature called Intelligent Tracking Protection (ITP) [3]. Google also recently announced that it will no longer support third-party cookies on Chrome [2]. Nevertheless, these measures have proven insufficient to prevent advertisers and trackers from tracking user activity. This section will review how advertisers and trackers could track users without third-party cookies.

3.1 First-party cookie leakage and ID synchronization

Modern websites have become intricate, often incorporating numerous external third-party sources. The Document Object Model (DOM), a programming interface for web documents, represents a page and allows it to be modified with a scripting language such as JavaScript [1]. Through the

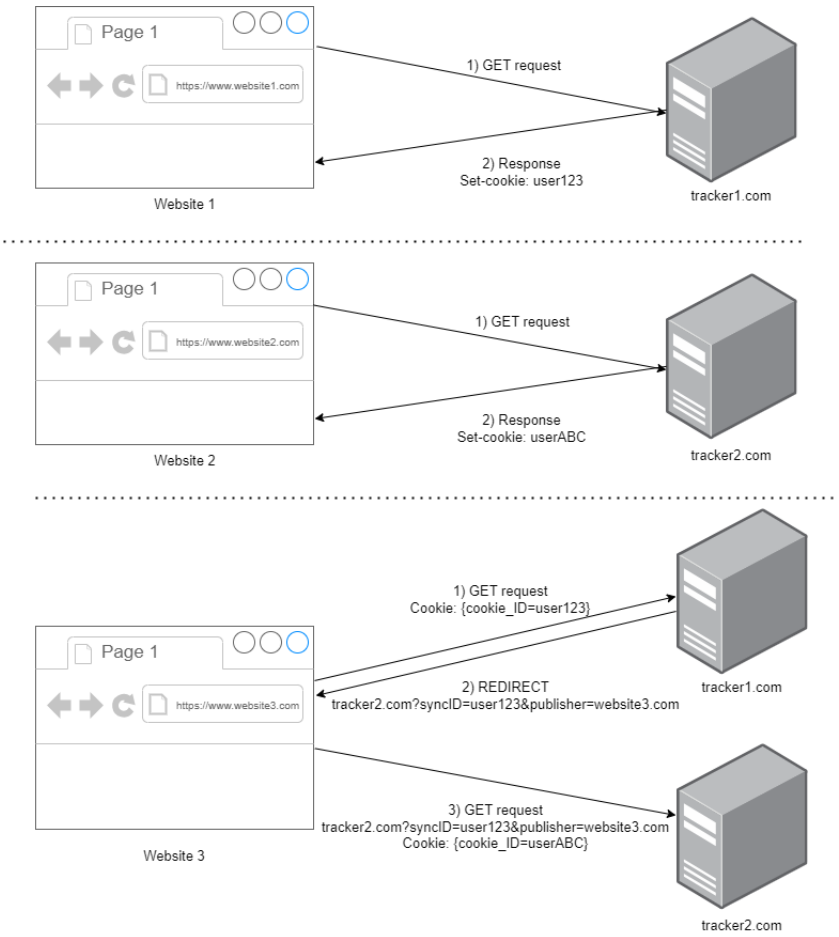


Figure 2. Example of cookie synchronization between trackers.

Document interface's cookie property, JavaScript code can read and write cookies associated with the page. Similarly, the embedded JavaScript code can set first-party cookies on behalf of the third-party domains. Chen et al. [7] call these first-party cookies set by the third parties *external cookies*. The script can return the value of the external cookie to the server from which the code originated by making an HTTP request. This technique has already been widely adopted, and according to Chen et al. [7], approximately 98% of the Alexa top 10k websites they crawled had external cookies.

Chen et al. [7] identified an additional vulnerability associated with external cookies. They revealed that scripts embedded within a web page can access and transmit external cookies set by other domains over the network. In other words, these cookies can be sent to a domain entirely different from the one initially created them. Chen et al. [7] analyzed the 13,323 unique non-session external cookies, discovering that 31.6% of them contained tracking IDs. Furthermore, they also found that 77.3% of these external cookies were leaked to third parties that did not set the

cookies originally [7]. Their research also identified the top 10 source domains whose cookies are shared with other third parties and the top 10 destination domains that receive external cookies from other third parties (different source domains). *google-analytics.com* shares its cookies with other third-party domains the most. The most shared cookie is *_ga*, shared in 3456 websites to 329 different third-party domains in total. On the other hand, *google-analytics.com* also received the highest number of external cookies from different source domains, with 427 cookies from 198 source domains. Their analysis shows that all third parties with scripts embedded in the page can access the external cookies set by other third-party cookies without any agreement, bypassing browsers' restrictions on third-party HTTP cookies.

Facebook is also one of the domains that most intensely attempts to retrieve and share users' information. Chen et al. [7] found that 2377 websites had external cookies set by *facebook.net*, named *_fbp*, which is shared to 76 different third-party destinations. Facebook employs pixels to track visitors' activities on a third domain [4]. A pixel is a JavaScript code containing a URL connecting to the Facebook server [4]. Upon the user's first visit to a website where the pixel is implemented, the pixel creates a first-party cookie called *_fbp* and sends the user's activities to Facebook during the user's visit to the domain [4]. Pixel performs event-driven tracking, with which the website can decide what event to track with standard events pre-configured by Facebook [4]. The event is sent to the Facebook server whenever the user triggers the event. According to Bekos et al. [4], 2308 out of the top 10k websites (approximately 23%) use Facebook Pixel to track their visitors in 2022.

Facebook can keep track of the user whether the user has a Facebook account or not. If the visitor to the third-party domain has a Facebook account, then Facebook can match the visitor with the account. When the user clicks an advertisement or a post of a third-party domain on Facebook, it sends an HTTP request to the third-party domain from where the advertisement or the post originated with a query parameter called *fbclid* [4]. The pixel on the third domain then creates a cookie named *_fbc* that saves the value of *fbclid*. Then, the pixel sends the values of *_fbc* and *_fbp* to the Facebook server, enabling Facebook to know that it is the user with a Facebook profile who visited this third-party domain [4]. Facebook gains further insights into its users through this synchronization. Facebook can uniquely track the user and the browsing history of the user on

the third-party domain. Even if Facebook does not know the identity of this visitor on the domain, for example, because the user does not have a Facebook Account, Facebook can still monitor this unknown visitor's activities on the site. Besides, upon the user's creation of a Facebook account, Facebook can connect the IDs and quickly learn the user's preferences and online browsing history. The first-party cookies set by Facebook have an expiration time of 90 days [4]. However, as long as the user continues to visit the website, then '_fbp' cookies can be renewed, enabling Facebook to track the user persistently [4].

3.2 Fingerprinting

As mentioned in the beginning of section 3, major web browsers have begun integrating built-in protection against cross-site scripting. However, this has raised concerns regarding trackers' potential shift toward stateless tracking techniques such as fingerprinting. Fingerprinting is a stateless tracking method that tracker can identify a user through a set of attributes obtained from the user's browser through JavaScript APIs and HTTP headers[10]. Indeed, many studies suggest that the deployment of fingerprinting across the web has been increasing over the years. A study in 2013 discovered that only 40 of the Alexa top 10K utilized fingerprinting codes [11]. However, a study in 2021 shows that 30.6% of the Alexa top-1K websites contain fingerprinting scripts, and 10.2% of the Alexa top-100K websites contain fingerprinting scripts [10]. These studies also indicate that the adoption of the fingerprinting script is more common on more popular websites. The author [10] also found that fingerprinting is especially prevalent on news sites, possibly due to the heavy reliance of news websites on advertising revenue.

Notable differences exist between fingerprinting and cookies based on their inherent characteristics. Cookies stored in a user's browser's local storage are readily visible to the user. In contrast, fingerprinting data lacks visibility within the user's browser environment. Furthermore, there is also a distinction in the level of user control over cookies versus fingerprint. Users can delete cookies, yet they cannot easily alter or erase their fingerprint.

As modern web browsers have continued to develop new features through JavaScript APIs, the chances of using those APIs to create unique fingerprint have also increased. For example, *Canvas*, *WebGL*, *fonts*, *extensions*, *plugins*, *audio* API, and even sensors can be used to identify the

device and build a browser fingerprint [10]. The author categorizes fingerprinting into two types: functionality fingerprinting and algorithmic fingerprinting. Functionality fingerprinting techniques probe for different functionality supported by the browser [10]. For example, the difference in permissions, peripherals, and APIs across browsers can be used as part of a fingerprint. Algorithmic fingerprinting techniques probe for functionality supported by the browser and process the inputs algorithmically using different JavaScript APIs [10]. The different implementations will process the inputs differently so that they can be used to create a fingerprint. For example, the *Performance* API can be utilized to calculate how long a particular function takes. The time would vary depending on browsers and underlying hardware settings. Besides, the trackers can also exploit animation, audio APIs, and the sensor to create an algorithmic fingerprint of the browser and client computer.

Lastly, the tracker can use fingerprint and cookies to respawn a cookie when the user deletes it from their browser [8]. It works by first creating a unique user identifier via fingerprinting and storing it in the cookies on the user's browser. If the user deletes the cookie, the tracker can respawn the same cookie based on the fingerprint. When a fingerprint changes over time, the tracker can use the old fingerprint stored in the cookie to link with the new fingerprint and update the cookie. Thus, a cookie can be respawned on the user's browser and exploited by the tracker.

4 Conclusion

Concerns surrounding user privacy have led to efforts to control cross-site tracking and limit the use of third-party cookies. While the phase-out of third-party cookies represents a significant step for user privacy, several researchers suggest alternative methods that could be exploited by the trackers for user identification and tracking across different websites. This study demonstrates two methods for tracking users without third-party cookies: first-party cookie leakage and fingerprinting. The trackers can leverage first-party cookies for data sharing with other domains. This could undermine the intended privacy benefit of restricting third-party cookies. The trackers can use browser fingerprinting techniques to generate a unique identifier for the user, raising significant privacy concerns. However, it is essential to note that there are mechanisms to find scripts that are used to generate fingerprinting, for example, by utilizing ma-

chine learning []. The limitation of this study is that it does not address the most recent strategies to counteract fingerprinting techniques.

References

[1] Introduction to the document object model. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Accessed: 2024-03-04.

[2] The next step toward phasing out third-party cookies in Chrome. https://blog.google/products/chrome/privacy-sandbox-tracking-protection/. Accessed: 2024-03-04.

[3] Safari privacy overview. Technical report, Apple, November 2019.

[4] Paschalis Bekos, Panagiotis Papadopoulos, Evangelos P Markatos, and Nicolas Kourtellis. The hitchhiker's guide to Facebook web tracking with invisible pixels and click ids. In *Proceedings of the ACM Web Conference 2023*, pages 2132–2143, 2023.

[5] Hal Berghel. Toxic cookies. volume 46, pages 104–107. IEEE Computer Society, 2013.

[6] Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. An empirical study of web cookies. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 891–901, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[7] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. Cookie swap party: Abusing first-party cookies for web tracking. In *Proceedings of the Web Conference 2021*, WWW '21, page 2117–2129, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Imane Fouad, Cristiana Santos, Arnaud Legout, and Nataliia Bielova. My Cookie is a phoenix: detection, measurement, and lawfulness of cookie respawning with browser fingerprinting. In *PETS 2022 - 22nd Privacy Enhancing Technologies Symposium*, Sydney, Australia, July 2022. Privacy Enhancing Technologies Symposium.

[9] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, and M.C. Schraefel. Network analysis of third party tracking: User exposure to tracking cookies through search. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 549–556, 2013.

[10] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1143–1161. IEEE, 2021.

[11] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy*, pages 541–555. IEEE, 2013.

[12] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *The World Wide Web Conference*, WWW '19, page 1432–1442, New York, NY, USA, 2019. Association for Computing Machinery.

Final paper for CS-E4000 Seminar: Neural video coding

Han Le
han.le@aalto.fi

Tutor: Matti Siekkinen

KEYWORDS: deep learning, neural networks, video coding, neural video coding

1 Introduction

Due to the surge in online video content, developing video compression systems capable of producing higher-quality frames within specific bandwidth budgets is essential. In recent years, deep neural network-based video coding (also known as neural video coding) has gained much research attention. Many studies have proposed new deep learning tools, showing that neural network-based video coding tools can outperform traditional and non-neural counterparts [1, 2, 3].

However, because of the fast development of neural network-based video coding technologies and the emergence of research publications on neural video codecs, an overview literature review of neural video coding and the recent advances in neural video coding is much appreciated. Therefore, this seminar paper aims to introduce the fundamentals of neural video coding, focusing on end-to-end neural video coding (i.e., deep schemes) instead of traditional video coding methods with deep learning-based components (i.e., deep tools). In addition, the paper presents two recent state-of-the-art studies of deep video coding.

This paper is organized as follows. Section 2 introduces the fundamentals of video coding and common metrics for comparison of different video

codecs. Section 3 gives an overview of neural video coding. Section 4 introduces two recent advanced neural video codecs, focusing on the quality perspective. Finally, Section 5 provides concluding remarks on the seminar paper.

2 Video coding

2.1 Video coding

Video coding is the foundational technology for computer vision and visual communication systems. According to Liu et al. [4], video coding typically refers to the technology that compresses videos into binary code, which is bits. This process aims to aid the storage and transmission of the visual video content. Video compression algorithms require an encoder to compress the video and a decoder to reconstruct the original video [5]. Together, the encoder and decoder form a codec.

Traditional video coding standards are developed on the predictive coding scheme (also known as P-frame coding), where a video frame (which is a picture) is decomposed into $m \times n$ pixels and the probability of each pixel is estimated one by one in the raster scan order. This strategy is illustrated in Figure 1.

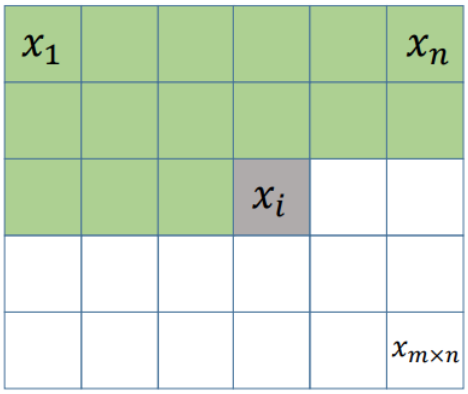


Figure 1. Illustration of a typical predictive coding scheme [4], where the pixels are encoded and decoded one by one. As an example, to predict the pixel value of the gray pixel x_i in the illustration above, all the previous pixels (in green) can be used as $condition$. The green area is called the *context* for pixel x_i.

In predictive coding, video codec predicts the relevant contexts from the previous reconstructed signals to reduce the spatial (intra-picture) and temporal (inter-picture) redundancy [2]. The compression efficiency of a codec relies on whether the current signal can find the relevant con-

texts from the previous reconstructed signals. Therefore, video codecs are designed to exploit both spatial and temporal redundancies within and across video frames to achieve high compression efficiency. A traditional video coding tool using this strategy is the High-Efficiency Video Coding (HEVC) Standard [6]. It is considered one of the standard non-neural video coding tools and is often used in performance benchmarking studies of neural networks-based counterparts [1, 2, 3].

According to Rippel et al. [3], typical predictive coding has two steps: (1) motion compensation and (2) residual compression. Motion compensation aims to reduce redundancy between consecutive video frames by accounting for the motion between frames, e.g., the motion of the objects and the camera. This process involves comparing blocks of target pixels in the current frame, for example, $\mathbf{x}_t$ for time step t, to those in previously transmitted reference frames and searching for the best match. The resulting motion vectors are then used to reconstruct the current video frame. After motion compensation, the difference between the original target frame and its motion-compensated approximation $\mathbf{m}_t$ is computed as $\Delta_t = \mathbf{x}_t - \mathbf{m}_t$. This difference is called the residual and is encoded using a compression algorithm tailored to the sparsity of the residual data.

2.2 Types of video compression

According to the review paper of Liu et al. [4], generally there are two types of video compression. The first type is lossless coding, where the goal of compression is the perfect reconstruction of video from the bits. The second type is lossy coding, where the compression does not aim at perfect video reconstruction. Why should we adopt such lossy compression strategy?

Efficient transmission of a compressed video over wired and wireless networks requires that the video is compressed at a ratio of hundreds to thousands compared to its original form. However, the compression ratios of existing lossless coding schemes are only approximately 1.5 to 3, which is way below the requirement ratio. Therefore, lossy coding is used to increase compression ratio but at the trade-off of incurring loss. The loss function is typically measured by mean squared error (MSE) to describe the difference between original and reconstructed images. The lower the loss is, the better. Then, the quality of the reconstructed image compared with the original image can be computed by the Peak Signal-to-

Noise Ratio (PSNR):

$$PSNR = 10 \log_{10} \frac{(max(\mathbb{D}))^2}{MSE}, \tag{1}$$

where $\mathbb{D}$ is the definition domain of a pixel and $max(\mathbb{D})$ is the maximal value in $\mathbb{D}$. As an example, for an 8-bit grayscale image, $\mathbb{D} = \{0, 1, \ldots, 255\}$, therefore $max(\mathbb{D}) = 255$. The greater the PSNR value, the higher the quality of the output image.

Liu et al. [4] suggest that to benchmark different lossless video coding schemes, comparing the compression ratio or the resulting coding rate (e.g., in bits per second) is enough. The lower the coding rate, the better—because a video codec aims to encode video data in the lowest number of bits possible. On the other hand, to benchmark different lossy coding schemes, both rate and quality need to be considered, because of the trade-off between compression ratios and error.

3 Introduction to neural video coding

Thanks to the rapid development of deep learning, neural network-based video codecs [7, 2] have shown comparable or better performance than standard, non-neural counterparts such as HEVC [6]. An advantage of deep neural networks is the data processing capacity with many levels of abstraction, and converting data into different kinds of representations [8]. As mentioned in the Introduction part, this paper focuses on deep coding schemes, which are novel coding schemes that are primarily built upon deep neural networks. In this section, we briefly introduce the auto-encoder (which is commonly used in deep coding schemes) and rate-distortion cost. The rate-distortion cost is an important metric, which is commonly used for the performance evaluation of video codecs.

Auto-encoder is a neural network architecture proposed by Hinton and Salakhutdinov [9]. Generally, the network has an encoding part and a decoding part and it is trained for dimensionality reduction. Fig. 2 presents the transform coding scheme, which relies on the architecture of auto-encoder.

In neural video compression, by considering the tradeoff between compression rate and quality, the neural network is trained to minimize the joint rate-distortion cost:

$$L = \lambda D + R, \tag{2}$$

where D is the difference (i.e., distortion) between the original image $\boldsymbol{x}$

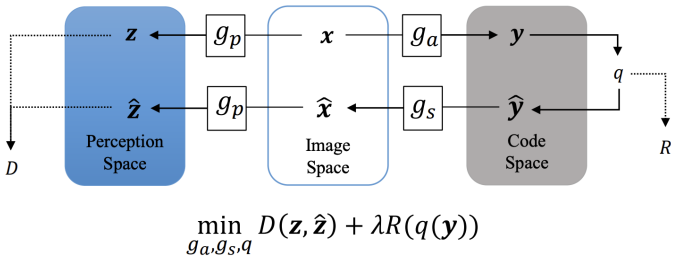


Figure 2. In a typical transform coding scheme [4], the original image $\boldsymbol{x}$ is transformed by an analysis function g_a to $y = g_a(\boldsymbol{x})$. The code $\boldsymbol{y}$ is quantized (denoted by q) and coded into bits. Here, the coding rate R is measured by the number of bits. The quantized code $\hat{\boldsymbol{y}}$ is then inversely transformed by a synthesis function g_s to achieve the reconstructed image $\hat{\boldsymbol{x}}$. Both $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are then transformed by a perceptual function g_p into $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$. The difference between $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$ is used to measure the distortion D.

and the transformed image $\hat{\boldsymbol{x}}$. D can be measured by a loss function such as MSE (mean squared error) or MS-SSIM (multiscale structural similarity) [10]. λ is the Lagrange multiplier which controls the trade-off between the distortion D and the coding rate R.

4 Recent advances in neural video coding

This section briefly introduces recent works on end-to-end deep learning-based video coding in the past five years. Specifically, we will get an overview of two paradigms of deep video coding schemes in recent years, namely residual coding and conditional coding. For residual coding, we use the study by Djelouah et al. [11] in 2019 as an example. For residual coding, we use the study by Djelouah et al. [12] in 2021 as an example. In the research publications using residual coding and conditional coding, each video codec is compared to other state-of-the-art neural and non-neural codecs at that time. For the comparison metric, the rate-distortion cost and PSNR are emphasized.

4.1 Residual coding

Residual coding generates a predicted frame in the pixel domain as the context and only uses subtraction to remove redundancy. A highlighted study in 2019 on residual coding is by Djelouah et al. [11], who propose an end-to-end deep learning codec that interpolates the predicted frame from previous frames and future frames, and then the frame residue is encoded. Three datasets are used for benchmarking: VTL, UVG [13], and MCL-JCV [14]. Both high-quality UVG (Ultra Video Group) dataset and

MCL-JCV dataset (which are common datasets used for comparing video codecs) have a resolution of 1920 $\times$ 1080 with a large variety of motion and content.

As a result [11] in Fig. 3, the rate-distortion performance of their neural video codec (marked as green line) on different datasets and resolutions is competitive with state-of-the-art video codecs by that time, including DVC [7] (which is an end-to-end deep video coding scheme that "deepens" the traditional video coding schemes).

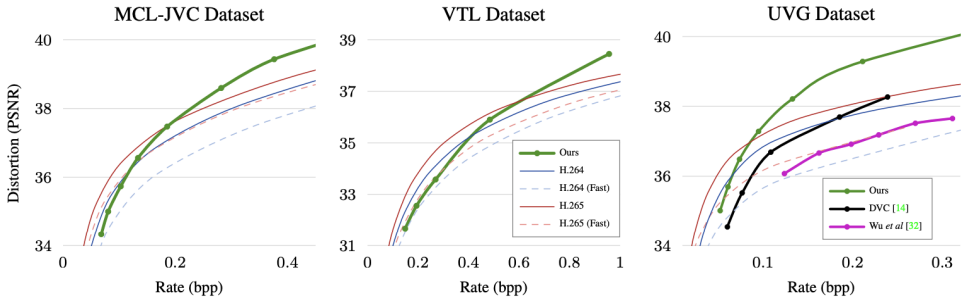


Figure 3. Compression results on three different video datasets using a key-frame interval of 12 frames [11].

4.2 Conditional coding

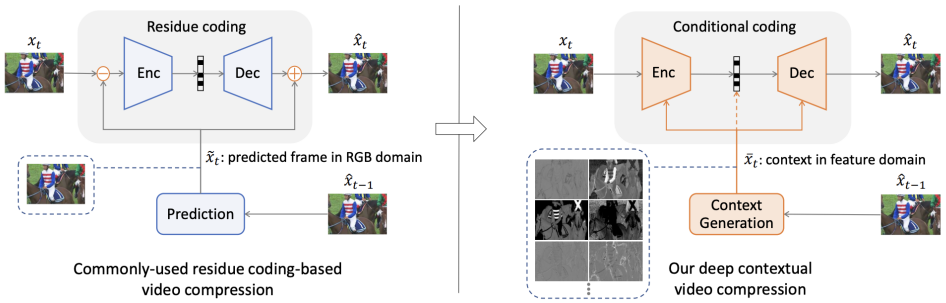


Figure 4. Schematic comparison between residual coding and conditional coding [12]. The current frame is marked as $\boldsymbol{x_t}$. $\hat{\boldsymbol{x}}_{\boldsymbol{t}}$ is the current decoded frame and $\hat{\boldsymbol{x}}_{\boldsymbol{t-1}}$ is the previous decoded frame.

The second category, conditional coding, implicitly learns feature domain contexts. The high-dimension contexts can carry richer information to facilitate encoding and decoding. Recent works have shown that conditional coding can outperform residual coding. Li et al. [12] proposed the DCVC framework (which is on the right side of Fig. 4) to learn feature domain contexts to increase context capacity. They define the condition as learnable contextual features with arbitrary dimensions that may be useful to compress the current frame. The contextual information is used

as part of the input of contextual encoder, contextual decoder, as well as the entropy modeling (marked in orange).

As a result [12] in Fig. 5, the rate-distortion of DVCVC (which is measured by PSNR) on different datasets and resolutions is competitive with state-of-the-art video codecs by that time, including DVC [7] and DVCPro.

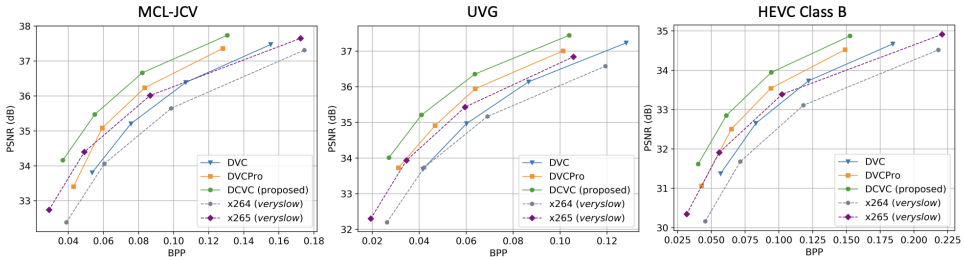


Figure 5. PSNR and bitrate comparison in [12]. The horizontal axis is bits per pixel (BPP) representing bitrate cost and vertical axis is PSNR representing reconstruction quality.

5 Conclusion and limitations

This paper has reviewed the basics of video coding and neural-based video coding and its advances in the past few years. Residue coding-based framework assumes the inter-frame prediction is always the most efficient, which might be inadequate, especially for encoding new contents. By contrast, conditional coding enables the adaptability between learning temporal correlation and learning spatial correlation. However, many neural codecs are still too computationally expensive to be applied to real-life settings, especially on mobile devices. Further research on practical neural codecs and benchmarking neural compression algorithms on-deviceare are needed.

References

[1] F. Mentzer, G. Toderici, D. Minnen, S.-J. Hwang, S. Caelles, M. Lucic, and E. Agustsson, "VCT: A video compression transformer," *arXiv preprint arXiv:2206.07307*, 2022.

[2] J. Li, B. Li, and Y. Lu, "Neural video compression with diverse contexts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22616–22626, 2023.

[3] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, "Learned video compression," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3454–3463, 2019.

[4] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–35, 2020.

[5] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 416–431, 2018.

[6] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[7] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11006–11015, 2019.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[11] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, "Neural inter-frame compression for video coding," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6421–6429, 2019.

[12] J. Li, B. Li, and Y. Lu, "Deep contextual video compression," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18114–18125, 2021.

[13] A. Mercat, M. Viitanen, and J. Vanne, "Uvg dataset: 50/120fps 4k sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 297–302, 2020.

[14] H. Wang, W. Gan, S. Hu, J. Y. Lin, L. Jin, L. Song, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, "MCL-JCV: a JND-based H.264/AVC video quality assessment dataset," in *2016 IEEE international conference on image processing (ICIP)*, pp. 1509–1513, IEEE, 2016.

User engagement in wearable technology

Hang Le
hang.1.le@aalto.fi

Tutor: Sanna Suoranta

Abstract

KEYWORDS: mobile health applications, user engagement, user motivation, wearable technology

1 Introduction

Currently, wearable technology has gained widespread popularity, becoming integral to various aspects of daily lives. The wearable market is projected to reach 160 billion dollars by 2026 [1]. While individuals initially embrace these devices with excitement, there is a recurring trend where this enthusiasm tends to wane over time. One study revealed that one-third of American users discontinue the use of wearable products within six months of acquiring them [3]. In an aging society, where health and fitness play crucial roles in maintaining well-being, understanding the factors that sustain user engagement in wearable technology becomes one of the key focuses.

As people age, the desire to stay physically active becomes increasingly important for overall health. Wearable devices can support users to monitor and encourage exercise routines. However, the challenge lies in using the elements that contribute to sustained engagement over the long

term. What motivates users to continue using these devices and adhering to their exercise routines?

This paper analyses existing research and experiments, leveraging data collected from wearable technology, to investigate the key components that keep users motivated and engaged. By examining various studies and real-world experiments, the goal is to identify successful strategies as well as potential shortcomings in current wearable technology approaches. Understanding these nuances is important for designing future iterations of devices and applications that not only capture users' initial excitement but also maintain their interest and commitment over extended periods.

This paper is organized as follows. Section 2 describes the concepts of user engagement and motivation within wearable technology. Furthermore, this section also examines some established frameworks and methodologies for quantifying user engagement. Section 3 discusses elements that have yielded positive outcomes in maintaining user motivation. Section 4 discusses elements that, on the other hand, discourage users from engaging with wearable technology. Section 5 provides a discussion that reflects the author's perspective on the subject of this paper as well as proposing potential directions for future research and development. Finally, Section 6 offers concluding remarks, summarizing the key findings in the field of wearable technology and user motivation.

2 Concepts of wearable technology and user engagement

Wearable technology has advanced significantly. This section explores the concepts of wearables in healthcare and user engagement. Additionally, it examines frameworks for evaluating user experience in wearable technology, highlighting the importance of user-centric design.

2.1 Wearable technology

The concept of wearable computers dates back to 1955 when Edward Thorp and Claude Shannon developed a device intended to forecast roulette results in Las Vegas[2]. Progress continued with experiments in smart glasses and helmets in the 1960s, assisting photographers in the 1970s, and developing augmented reality and smart shirts in the 1980s and 1990s [2].

In the context of health care, this paper focuses on wearables that allow users to collect and centralize all health and fitness data conveniently. Wearable technologies can be classified into two types [4]. Primary devices function autonomously and act as central hubs for connecting with other devices or accessing information. Examples include wrist-worn fitness trackers and smartphones. Secondary devices are intended to record specific actions or measurements, such as chest-worn heart rate monitors. These secondary devices then transmit the collected data to a primary wearable device for analysis [4].

2.2 User engagement

User engagement, according to Lalmas et al, [6] refers to the depth of a user's positive interaction with an online application, characterized by a willingness to continue using and return to the application. This concept is important in the design of online platforms across various devices since successful applications garner not just usage, but active engagement [6]. Engaged users devote their time, attention, and emotional investment, aiming to fulfill both practical and pleasurable needs [6].

2.3 Frameworks in measuring user experience in wearable technology

As illustrated by Kim et al [7], the framework for evaluation of wearable devices follows product design space and evaluation factors. As depicted in Figure 1, this framework evaluates the physical and functional aspects of wearable technology in the context of user type, device type, task type, and environment [7]. The emphasis here is on identifying and improving design spaces to align with user values and enhance usability, hedonic, and aesthetic qualities.

On the other hand, Asimakopoulos et al [8] examine the relationship between user motivation, self-efficacy, and user experience with fitness tracking mobile applications, using a diary study and validate questionnaires over a four-week period. Participants provide insights into their engagement with the applications, revealing key motivational factors and potential barriers to sustained use [8]. This framework employs a user-centric research method, involving diary studies and questionnaires to gauge self-efficacy and motivational factors affecting user engagement [8]. It highlights the importance of user feedback in understanding the effec-

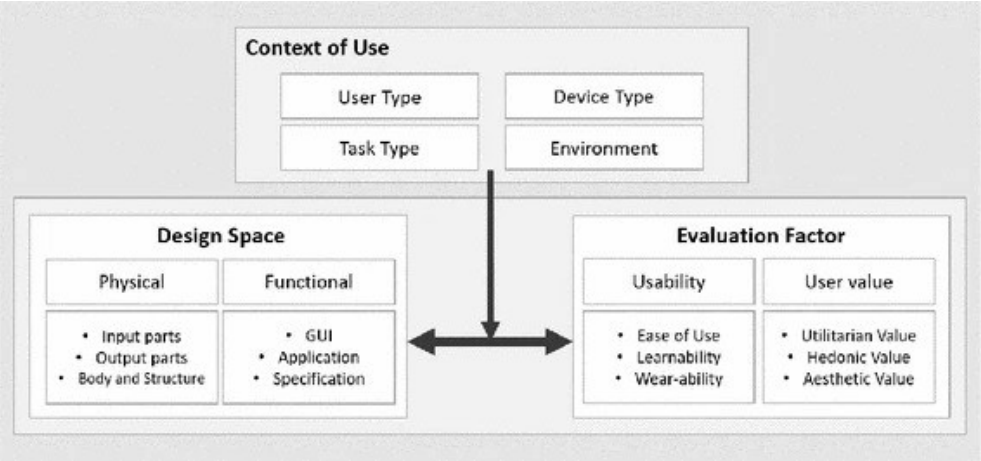


Figure 1. Evaluation framework for wearable devices [7]

tiveness of wearable technology and the factors that motivate long-term use.

Both of the aforementioned frameworks, while differing in methodology, underscore the necessity of integrating user feedback and design considerations to foster a more engaging and valuable user experience. This suggests that the focal point for measuring user engagement within wearable technology is not only the functionality of the device but also the user's interaction with the device, which is shaped by individual motivations, habits, and the context of use.

3 Elements that support user engagement in wearable technologies

The integration of wearable technologies in healthcare settings necessitates a deep understanding of the factors that sustain user engagement. This section describes some of the important elements that maintain user engagement in wearable technology within healthcare contexts. The discussed elements include clearly defined problem, empowerment through diagnosis and behavior change, and comfortable design.

3.1 Clearly Defined Problem

According to Smuck et al [9], one of the first focus for wearable technologies is addressing specific healthcare challenges to maintain user interest and engagement effectively. For instance, some physicians in this study identify issues that limit the effective management of diabetes and hypertension, such as variability in core disease measures obtained outside of

clinical settings [9]. They develop solutions, including wearables, as part of larger digital health programs to address these challenges [9]. By aligning the features and functionalities of wearables with distinct healthcare issues, developers can ensure that users find value and relevance in using these devices.

3.2 Empowerment through Diagnosis and Behavior Change

An interesting finding is that wearables that have the ability facilitating diagnosis, encouraging behavior change, and enabling self-monitoring tend to exhibit higher levels of user engagement [10]. Wearable devices provide a safe space for users to continue striving towards their goals, even if they experience setbacks, which helps to create an internal drive to remain dedicated and committed [5]. By offering features that empower users to take control of their health through real-time data insights and actionable feedback, wearables become indispensable tools that users actively engage with to improve their well-being.

3.3 Comfortable Design

As wearables have direct contact to human skin, "comfort-of-wear" is important for designing wearables that could encourage user's willingness to engage with wearables over time [12]. Factors such as adhesion, breathability, thermal management, soft power and signal processing [12] should be considered when designing a device. Moreover, factors such as colors and sizes of wearables have an influence on usage [11].

By focusing on these positive elements, developers and healthcare providers can optimize the retention of user interest in wearable technologies within healthcare settings, ultimately improving health outcomes and enhancing the overall user experience.

4 Challenges that hinder users from engaging with wearable technology

This section describes some of the barriers that users face when engaging with wearable technology. Through an examination of challenges such as psychological safety, environmental factors and affordability, one may gain more insights into the challenges that users encounter. These insights aid developers in producing more effective and user-centric devices,

thereby enhancing adoption and utility of wearables in healthcare and daily life.

4.1 Psychological safety

Trust emerges as a central theme, with participants emphasizing the importance of safety and confidence in wearable technologies [13]. Safety and trust were key considerations for both patients and health providers in relation to telemonitoring devices, in an experiment by Ferguson et al [13]. Participants highlighted the devices' impact on communication, health promotion, and disease detection [13]. Montgomery et al [14] found that wearables have the potential to address certain public health issues. However, they also highlight significant privacy concerns due to the possibility of combining the collected data with personal information from other sources, which could lead to the creation of discriminatory profiles, manipulative marketing tactics, and data breaches. Excessive self-monitoring may also have negative consequences, as reported in studies of type 2 diabetics where patients found the data provided by wearables uncomfortable, intrusive, and unpleasant [15].

4.2 Environmental factors

Interestingly, a study found that weather conditions such as temperature and sunshine, as well as certain weekdays have a notable influence on wearable usage patterns [16]. Participants in the study tend to use the wearable device less during days with extreme temperatures [16]. For example, physical activity reduces on days where the temperature is higher than 25 Celcius degrees [16]. Additionally, the study observes that weather conditions with a few hours of sunshine and moderate temperatures result in more wearable usage compared to days completely absent of sunshine [16]. The study also reveals a decrease in wearable device usage on weekends, particularly on Sundays and Saturdays, in contrast to weekdays [16]. This decline in usage behavior on weekends suggests that individuals are less inclined to wear the device during leisure days compared to regular workdays.

4.3 Affordability

Multiple studies highlight affordability as a significant barrier to the adoption of wearable technology. In a study conducted by the All of Us Re-

search Program, it was found that a considerable percentage of individuals expressed interest in fitness trackers but cited affordability as a reason for not owning one [17]. In another study about the adoption of wearables in the workplace, despite the willingness to invest in wearable technology from the respondents, concerns about cost are evident [18]. For example, some respondents express concerns about the initial investment required to purchase the devices, as well as ongoing costs related to maintenance, software updates, and potential replacements [18]. Additionally, in a study that aims at both users and non-users of wearable technology, both groups highlight concerns about the expense of wearables compared to the perceived benefits [19].

5 Discussion

This paper reviews the significance of addressing user engagement to sustain long-term adoption and utilization of wearable technology in health care context. By analyzing various factors that influence users both positively and negatively, this paper provides insights that could enhance user experience and promote continued use of wearable devices.

The findings of this paper align with the broader inquiry into the factors that contribute to sustained engagement with wearable technology. To keep a user engaged, some of the focus should be on having a clearly defined problem, enabling diagnosis and behavior change for the users, and comfortable design.

Moreover, this study resonates with existing research by emphasizing the importance of addressing barriers to engagement, such as users' trust, affordability of the devices and external environmental factors such as weather and sunshine. By acknowledging these challenges, developers can refine their designs and functionalities to better align with user preferences, thereby enhancing their satisfaction and adoption rates.

However, while this paper discusses several aspects of user engagement in wearable technology, it also leaves some important questions unanswered. For instance, further research is needed to explore the interplay between user engagement and health outcomes. Another intriguing question is about the long-term impact of wearable technology on behavior change and lifestyle modifications.

One demographic that could particularly benefit from the insights of this paper is the elderly population, which is not only growing in numbers

but also facing an increasing demand for wearables. These devices are particularly helpful for the elderly as they often experience a greater number of chronic health conditions [20]. Elderly individuals often require ongoing monitoring and management of conditions such as diabetes or cardiovascular disease. For instance, consider a wearable glucose monitoring device designed specifically for elderly individuals with diabetes. This device incorporates features enabling precise and real-time glucose monitoring, together with user-friendly interfaces, which provides actionable insights and supports positive behavior change. Such devices have the potential to empower elderly users in effectively managing their health conditions.

On the market, there are several wearables that have incorporated the findings from this paper to enhance their user experiences, one of the well-known wearable devices in Finland is the Oura ring. The Oura ring is designed to be sleek, lightweight, and unobtrusive, enabling users to wear it comfortably during the day and night without interference [21]. By prioritizing comfort of wear, the Oura ring encourages continuous usage, enabling users to seamlessly integrate it into their daily routines without disruption [21]. In terms of empowerment through diagnosis and behavior change, the Oura ring utilizes advanced sensors to track various physiological metrics, including heart rate variability, body temperature, and sleep patterns. By analyzing these data points, the Oura ring provides users with valuable insights into their health and well-being, empowering them to make informed decisions about their lifestyle and behavior [22].

6 Conclusion

This paper overviews the significance of user engagement in wearable technology in healthcare sector. It highlights the importance of positive elements such as clearly defined problems, empowering features for diagnosis as well as behavior change and comfortable design. In addition, the paper addresses barriers like trust, affordability, and environmental elements such as weather. By examining various influencing factors, this paper offers insights to improve user experience and extend user usage of wearable technology.

References

[1] Jayathilaka, W. A. D. M., Qi, K., Qin, Y., Chinnappan, A., Serrano-García, W., Baskar, C., . . . Ramakrishna, S. "Significance of Nanomaterials in Wearables: A Review on Wearable Actuators and Sensors," *Advanced Materials*, vol. 31, issue 7, article 1805921, 2019, doi: 10.1002/adma.201805921.

[2] Fernández-Carmés, T. M., Fraga-Lamas, P. "Towards the Internet of Smart Clothing: A Review on IoT Wearables and Garments for Creating Intelligent Connected E-Textiles," Electronics, vol. 7, issue 12, article 405, 2018, doi: 10.3390/electronics7120405.

[3] Lee, J., Kim, D., Ryoo, H.-Y., Shin, B.-S. "Sustainable Wearables: Wearable Technology for Enhancing the Quality of Human Life," *Sustainability*, vol. 8, article 466, 2016, doi: 10.3390/su8050466.

[4] A. Godfrey, V. Hetherington, H. Shum, P. Bonato, N.H. Lovell, S. Stuart, "From A to Z: Wearable technology explained," Maturitas, vol. 113, pages 40-47, 2018, ISSN 0378-5122, July 2018, doi: 10.1016/j.maturitas.2018.04.012.

[5] Wulfovich, S., Fiordelli, M., Rivas, H., Concepcion, W., Wac, K. "I must try harder: Design implications for mobile apps and wearables contributing to self-efficacy of patients with chronic conditions". Frontiers in Psychology, vol. 10, article 2388, 2019, doi: 10.3389/fpsyg.2019.02388

[6] M. Lalmas, H. O'Brien, and E. Yom-Tov, "Measuring User Engagement". Springer Nature, page 8, 2015, doi: 10.1007/978-3-031-02289-0.

[7] Y. W. Kim, S. H. Yoon, H. Hwangbo, and Y. G. Ji, "Development of a User Experience Evaluation Framework for Wearable Devices," in Human Aspects of IT for the Aged Population 2017. Lecture Notes in Computer Science, J. Zhou and G. Salvendy, Eds., vol. 10298, Cham: Springer, 2017, doi 10.1007/978-3-319-58536-9.

[8] S. Asimakopoulos, G. Asimakopoulos, and F. Spillers, "Motivation and User Engagement in Fitness Tracking: Heuristics for Mobile Healthcare Wearables," Informatics, vol. 4, no. 1, page 5, January 2017, doi: 10.3390/informatics4010005.

[9] Smuck M, Odonkor CA, Wilt JK, Schmidt N, Swiernik MA, "The emerging clinical role of wearables: factors for successful implementation in healthcare", NPJ Digital Medicine. vol. 4, issue 1, page 45, March 2021, doi: 10.1038/s41746-021-00418-3.

[10] Kang HS, Exworthy M. "Wearing the Future-Wearables to Empower Users to Take Greater Responsibility for Their Health and Care: Scoping Review.", JMIR Mhealth and Uhealth, vol. 10, issue 7, identifer e35684, July 2022, doi: 10.2196/35684.

[11] Auerswald, T., Meyer, J., von Holdt, K., Voelcker-Rehage, C. "Application of activity trackers among nursing home residents—a pilot and feasibility study on physical activity behavior, usage behavior, acceptance, usability and motivational impact", International Journal of Environmental Research and Public Health, vol. 17, issue 18, pages 1–21, 2020, doi: 10.3390/ijerph17186683.

[12] T. Shimura, S. Sato, P. Zalar, N. Matsuhisa, "Engineering the Comfort-of-Wear for Next Generation Wearables", Advanced Electronic Materials, vol 9, identifier 2200512, 2023, doi: 10.1002/aelm.202200512.

[13] Ferguson C, Hickman LD, Turkmani S, Breen P, Gargiulo G, Inglis SC. "Wearables only work on patients that wear them: Barriers and facilitators to the adoption of wearable cardiac monitoring technologies", Cardiovasc Digit Health Journal, vol. 2, issue 2, pages 137-147, Feb 2021, doi:10.1016/j.cvdhj.2021.02.001

[14] Montgomery, K., Chester, J., Kopp, K. "Health Wearables Ensuring Fairness, Preventing Discrimination, and Promoting Equity in an Emerging Internet-of-Things Environment," Journal of Information Policy, vol. 8 (Special Issue), pages 34–77, 2018, doi: 10.5325/jinfopoli.8.2018.0034.

[15] Piwek, Lukasz, Ellis, David, Andrews, Sally, Joinson, Adam. "The Rise of Consumer Health Wearables: Promises and Barriers," *PLOS Medicine*, vol. 13, article e1001953, 2016, doi: 10.1371/journal.pmed.1001953.

[16] Hendker, A., Jetzke, M., Eils, E., Voelcker-Rehage, C. "The Implication of Wearables and the Factors Affecting Their Usage among Recreationally Active People," International Journal of Environmental Research and Public Health, vol. 17, issue 22, article 8532, November 2020, doi: 10.3390/ijerph17228532.

[17] Holko, M., Litwin, T. R., Munoz, F., et al. "Wearable fitness tracker use in federally qualified health center patients: strategies to improve the health of all of us using digital health devices," npj Digital Medicine, vol. 5, article 53, 2022, doi: 10.1038/s41746-022-00593-x.

[18] Schall, M. C. Jr., Sesek, R. F., Cavuoto, L. A. "Barriers to the Adoption of Wearable Sensors in the Workplace: A Survey of Occupational Safety and Health Professionals," Human Factors, vol. 60, issue 3, pages 351-362, 2018, doi: 10.1177/0018720817753907.

[19] Burford, K., Golaszewski, N. M., Bartholomew, J. "“I shy away from them because they are very identifiable”: A qualitative study exploring user and non-user’s perceptions of wearable activity trackers," Digital Health, vol. 7, 2021, doi: 10.1177/20552076211054922.

[20] Nasir, S., Yurder, Y. "Consumers’ and Physicians’ Perceptions about High Tech Wearable Health Products," In Sener, S., Saridogan, E., Staub, S. (Eds.), World Conference on Technology, Innovation and Entrepreneurship, pages 1261–1267, Amsterdam, Netherlands: Elsevier Science Bv, 2015. doi: 10.1016/j.sbspro.2015.06.279.

[21] Oura, "Inside the Ring: Oura Horizon." Oura Blog, accessed on April 4th 2024, available at: https://ouraring.com/blog/inside-the-ring-oura-horizon/

[22] Oura. "The Oura Difference." Oura Blog, accessed on April 4th 2024, available at: https://ouraring.com/blog/the-oura-difference/

Perfecting the Orchestration of Workflows in the context of Microservices Architectures

Ionuț Groza
ionut.groza@aalto.fi

Tutor: Ylä-Jääski Antti

Abstract

This survey explores the central role of workflow orchestration in the context of microservices. It begins with an overview of the evolution from monolithic structures to the microservices paradigm, which brings several opportunities in terms of agile development but imposes the challenges of distributed computing environments. Focusing on the communication patterns between microservices components, the survey contrasts the orchestration and choreography methods, underlining their advantages and limitations. The paper analyses two orchestration engine typologies: standalone serverful orchestrators and provider-hosted orchestrators, along with their construction, advantages, and real-world usage scenarios. Furthermore, it discusses emerging trends such as application-level orchestrators and two novel technologies: Temporal and DF/Netherite, highlighting their impact on workflow management as state-of-the-art orchestration engines. By synthesizing insights from existing literature and recent advancements, this survey provides valuable perspectives for researchers and organizations and aims at optimizing workflow orchestration within microservices architectures.

KEYWORDS: Workflow, Orchestration, Microservices, Distributed Systems.

1 Introduction

In the ever-evolving society, people automate tasks to avoid engaging in trivial, side activities that make them lose focus of the intended outcome. Businesses leverage task automation through software [5] to produce revenue and advance society.

One approach to building task automation software is by adhering to a method known as microservices architecture. The microservices architecture [10] involves multiple smaller, specific, individual components that communicate with each other to complete a task. There are two representative communication strategies between the components of a microservices architecture: orchestration and choreography. Orchestration [17] relies on a central entity to distribute tasks to the participating components. Components emit events concerning their processing state. The orchestrating unit understands the events and understands how to coordinate the components toward the completion of a task. Choreography [19] assigns this responsibility to the components themselves, which are responsible for processing and delegating actions that are outside of their scope to more specialized components.

The microservice architecture has gained increasing interest because it enables rapid adoption of new business approaches and improved resource management, both human and computational. However, it implies increased complexity related to the communication between the containing components [13]. Addressing these communication challenges is important, as unstable communication may prevent, or interrupt the successful execution of a task, or produce unintended effects.

This paper analyzes orchestration typologies that ensure a set of tasks is executed once and to completion despite communication failures.

Section 2 is a history outline. It discusses the adoption of microservices and the communication patterns they involve. Section 3 presents two typologies of orchestration mechanisms, their adoption, architecture and a practical use-case scenario. Section 4 outlines the contributions of this paper and provides links to other solutions that showcase similar outcomes. Finally, Section 5 concludes the paper.

2 Background

Historically, software was designed around two large components describing the user interface and the system internals. This approach is known as monolithic architecture [11]. Businesses are constantly evolving to remain relevant in the market. In a monolithic architecture, reacting to changes is complex and lasting [18].

As a response, researchers created Service-Oriented Architecture (SOA) [12] to deliver software agility in terms of development. Instead of one or two large pieces of software, SOA consists of simpler services that can be developed independently by smaller teams and combined to deliver new functionalities [23]. The services communicate between themselves using a central messaging unit and a specialized messaging protocol [11]. As the application evolved, managing the communication component became too complex, making it unusable for large-scale, real-world applications. The issue became so serious that SOA started to be seen as a set of monolithic applications [9].

In this context, the microservices architecture arises from the successful implementations of SOA, as reported by the leading tech businesses [18].

2.1 Microservices

In 2006, Amazon, a high-performing retail business, described how it disengaged from a monolithic system and adopted a more distributed approach [9]. Similar to SOA, the method describes building around independent, smaller services that can be improved upon iteratively, facilitating the adoption of new business ideas. Other leading companies as well as prominent figures in the computer science field outlined and proved the validity of the microservice architecture, arguing for better cohesion with the underlying data storage, more clear boundary lines between teams, faster deployments, and release cycles [9].

A further evolution in the context of distributed systems is serverless computing, which offers a paradigm shift in how applications are developed, deployed, and managed. Similar to microservices, serverless computing relies on decomposing applications into smaller, independent units of functionality. However, unlike traditional microservices, serverless applications allow developers to focus on implementing the business logic, while cloud providers abstract away infrastructure management and dy-

namically allocate resources to match demand. This approach simplifies deployment and scaling and represents a lower cost proportional to resource usage [6].

Applications that adopt the microservices architecture inherit all the benefits of a distributed system. At the same time, they also employ the challenges related to such systems, including the complexity associated with distributed transactions, and the necessity of compensating for failures, which can lead to performance issues or degraded quality [23].

2.2 Microservices Composition

One of the most significant challenges in a microservices architecture is combining multiple components to accomplish a unified goal. The two proposed methods to overcome the challenge of composition are choreography and orchestration.

Choreography. In choreography, each service performs the business logic independently [1]. The services emit events to communicate with each other in a decoupled manner. Each service publishes events as they drive their computation. Similarly, each service listens to events emitted by other services and reacts accordingly.

Orchestration forces the components to communicate with a specialized entity, known as the orchestrator. The orchestrator communicates with all the other components and controls the flow of the business process.

In literature, choreography is almost always preferred over orchestration. In his book, Newman suggests that orchestrated components tend to become smart 'god' services that call on trivial CRUD (Create Read Update Delete) services. Following Newman's suggestion, such systems develop into a set of monoliths with the added cost of composition [18]. Others see choreographed systems as event-driven and reactive while orchestrated systems are synchronous and follow a request/response communication.

Recently, however, there has been an increased interest in orchestrated systems, especially in the context of reliable or long-running workflows. Such processes often employ a technique that compensates for failing actions, known as Saga [21]. Richardson advises the adoption of orchestration for complex sagas, stating that choreography can only be employed in simple scenarios [20].

3 Workflow orchestrators

According to Souza et. al. [22], workflows are a group of tasks that abstract a business process. In modern businesses, workflows tend to be lasting or involve multiple services. These services that execute critical-for-businesses workflows are often unreliable and involve complex steps. As the workflows must be successful, the services must guard against unexpected failures, such as requests that take too long to respond, interrupted connections or network congestion. These types of issues are usually overcome by timeouts and retries, which add a increasing complexity to the workflow implementation and the infrastructure that executes them.

Workflow orchestrators allow developers to express workflows in terms of their business logic. They also simplify building and managing the infrastructure needed for detecting and recovering from failures. This section presents two state-of-the-art orchestrators in the context of workflow-based applications.

3.1 Standalone Serverful Orchestrator - Temporal

Usage

Temporal is employed in production by well-known companies including Netflix, Stripe, Snap Inc. and Datadog, for writing workflow-based applications at scale. One of the advantages of Temporal is that it allows writing workflows as code and it evolves around an active community forum, offering a better developer experience.

Architecture

The main components of a temporal system are workflows and activities, workers and task queues, and the central temporal service.

Workflows are the main components of a temporal system. They represent business logic that oftentimes involves dependencies on multiple systems that need to be handled reliably. A workflow is formed out of steps, where each step could invoke other workflows or activities. The workflows must be idempotent, as they may be rerun by the temporal service in case of failures.

Activities. Actions that do not respond in a deterministic approach, could be performed by activities. Common use cases for activities are accessing a database, calling an endpoint through the network or handling

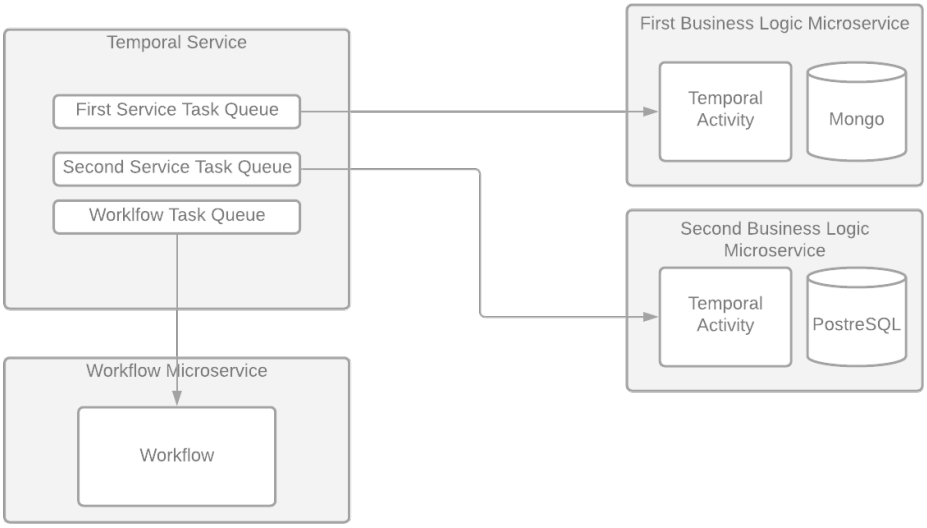


Figure 1. Microservice Architecture with Temporal [https://community.temporal.io/t/springboot-microservices-managed-by-temporal-io-rabbitmq/1489/6]

randomness. These actions are not deterministic as, if executed twice, they may not yield the same result. Determinism is important for reliable systems where certain steps can be executed multiple times because it guarantees each step will be executed at least once.

Workers and Task Queues. Workers are the entities that run the workflows and activities. Each worker listens to one or more task queues. Task queues contain events that instruct the worker what step they need to execute at a certain time.

Temporal Service is the component that pushes events about the state of the workflow execution to the task queues. For security reasons and for maintaining a loose coupling between logic and infrastructure, the temporal service does not execute any application logic but rather schedules the steps that need to be executed by the workers so that a workflow completes reliably. The orchestration takes place at the temporal service level. At any point in time, the service is aware of the current state of the application including the steps that need to be executed, the history of the events and the map of relations between workers and task queues.

Use-case

Figure 1 depicts how Temporal can be adopted to manage microservices composition.

The picture depicts two services that are responsible for implementing the business logic. The temporal service maintains a pair of a task queue

and a worker for each business component. Another pair is deployed for running a workflow that involves both business components. Mirroring the online shop example, the workflow is initiated when the user places an order on the website. The temporal service receives the event and starts scheduling the tasks in the order they need to be executed by the workers. At a closer look, this can include scheduling an activity to the storage service to reserve the items and another one to the payments service to charge the customer.

The temporal service performs the workflow execution using events that are persisted in storage. This allows the temporal service to supervise the workflow execution and retry the same steps dictated by the history of events in case of unexpected failures. Other actions can be taken in case of failure. These include timeouts, rate limiting, workflow cancellation or marking an execution as idle in case of long-running workflows.

3.2 Provider-Hosted Orchestrator - DF/Netherite

Usage

Although not there yet, Temporal offers software as a service (SaaS) services and it is in an ongoing process to support workflows running as functions in cloud [4]. However, Researchers at Microsoft have already seen the demand for running workflows serverless and have produced a solution extending the durable function (DF) service. The solution is deployed in production in nearly all applications built with DF, which count for about 6% [7] of all serverless function apps deployed on Azure and it is constantly improved upon, with specialized backend engines such as Netherite [3].

Architecture

DF abstracts business functionality in work items. These can be simple functions, usually stateless, or stateful functions, which require some sort of storage to maintain their state. There are several approaches to implementing serverful functions in the cloud. One example is Azure's Durable functions (DF) service, which enables stateful serverless functions, usually hiding event queues that keep the history of execution so it can be reproduced in case of failures and the cloud storage where these persist.

By itself, DF can be seen as an orchestrator. It allows writing business logic in terms of orchestrations, which are stateful functions. It allows the

adoption of an object-oriented approach through its entity framework. Orchestrations could be adopted for writing parallel tasks while entities create a medium where operations that modify state are processed as serial tasks. The orchestration is participates with the execution of workflows, known as orchestrations in the DF, but similar to Temporal, the orchestrator does not execute the tasks himself but rather schedules the tasks that need to be executed.

In temporal, the workers are responsible for task execution. In DF, tasks are executed as cloud functions, with the benefit of scaling high and scaling to zero. Another novelty in DF is the extensibility of backend engines. Netherite, for example, is a backend engine that improves latency, throughput and cost, by using partitions, which abstract a set of stateful instances [8, 7].

Use-case

The same e-commerce workflow can be implemented in DF. When the customer checks out an order, DF starts a check-out orchestration, using Azure functions, the Function as a Service (FaaS) implementation from Microsoft. The check-out could invoke the price calculation orchestration, the payment and the shipping orchestrations sequentially. The price calculation orchestration could start three tasks in parallel for calculating discounts, shipping prices and taxes. The event history is persistent at every step, such that, if any of the invoked functions fails, the engine can retry, or halt the workflow waiting for human interaction.

DF provides a many useful patterns, including function-chaining, branching, fan-in/fan-out, human interaction and timed invocations [2]. Moreover, as the workflow executes entirely on the cloud, it provides the benefits of high scalability and availability and reduced cost, proportional to the scalability rate.

4 Contribution and Further Reading

The main contribution of this paper is identifying the orchestrator typologies and providing contrasting information about the advantages, construction and usage of each type. These typologies are brought into discussion after outlining the history of developing workflow-based applications, from monoliths to microservices and functions, where the latter two demand a specific composition pattern, namely orchestration.

The two main typologies discussed are standalone serverful orchestrators and provider-hosted orchestrators. Recently, researchers have investigated another typology, application-level orchestrator [16], which promises the same correctness of execution as the existing engines, without an extra service that needs to be maintained and possibly paid for, and avoids vendor lock-in as it is packaged as a runtime library.

All presented typologies are being continually improved upon. Moreover, In 2023, [15] introduced Composable Resilient Steps (CReSt) and Deduplicated Asynchronously Recoverable Queues (DARQ) as a CReSt implementation for building cloud-native applications. The authors advertise DARQ as being a foundational block for creating orchestrators such as Netherite and Temporal.

Although they present benefits such as fault-tolerance, scalability, availability, ease of development and reduced cost, orchestration engines are not built for systems that require low latency. Researchers continue to investigate how orchestration engines can be optimized at each level to correspond to the needs of low-latency systems with Nightcore [14] being one such example.

5 Conclusion

In conclusion, workflow orchestration is a building block to implementing efficient, scalable and reliable microservices solutions in modern software development. This article has outlined the evolution of architectural approaches, from monolithic structures to agile microservices, highlighting the role of orchestration in designing seamless coordination and communication between distributed components. Through the exploration of various orchestration mechanisms, such as standalone serverful orchestrators and provider-hosted orchestrators, the paper emphasizes the need for robust solutions capable of ensuring the successful execution of complex business processes.

Recent advancements in workflow orchestration technologies suggest a promising trend for future innovation. This positions the field of workflow orchestration in the context of microservices architectures as a fertile ground for further exploration within the science of software engineering.

References

[1] Choreography vs orchestration. https://theburningmonk.com/2020/08/choreography-vs-orchestration-in-the-land-of-serverless/. Accessed: 2024-03-04.

[2] Durable functions. https://learn.microsoft.com/en-us/azure/azure-functions/durable/. Accessed: 2024-04-04.

[3] Netherite. https://microsoft.github.io/durabletask-netherite//. Accessed: 2024-04-04.

[4] Temporal - support for serverless workflows. https://community.temporal.io/t/support-for-serverless-workflow/8968. Accessed: 2024-04-04.

[5] Rainer Alt, Jan Marco Leimeister, Thomas Priemuth, Stephan Sachse, Nils Urbach, and Nico Wunderlich. Software-Defined business. *Business & Information Systems Engineering*, 62(6):609–621, December 2020. doi: 10.1007/s12599-020-00669-6.

[6] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. Serverless computing: Current trends and open problems. *Research advances in cloud computing*, pages 1–20, 2017.

[7] Sebastian Burckhardt, Badrish Chandramouli, Chris Gillum, David Justo, Konstantinos Kallas, Connor McMahon, Christopher S Meiklejohn, and Xiangfeng Zhu. Netherite: Efficient execution of serverless workflows. *Proceedings of the VLDB Endowment*, 15(8):1591–1604, 2022.

[8] Sebastian Burckhardt, Chris Gillum, David Justo, Konstantinos Kallas, Connor McMahon, and Christopher S Meiklejohn. Serverless workflows with durable functions and netherite. *arXiv preprint arXiv:2103.00033*, 2021.

[9] C. Carneiro and T. Schmelmer. *Microservices From Day One: Build robust and scalable software from the start*. Apress, 2016.

[10] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: Yesterday, today, and tomorrow. In Manuel Mazzara and Bertrand Meyer, editors, *Present and Ulterior Software Engineering*, pages 195–216. Springer International Publishing, Cham, 2017. doi: 10.1007/978-3-319-67425-4_12.

[11] D. Gonzalez. *Developing Microservices with Node.js*. Community experience distilled. Packt Publishing, 2016.

[12] T.O. Group. *SOA Source Book*. TOGAF Series. van Haren Publishing, 2009.

[13] 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. doi: 10.1109/MS.2018.2141039.

[14] Zhipeng Jia and Emmett Witchel. Nightcore: efficient and scalable serverless computing for latency-sensitive, interactive microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 152–166, 2021.

[15] Tianyu Li, Badrish Chandramouli, Sebastian Burckhardt, and Samuel Madden. Darq matter binds everything: Performant and composable cloud programming via resilient steps. *Proceedings of the ACM on Management of Data*, 1(2):1–27, 2023.

[16] David H Liu, Amit Levy, Shadi Noghabi, and Sebastian Burckhardt. Doing more with less: orchestrating serverless applications without an orchestrator. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1505–1519, 2023.

[17] Manuel Mazzara and Sergio Govoni. A case study of web services orchestration. In *Coordination Models and Languages*, pages 1–16. Springer Berlin Heidelberg, 2005. doi: 10.1007/11417019_1.

[18] S. Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015.

[19] C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003. doi: 10.1109/MC.2003.1236471.

[20] C. Richardson. *Microservices Patterns: With examples in Java*. Manning, 2018.

[21] A. Rotem-Gal-Oz. *SOA Patterns*. Manning, 2012.

[22] Milton Secundino de Souza-Júnior, Nelson Souto Rosa, and Fernando Antônio Aires Lins. An execution environment as a service for adaptive long-running workflows, March 2021. doi: 10.1108/ijwis-12-2020-0077.

[23] E. Wolff. *Microservices: Flexible Software Architecture*. Pearson Education, 2016.

Comparative study of the modern cross-platform tools for mobile application development

Isroilkhon Salikhodjaev
isroilkhon.salikhodjaev@aalto.fi

Tutor: Simo Aaltonen

Abstract

This article provides an overview and comparative analysis of three modern cross-platform mobile development technologies: React Native, Flutter, and Kotlin Multiplatform. Each technology is examined in terms of its architecture, advantages, drawbacks, and recommended use cases. React Native, relying on JavaScript ecosystem, offers rapid development and code reusability. Flutter, built on Dart programming language, excels in performance and user interface consistency. Kotlin Multiplatform, utilizing Kotlin, enables code sharing while maintaining a native user experience. Recommendations for technology selection are provided based on project requirements and development priorities. As mobile development evolves, informed decision-making and ongoing evaluation of these technologies are essential for successful cross-platform application development.

KEYWORDS: Cross-platform, Mobile Applications, React Native, Flutter, Kotlin Multiplatform

1 Introduction

In the last few years, smartphones have seen a rapid increase in usage as a result of the technological advancements that make the devices ex-

tremely convenient to use. The demand for high-quality mobile applications has been steadily increasing ever since, prompting an increase in application development tools. To simplify the development experience, modern cross-platform frameworks have emerged, allowing developers to utilise a unified code base for their applications. The most popular among them are React Native, Flutter, and the newly released Kotlin Multiplatform. These frameworks mark a departure from the traditional practice of having separate native code bases for iOS and Android applications.

The frameworks provide a better development and maintenance experience, but it often comes at a cost of performance efficiency. As a result, developers are commonly faced with a challenge to decide whether the ease of development outweighs the small decrease in application performance. With an increase in popularity of the cross-platform solutions, it becomes increasingly more important to carefully examine the strengths and limitations of these frameworks to understand their intended use-cases.

This paper provides a review of recent developments in the aforementioned cross-platform solutions for mobile application development. Specifically, the paper aims to provide a comparison between the capabilities of each framework and to suggest use-case scenarios for each one.

This paper is organized as follows. Section 2 provides a general overview and background of the leading cross-platform frameworks technologies. Section 3 examines the architecture of each tool in more detail. Section 4 presents a discussion on particular strengths and shortcomings of each framework based on the available literature to find a suitable use case for each one. Finally, Section 5 provides conclusions based on the conducted review.

2 Overview of cross-platform solutions

2.1 React Native

Released back in 2015, React Native is a cross-platform mobile application development framework based on the existing React library for JavaScript (JS)[3]. The framework allows developers to write a shared JavaScript code both for Android and iOS platforms while keeping the code structure akin to a web application. User interface (UI) elements in

React Native are created almost the same way as HTML components in web applications, but the components are translated to native elements during compilation. Cross-platform applications built with React Native rely on an interpreter that executes the source code at runtime for the target host platform.

Within the React Native ecosystem, developers can choose to utilise the Expo development platform, which provides a set of tools built on top of React Native to simplify building and distributing mobile applications [6]. Expo abstracts away platform-specific Application Programming Interfaces (APIs) and instead provides a list of platform-agnostic APIs that simplify the development process even further. However, such simplification often comes at the cost of flexibility, as Expo offers only limited support for platform native libraries.

2.2 Flutter

Flutter is a cross-platform software development kit developed by Google and released in 2017 [14]. Unlike React Native, Flutter uses Dart programming language for creating and managing application UI. Flutter uses its own set of customisable widgets for creating UI elements, so mobile applications written with Flutter do not rely on native components. As a result, applications tend to look and behave identical across the platforms. Flutter offers a straightforward way for developers to create responsive and customisable applications.

Flutter's distinct feature is its own rendering engine, which is used to create the UI components for the application [11]. Instead of using the platform's native components, Flutter's engine relies on special "Canvas" elements to precisely draw requested widgets. Thanks to this approach, cross-platform applications made with Flutter tend to look and behave identically across different platforms, as the engine is made to control every pixel on screen.

2.3 Kotlin Multiplatform

Kotlin Multiplatform (KMP) is a tool developed by JetBrains that allows developers to have a shared Kotlin module that is used both by Android and iOS applications [10]. According to JetBrains, KMP aims to reduce time and effort spent on maintaining two entirely separate projects while maintaining flexibility of native application development. This is typically

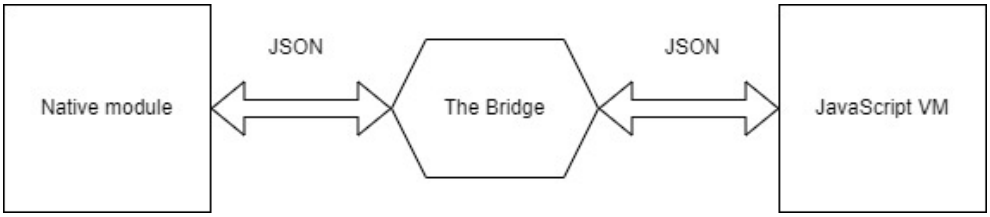


Figure 1. React Native Bridge communication

achieved by delegating the application's business logic to a shared Kotlin module while keeping the corresponding UI elements as native.

In addition to KMP, JetBrains also provides the Compose Multiplatform framework for creating a common UI for different platforms in a declarative manner [7]. This framework is based on the existing Jetpack Compose toolkit, which provides a modern way for managing UI components in native Android applications. At the time of writing this paper, Compose Multiplatform have not yet reached the production-ready status, so it is deemed to be out of scope for this work.

3 Evaluation

A good starting point for evaluating each technology's capabilities is to analyse their architecture. This paragraph aims to provide an outline of platform architecture and application structure that comes together with each of the discussed cross-platform tool.

3.1 React Native architecture

Traditionally, React Native applications consist of two main components: JS bundle and native module, each running on a separate thread. The JS bundle, also known as the JS virtual machine, is mainly responsible for processing the application's business logic written in JavaScript. The native module, on the other hand, is responsible for handling the incoming commands from the JS bundle thread to perform the native platform's operations. These two components are able to communicate with each other thanks to React Native's special Bridge component [4]. This communication model is illustrated in the Figure 1.

The Bridge component utilises asynchronous communication model for transferring messages between the JS and native modules. The messages themselves are encoded in a JSON format before being sent to the other side. The content of messages includes UI components that need to be

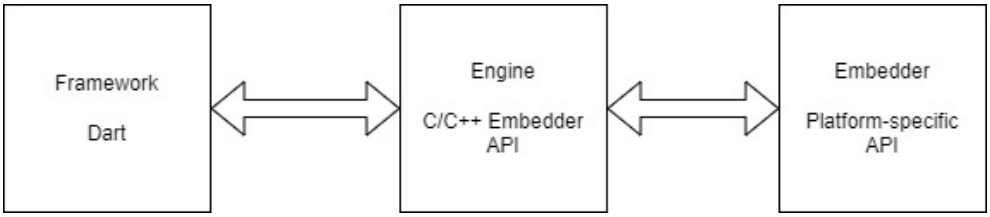


Figure 2. Flutter architecture

drawn, network requests that rely on platform's engine, and various touch events initiated by the application's user.

Starting from the version 0.68 released in 2022, React Native team introduced an alternative approach to the bridge architecture called the JavaScript Interface (JSI). With this new model, the JS thread no longer needs to rely on the bridge to transport json-encoded messaged to the native side. Instead, the JS thread now has access to the JSI layer, written typically in C++, through which the native functions can be called directly. This is achieved by having an intermediate object which holds a reference to the requested function. The JSI architecture allows the native and JS parts of the application to communicate synchronously without an additional overhead that is required for bridge communication. [5]

3.2 Flutter architecture

Flutter architecture consists of three main parts: embedder, engine, and framework. The embedder acts as an entry point for flutter applications and is responsible for coordination with the underlying operating system of the device. The engine, written in C and C++, provides a set of low-level APIs that form a core of each Flutter application. Finally, the framework, implemented in Dart, includes the Flutter UI components alongside a rich set of libraries and packages. Flutter developers primarily interact with the framework components, though it is possible to access the engine APIs through a package that wraps C++ code into Dart classes [9]. A high-level overview of Flutter's architecture is shown in the Figure 2.

Unlike React Native, which utilises interpreted approach, Flutter relies on cross-compilation for executing applications in the target platform. With this approach, a special cross-compiler is used for compiling application's source code into executable binaries for desired operating systems. Flutter achieves this by using Android's Native Development Kit (NDK) and iOS's Low-Level Virtual Machine (LLVM) to compile the C/C++ code generated by the Flutter engine. [15]

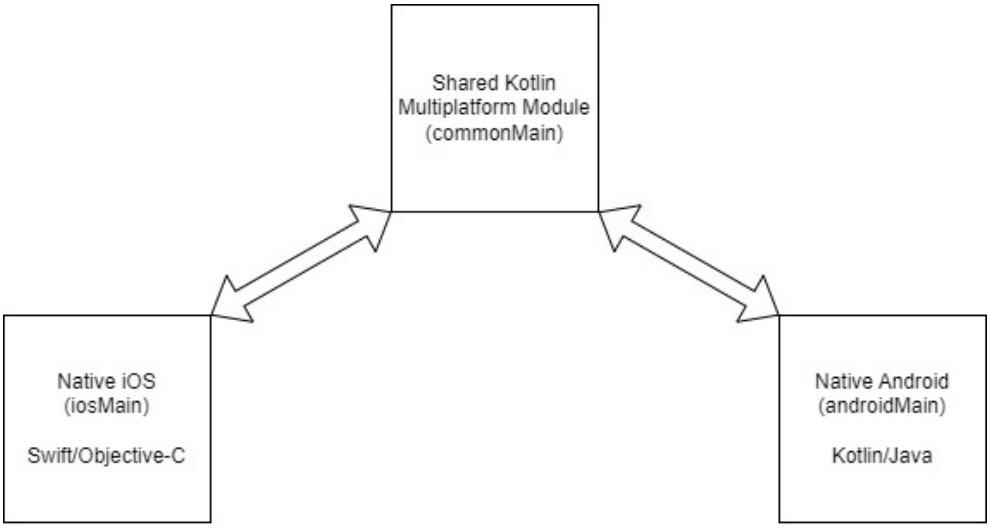


Figure 3. Kotlin Multiplatform project architecture

3.3 Kotlin Multiplatform architecture

Kotlin Multiplatform projects usually consist of three separate modules: the Android native part (androidMain), the iOS native part (iosMain), and the shared Multiplatform part (commonMain). The Android and iOS parts contain the corresponding native code that developers decide to keep separate. Such code typically is typically limited to platform-specific APIs and UI elements, but developers are free to keep other parts of the application separate as well. The Multiplatform part is intended to include parts of code which is not inherent to the specific platform's implementation. This includes but not limited to network and database operations along with class and interface logic. The shared module itself is divided into common Android, common iOS, and overall common parts for cases when platform-specific API calls are needed from within the Multiplatform code [12]. This architecture is presented in the Figure 3.

Similar to Flutter, Kotlin Multiplatform makes use of the cross-compilation approach for running the applications. During the build process, applications targeting Android are compiled using the Kotlin/JVM platform, while applications targeting iOS are built with Kotlin/Native technology. Using Kotlin/JVM, Multiplatform applications are built with the same technology as normal native Android applications due to Kotlin's complete interoperability with Java. Kotlin/Native, on the other hand, is based on the LLVM technology, which allows developers to build native binaries that can be executed without a separate VM. [10]

4 Discussion

Each cross-platform development solution has its own set of advantages and disadvantages, so it is crucial for mobile application developers to make informed decisions when choosing the development technology. This sections aims to present an overview for a selection of qualities inherent to each of the discussed technology.

4.1 React Native

React Native's great advantage compared to other technologies is its JavaScript runtime. Because of this, the technology was exposed to an ever-increasing JavaScript ecosystem from its start. Moreover, React Native's similarity to the React library on the web makes it a perfect starting point for transforming an existing web application into a mobile native one. If the target application does not require complex platform-specific operations, the development process can be expedited even further by utilising the Expo platform [2].

The development speed offered by React Native might be enough of a factor to co, but it may also lead to some unwanted consequences. Firstly, the application performance might be slowed down due to the additional overhead from the React Native architecture. While this can be mitigated by making use of the JavaScript Interface, a portion of applications and third-party packages in React Native ecosystem are still reliant on the older bridge component. This performance degradation can be particularly noticeable on devices with older hardware, as reported in [4].

Additionally, maintaining existing React Native applications may pose certain challenges for developers, especially when it comes to package version updates. The problem is often caused by certain third-party packages that introduce breaking changes to their code, which makes them incompatible with other libraries used in the application [13]. Because of this, React Native developers often need to maintain a careful versioning system in order to lessen the impact of such updates, which may lead to some decrease in the development speed.

4.2 Flutter

Flutter offers several great advantages compared to other similar technologies in cross-platform development field. Applications built with Flut-

ter often display increased consistency and reliability in their behavior. This quality is achieved thanks to Flutter's engine rendering own components without relying on native components. Additionally, UI components made with Flutter can be customised to great extent, which is often seen as a desirable feature in cross-platform frameworks.

Another notable feature of Flutter applications is the performance speed. Flutter's reliance on the cross-compilation approach for building mobile applications provides an improved performance, which at times can be comparatively close to native application performance. [8]

As with every other technology, there are several caveats that need to be considered when discussing Flutter's characteristics. For instance, the Dart programming language has little usage outside of the Flutter framework, so using Flutter necessitates a certain learning curve for developers who are not already familiar with Dart. Furthermore, Flutter applications tend to have a larger final build size and increased random-access memory consumption, as reported in [1]. Such increased resource consumption may not be desirable for applications that require performance-heavy operations.

4.3 Kotlin Multiplatform

KMP's primary appeal lies in the fact that a large portion of modern native Android applications are already written in Kotlin, so the shared module automatically offers a complete interoperability with the Android ecosystem. In practice, this means that Android applications that make use of the KMP modules have no additional overhead, as the shared code by default works the same way as native code. As analysed in [12], KMP code on iOS platform can perform on par or even faster than the native Swift code, though it is shown to consume higher amount of memory in the process.

Kotlin Multiplatform stands out among other cross-platform technologies in that it does not aim to reduce the entire application structure to a single codebase. Rather, it provides developers a flexibility to have a shared module only for desired parts of the application. As a result, developers can introduce the Multiplatform code to existing native applications incrementally, without having to rewrite everything from scratch. This also gives an option to keep the performance-critical parts of application in native code, thus avoiding additional overhead that comes with other cross-platform tools.

On the other hand, having separate Android and iOS codebases inevitably requires expertise in more than one programming language, which in case of KMP is Kotlin and Swift. Because of this, the learning curve for KMP might be the largest among cross-platform frameworks. Kotlin Multiplatform is also the newest among the discussed technologies, so the KMP ecosystem is not yet as rich as the ones in React Native or Flutter. The majority of Kotlin libraries for native Android predicate on the Kotlin/JVM APIs, which makes it incompatible with Kotlin/Native by default. Therefore, maintainers of open-source libraries need to put extra effort to ensure that their package is Multiplatform-compatible.

4.4 Recommended use cases

It is clear that all cross-platform solutions have their own unique characteristics, so there are distinct situations in which employing a particular technology is optimal. These situations depend on the nature of the target application, the skill set of developers working on the project, and the existing codebase of the project as a starting point. For example, if the desired mobile application needs to be created from the start, and the nature of the application is not remarkably complex, developers can choose the utilise React Native or Flutter frameworks. Between the two, React Native might be the better option if developers have expertise in web development technologies like React, and Flutter can be utilised for other cases. On the other hand, for projects with already existing native codebase, introducing KMP can result in reduced work efforts for the application, as KMP allows to reuse shared Kotlin code across the platforms.

5 Conclusion

In conclusion, this paper has explored and compared three prominent modern cross-platform mobile development technologies: React Native, Flutter, and Kotlin Multiplatform. Each technology offers unique features and capabilities, catering to different development scenarios and project requirements. React Native provides a great increase in development speed for application thanks to its reliance on JavaScript ecosystem. Flutter, on the other hand, excels in providing highly customisable UI experience without a big compromise in application performance, while still maintaining a single shared codebase both for Android and iOS platforms.

Finally, Kotlin Multiplatform, leveraging Kotlin as its primary language, shines in enabling code sharing between platforms while also maintaining a native user experience when necessary, thus providing a balanced approach to cross-platform development.

The selection of the technology for mobile application development primarily depends on the specific needs and goals of the project. More importantly, the choice of technology should align with the development team's knowledge and experience in order to ensure a better development and maintenance experience. It is also important to note that the mobile development landscape continues to evolve rapidly, so each technology has the potential to improve upon its existing drawbacks. Therefore, ongoing evaluation and analysis of these technologies is crucial for facilitating successful delivery of cross-platform mobile applications.

References

[1] Hugo Allain et al. Improving productivity and reducing costs of mobile app development with flutter and backend-as-a-service. Master's thesis, 2020.

[2] Andreas Biørn-Hansen, Christoph Rieger, Tor-Morten Grønli, Tim A Majchrzak, and Gheorghita Ghinea. An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, 25:2997–3040, 2020.

[3] Bonnie Eisenman. *Learning React Native: Building native mobile apps with JavaScript*. " O'Reilly Media, Inc.", 2 edition, 2017.

[4] Rasmus Eskola. React native performance evaluation. July 2018.

[5] Timur Fatkhulin, Rafif Alshawi, Alena Kulikova, Alexander Mokin, and Anna Timofeyeva. Analysis of software tools allowing the development of cross-platform applications for mobile devices. In *2023 Systems of Signals Generating and Processing in the Field of on Board Communications*, pages 1–5, 2023.

[6] Hugo Hutri. Comparison of react native and expo. 2023.

[7] JetBrains. Compose multiplatform. JetBrains: Developer Tools for Professionals and Teams.

[8] Wellington Oliveira, Bernardo Moraes, Fernando Castor, and João Paulo Fernandes. Analyzing the resource usage overhead of mobile app development frameworks. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, EASE '23, page 152–161, New York, NY, USA, 2023. Association for Computing Machinery.

[9] Matilda Olsson. A comparison of performance and looks between flutter and native applications: When to prefer flutter over native in mobile application development. 2020.

[10] Nagy Robert. *Simplifying Application Development with Kotlin Multiplatform Mobile: Write Robust Native Applications for IOS and Android Efficiently / Robert Nagy.* Birmingham Packt Publishing, Limited, 2022.

[11] Kewal Shah, Harsh Sinha, and Payal Mishra. Analysis of cross-platform mobile app development tools. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–7, 2019.

[12] Anna Skantz. Performance evaluation of kotlin multiplatform mobile and native ios development in swift. July 2023.

[13] Farzaneh Tajik and Josefin Lindström. Swift vs react native: A performance comparison for automatization of gamification using qr-codes. October 2023.

[14] Eric Windmill. *Flutter in action*. Simon and Schuster, 2020.

[15] Wenhao Wu. React native vs flutter, cross-platforms mobile application frameworks. March 2018.

A Survey of Clustering: Algorithms and Optimization

Jiaqi Chen
jiaqi.chen@aalto.fi

Tutor: Parinya Chalermsook

Abstract

Clustering is the process of grouping similar objects based on inherent patterns or similarities. It plays an important role in data mining research and application, helping with understanding various phenomena. This paper presents an overview of typical clustering algorithms and their strengths and weaknesses, summarizes common measurement metrics for clustering problems and discusses the challenges of clustering algorithm design.

KEYWORDS: Clustering, Clustering algorithm, Clustering measurement

1 Introduction

Clustering is a widely used method in various areas, including machine learning, data science and statistics. This method refers to dividing objects into subsets based on their inherent similarities. According to different data types and application scenarios, researchers can use different clustering algorithms to discover the patterns within the data, thus obtaining a better understanding of their research objects.

No clustering algorithm can be generally used on all different types of data or in all scenarios. When applying clustering algorithms to a specific

problem, the performance relies on many factors, such as data presentations, measurements, and the initial cluster state. With the rise of high-dimensional and big data, effective clustering algorithms are needed to handle more complex problems.

In order to provide references for employing, designing and improving clustering algorithms, this paper surveys theoretical developments of clustering, with a focus on algorithmic and optimization challenges of clustering problems.

The paper is structured as follows. Section 2 introduces some common clustering algorithms and typical improvements on the algorithms. Section 3 describes metrics for algorithm performance measurement. Section 4 analyses challenges in current clustering methods and discusses future research trends in this field. Finally, Section 5 presents the conclusion.

2 Clustering algorithms

2.1 Center-based clustering

Center-based algorithms are designed based on the assumption that the center of a cluster is the center of data points that belong to the same group. In these algorithms, each data point is assigned to its nearest center, and an objective function is used to evaluate the quality of the partition. Starting with an initial partition, the clustering is optimized by iteration until it reaches an optimal result for its objective function.

The objective function can be defined in different ways. Common optimization criterion often used are k-center, k-median and k-means, in which the goal is defined as minimizing the maximum distance, the sum of distances, and the sum of squares of distances between data points and their corresponding cluster center, respectively.

2.2 Density-based clustering

Different from center-based clustering, density-based clustering algorithms assume that data points in the same cluster are located closely to each other, composing a contiguous region of high density, with sparse or empty areas around it to separate one cluster from others. These algorithms are good at identifying clusters of arbitrary shapes, are resistant to noise and outliers, and do not need to pre-define the cluster number[15]. However,

with high efficiency, density-based algorithms are usually computing resources consuming, which will become a problem when dealing with large data sets.

Density-based spatial clustering of applications with noise (DBSCAN)[5] is the most well-known density-based clustering algorithm. It follows the basic idea of density-based clustering, and is sensitive to its two parameters, the radius of the neighborhood and the point number threshold in a neighborhood. To solve this problem, ordering points to identify the clustering structure (OPTICS)[2] was proposed to produce a hierarchical output by creating an augmented ordering of data points based on their reachability distance.

2.3 Graph Theory-based clustering

Graph is a commonly used data structure in data analysis field. It consists of nodes and edges that connect the nodes, modeling the relationships between objects and their features. In clustering based on graph theory, dataset is converted to a weighted, undirected graph, where objects, i.e., data points, are represented as nodes in the graph, and the relationship among data points are represented as edges, with the weight on each edge reflecting the proximity between data points.

2.4 Model-based clustering

Model-based clustering assumes that data are generated from a specific statistical model, usually a combination of a basic probability model[3]. This clustering technique aims to find an appropriate model and optimize it to better fit the observed data distribution. With the obtained mixture model, data is represented by the model, and the components of the model represent the clusters.

Model parameters can be estimated by Maximum Likelihood Estimation (MLE) [12] and the Bayesian information criterion (BIC) [8]. A commonly used model-based clustering algorithm is expectation-maximization (EM) algorithm [10]. This algorithm iteratively maximizes the likelihood estimates of variables and can be applied to many different probabilistic models.

Compared to heuristic methods, model-based clustering is less computationally efficient but more robust[9]. Although its clustering result is dependent on model choices and parameter settings, it requires less sub-

jective decisions, such as dissimilarity measures and number of clusters.

2.5 Kernel-based clustering

Kernel-based clustering algorithms are developed based on Cover's theorem on the separability[4]. These algorithms use kernel method to transform the data into a high dimensional space by performing a nonlinear mapping, in which way some complex patterns that are not linearly separated can be linearly separated in higher dimensional space. Kernel method can be useful when using traditional center-based methods on data while the clusters seem not to be linearly separable.

Since the original data is mapped to a higher dimensional space, the curse of dimensionality appears. To solve the problem, the kernel trick[11] is applied to avoid explicitly defining the mapping process, using inner-product to reduce time consumption.

3 Performance measurements

3.1 Similarity

To decide the clustering, a standard is needed to measure the dissimilarity (distance) or similarity between objects, objects and clusters, and clusters[14]. All the clustering methods define their similarity measures to determine the closeness degree among data points[7]. The most common similarity measures are listed below.

Minkowski distance is a distance measure defined as

$$D(X_i, X_j) = \left(\sum_{k=1}^{m} (X_{ik} - X_{jk})^p \right)^{1/p}, \tag{1}$$

where X_i, X_j are two data points or objects, and m is the dimension of data. This distance is a generalized metric and has some special cases. When $p = 1$, it becomes Manhattan distance; when $p = 2$, it becomes Euclidean distance, which is the most well-known metric for clustering of numerical data; when $p = \infty$, it becomes Chebyshev distance, computed as Eq.(2)

$$D(X_i, X_j) = max_{1 \le k \le m} |X_{ik} - X_{jk}| \tag{2}$$

Cosine distance measures the similarity between two data points by compute the cosine of the angle between them. With data points represented as vectors, this metric measures the similarity as Eq.(3) and is

commonly used in information retrieval and text analysis.

$$D(X_i, X_j) = \frac{X_i \cdot X_j}{\|X_i\| \, \|X_j\|} \tag{3}$$

3.2 Validation Measures

Different clustering algorithms can generate different clusters based on the same data set. In addition, the parameter setting, initial state and data presentation used in the algorithm may also affect the clustering result. To evaluate the quality of the clustering, two main categories of metrics are used to validate the results of clustering algorithms, internal criteria and external criteria.

Internal criteria
Internal criteria do not depend on prior knowledge, but directly evaluate the clustering results based on the intrinsic characteristics of the original data. These metrics measure various aspects of clustering results, such as compactness inside the cluster and separation among clusters. Some common internal validation criteria are sihouette coefficient, Davies-Bouldin index and Dunn's index.

External criteria
External criteria require prior knowledge about data, using external information to validate clustering solutions. The underlying structure of data is usually unknown in real-world scenarios. Nevertheless, these metrics provide a valid standard, which can be used to assess the matching between the clustering result and the pre-defined structure of the data set. Some common internal validation criteria are Rand Index (RI), Normalized Mutual Information (NMI), Fowlkes-Mallows Index (FMI) and F1 score.

4 Challenges with clustering

4.1 Determining the number of clusters

One of the major challenges of clustering is to determine the number of clusters. In many application areas, the shape, size and density of clusters are arbitrary, and the cluster number is unknown. Due to the lack of prior knowledge, it is difficult to identify the number of clusters embedded in

the data. Many existing algorithms require users to specify the number of clusters as an input parameter. New approaches are needed to utilize the data properties and discover the number of clusters automatically.

4.2 High-dimensional data

When clustering high-dimensional data, algorithms such as K-means and Gaussian mixture model-based clustering are not very effective[6]. Traditional distance or similarity metric become less meaningful and cannot accurately capture the similarity information among data points. Sparsity issues may occur due to high dimensionality. In addition, high dimensionality of data can also increase the computational complexity of algorithms. To solve this problem, high-dimensional data can be projected to a lower-dimension subspace by dimensionality reduction techniques or feature selection methods.

4.3 Scalability

As the size of data set increases, some clustering algorithms have difficulty scaling to data sets with large-size instances, since the process may become time-consuming or even infeasible. One direct way to solve the problem is using high-capacity GPU to acquire more computational resources[13]. Moreover, some techniques can help with processing large-scale data, such as mini-batch clustering and distributed computing[16].

4.4 Other challenges

In real world, noise and outliers are inevitable, thus the robustness against noise and outliers is important for a clustering algorithm. A robust clustering algorithm should be insensitive to the order of input, resistant to noise and able to identify outliers.

Although many algorithms are designed for numerical data, applications may require clustering data with different types of attributes, such as categorical, ordinal, binary and mixed data types[1]. Therefore, clustering algorithms are required to be flexible in processing different data types.

When using the most commonly used distance measures, Euclidean distance, clustering algorithms tend to discover spherical clusters. However, the shape of the cluster could be irregular. Advanced algorithms such as density-based and spectral algorithms show a better performance

in detecting clusters with arbitrary shapes.

5 Conclusion

Clustering algorithms are widely used in both research and practical applications of data mining. They can reveal hidden relationships among different data points, extracting meaningful insight. With a wide range of clustering algorithms and optimization techniques, the strengths and weaknesses of different algorithms need thorough analysis, and the challenges of clustering problems require careful consideration. Further research can focus on enhancing the robustness, effectiveness and scalability of clustering methods, and utilize the characteristics of data, thus ensuring the quality and reliability of clustering results in various complicated domains.

References

[1] Parul Agarwal, M Afshar Alam, and Ranjit Biswas. Issues, challenges and tools of clustering algorithms. *arXiv preprint arXiv:1110.2610*, 2011.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, page 49–60, New York, NY, USA, 1999. Association for Computing Machinery. https://doi.org/10.1145/304182.304187.

[3] Charles Bouveyron and Camille Brunet-Saumard. Model-based clustering of high-dimensional data: A review. *Computational Statistics Data Analysis*, 71:52–78, 2014. https://doi.org/10.1016/j.csda.2012.12.008.

[4] Thomas M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, 1965. doi: 10.1109/PGEC.1965.264137.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[6] Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022. https://doi.org/10.1016/j.engappai.2022.104743.

[7] Absalom E. Ezugwu, Amit K. Shukla, Moyinoluwa B. Agbaje, Olaide N. Oyelade, Adán José-García, and Jeffery O. Agushaka. Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature. *Neural Computing and Applications*, 33(11):6247–6306, jun 2021.

[8] Chris Fraley and Adrian E Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

[9] Isobel Claire Gormley, Thomas Brendan Murphy, and Adrian E. Raftery. Model-based clustering. *Annual Review of Statistics and Its Application*, 10(Volume 10, 2023):573–595, 2023. https://doi.org/10.1146/annurev-statistics-033121-115326.

[10] Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[11] Saburou Saitoh. Theory of reproducing kernels and its applications. *Longman Scientific & Technical*, 1988.

[12] Keshav Sanse and Meena Sharma. Clustering methods for big data analysis. *International Journal of Advanced Research in Computer Engineering & Technology*, 4(3):642–648, 2015.

[13] Ali Seyed Shirkhorshidi, Sr Aghabozorgi, Teh Wah, and Tutut Herawan. Big data clustering: A review. In *Computational Science and Its Applications – ICCSA 2014*, pages 707–720. Springer International Publishing, 2014. https://doi.org/10.1007/978-3-319-09156-3_49.

[14] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005. doi: 10.1109/TNN.2005.845141.

[15] Hui Yin, Amir Aryani, Stephen Petrie, Aishwarya Nambissan, Aland Astudillo, and Shengyuan Cao. A rapid review of clustering algorithms. *arXiv preprint arXiv: 2401.07389*, 2024.

[16] Btissam Zerhari, Ayoub Ait Lahcen, and Salma Mouline. Big data clustering: Algorithms and challenges. In *BDCA'15: Proceedings of the international Conference on Big Data, Cloud and Applications*, 2015.

Energy Prediction in Cloud Computing: Contrasting Approaches in Virtual machines and Containers

Jimena Bermudez Bautista
jimena.bermudezbautista@aalto.fi

Tutor: Jaakko Harjuhahto

Abstract

In cloud computing, energy efficiency has quickly become an important issue, as it is transforming IT resources by providing scalable solutions, raising concerns about its environmental effect, mainly on the energy used in data centers. This paper compares energy prediction and calculation methodologies between two leading used technologies: Virtual machines (VMs) and containers. This research examines the energy consumption condition in the cloud computing sector to unveil ways to increase resource utilization. The research explores the role of VMs and containerizing technologies in energy efficiency. It reviews existing procedures for estimating energy consumption and their application to determine the optimal strategies to decrease energy consumption. The comparative analysis helped find the current state of the art of energy calculation and prediction and a new architecture proposal for sustainable cloud infrastructures. This new proposed architecture utilizes an energy optimization strategy to implement VM and container advantages, opening a new frontier of efficiency in cloud computing.

KEYWORDS: cloud computing, energy consumption, virtual machines, containerization, energy efficiency, machine learning, energy prediction

1 Introduction

Cloud computing has revolutionized data and applications management, access, and storage by remotely offering scalable and efficiently measurable IT resources. This technology facilitates remote access to decentralized IT infrastructure, including networks, servers, storage, applications, and services designed to provide scalable and measurable IT resources [1]. Cloud computing has opened up a new horizon for software development and emerging technologies, increasing the need for these services. On the other hand, the environmental impact of cloud computing has become noticeable, with data centers around the world consuming about terawatt-hours (TWh) of electricity by 2022, accounting for about 22% of global electricity energy consumption [3]. This usage is driven by energy demands in computing power and cooling systems at these facilities, where artificial intelligence (AI) workloads and cryptocurrency mining play a significant role. By 2026, data center capacity consumption is predicted to double, reaching between 650TWh and 1,050TWh, based on factors such as business development, AI, and cryptocurrency development [3][14]. With these concerning predictions, this paper aims to review the current state of the art of energy consumption of cloud computing to find the strategies for resource efficiency and investigate the impact of virtual machines and containerization to deepen the research of the strategies used and their energy implications. By examining the methodologies for energy consumption estimation or calculation and the effectiveness of existing optimization strategies, this research seeks to contribute to the sustainable advancement of cloud computing practices. This paper is organized as follows. Section 2 lays the foundational groundwork by discussing the core principles of cloud computing and the critical role of virtualization and containerization technologies in this context. Section 3 discusses energy prediction and the methodologies for virtual machines and containers. Section 4 summarizes the core papers that calculate or predict energy consumption on VMs or containers. Section 5 compares the methods from the papers and analysis of the key findings. Finally, Section 6 presents concluding remarks and future work.

2 Cloud Computing

Cloud computing has changed how IT resources are used by providing scalable resources via the Internet. Historically, "cloud" served as a metaphor for the Internet—a vast, interconnected network enabling access to decentralized computing resources. The cloud symbol commonly illustrates this metaphorical representation in various technical documents and specifications related to web-based architecture. Within these boundaries, various cloud services like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Business Process as a Service (BPaaS) are provided to cater to business and individual needs. The rise of cloud computing marks a shift from localized computing setups to remote and highly scalable IT resources, revolutionizing how computing services are accessed and utilized[1][13].

Virtualization

IBM initiated the history of virtualization in the 1960s by virtualizing its mainframe, CP-40, which was later converted into CP-67, which employed partitions to run multiple processes simultaneously [7]. The primary purpose of virtualization is the abstraction of physical IT resources to virtual ones, such as servers, storage, and networks [8]. Therefore, they can be manipulated faster and more efficiently, increasing flexibility and availability [9].

Virtual machine-based virtualization

Virtual machine-based virtualization is a process that abstracts the entire OS and emulates the hardware needed to create a virtual machine (VM). In its most basic form, a VM is an independent system with an application, an application environment, and its operating system referred to as the guest OS. As seen in Figure 1, a layer on top of the host machine's operating system called a hypervisor allows for multiple virtual machines per host or physical machine. This type 2 hypervisor is a hardware proxy, allowing guest operating systems to believe they are operating on dedicated hardware [2]. The complete ISA (instruction set architecture) is virtualized. Multiple operating systems can share the same hardware resources, and the virtualized representations of each resource are available to the VM [7].

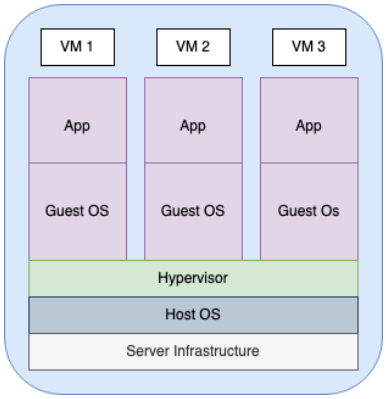


Figure 1. Basic virtual machine architecture. (adapted from P. S. Kocher [2]).

Containerization

Containerization is a virtualization technique that allows the creation of virtual hosting environments known as "containers" without requiring a virtual server deployment for each solution. Like virtual servers, shown in Figure 2, a container provides an environment with operating system resources to host software programs and other IT resources. Containers provide isolation for the process and the application. However, the OS's kernel, which all processes use, is shared by them all. The way this works is that Linux kernel resource isolation features like containers use control groups and namespaces to enable the execution of separate processes inside of a single Linux image. This relates to the original reason why, unlike virtual machines, individual applications do not have their operating system. This also means that virtual machines offer higher levels of isolation than containers. That being said, this very feature makes containers incredibly lightweight and portable. Given their lightweight design, containers allow you to run more than virtual machines (VMs) on a given hardware configuration. Containers allow you to use your hardware resources better [8][1].

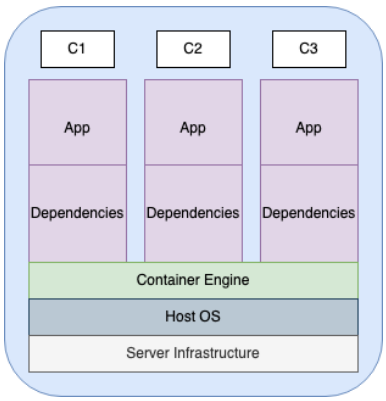


Figure 2. Basic container architecture. (adapted from P. S. Kocher [2]).

3 Energy Prediction

In the cloud computing environment, “application” refers to a collection of software services delivered over the internet. These services could either have monolithic architectures, which are single-tiered software applications where the user interface and data access code are combined into a single program from a single platform, to microservices, which are a group of small services, each running in their process and communicating lightweight, often with an HTTP resource API [2]. The system model maps the energy consumed by the deployment and maintenance of the applications. It considers the servers that do the computations, the storage devices that hold the data, the networking infrastructure that connects services, and the cooling systems that control the heat generated by these components, as visualized in Figure 3 [3]. Therefore, the energy consumption of cloud computing is primarily caused by the power it requires to operate the hardware infrastructure, and the amount of energy consumed by each one depends on various factors, such as utilization levels, workload distribution, and cooling efficiency. Research shows that servers are the primary consumers of energy in data centers, with around 60-70% of the total energy consumption attributed to server infrastructure [10]. Several studies have tried to find a way to reduce the energy consumption of cloud computing infrastructure. These include the use of more energy-efficient hardware components, the adoption of virtualization technologies, and the implementation of energy-efficient cooling systems [11]. In addition, efficient resource allocation algorithms can also reduce energy consumption by optimizing the allocation of resources based on workload characteristics [12].

Data Center Components	% Energy in use
Servers	50
Cooling systems	35
Networks	8
Others (UPS, PDU, among others.)	7

Figure 3. Data center energy usage by its different components (adapted from R. Buyya et al., 2024 [13]).

This paper compares and analyzes energy prediction in cloud computing, specifically examining the differences and similarities between VMs and containers. This research investigates how each technology approaches

energy consumption, its efficiencies, and the potential for optimization by evaluating current models and strategies utilized within both VMs and containers. The objective is more effective energy prediction and resource allocation within data centers.

4 Related work

This section explores meaningful research papers that tackle energy prediction and calculation for energy efficiency strategies associated with VMs and containerized systems. These papers show the current state-of-the-art energy management in cloud infrastructures, highlighting the challenges and solutions. Analyzing these techniques synthesizes the paper's analysis and results, which will be the research's core.

4.1 Machine Learning Approach for Energy Consumption Prediction in Datacenters

In their study, Merizig et al. [15] focused on proposing an architecture that includes the usage of machine learning techniques, specifically Support Vector Regression (SVR) and Artificial Neural Networks (ANN), to predict energy consumption in data center environments and evaluate historical energy usage data combined with various operating metrics as the primary goal is enabling an efficient predictive model. Verifying the diverse methods compared to the Support Vector Machine (SVM) method resulted in a performance that showed a high correlation with the actual data points. This study confirmed that machine learning algorithms, especially SVM and ANN, were potential tools for decreasing energy consumption in data centers. It could help cloud service companies manage their infrastructure efficiently. The importance of using time series methods for forecasting was highlighted, and it was suggested that incorporating additional features in future work could further enhance the predictive model features.

4.2 Energy Saving Strategy of Power System Cluster Based on Container Virtualization

Zheng et al. [16]worked to come up with ways that containers may be adopted within power system clusters, mainly through energy-saving container scheduling and migration algorithms, comparing the impact of these

algorithms on power system platforms and providing a deeper insight into how container orchestration and mobility can influence energy efficiency in data center environments. By exploring container scheduling and migration, the researchers found how container orchestration and mobility are potent factors in energy efficiency in data centers. Therefore, container scheduling and migration substantially contributed to the discussion about container scheduling for energy savings. Scheduling algorithms define how and where the containers are initially placed inside the cluster, as these algorithms consider optimum resource utilization and energy conservation while doing this. Contrarily, migration algorithms incorporate optimizations in terms of the dynamic adjustments of the containers and their repositioning across the cluster based on real-time energy efficiency assessment. Through this thorough investigation, the scientists conclude that container technology is one of the most effective solutions for making data center operations more energy efficient.

4.3 Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers

Nguyen Trung Hieu et al. [18] discuss reducing power consumption in cloud data centers through virtual machine (VM) consolidation algorithms combined with Multiple Usage Prediction algorithms. These algorithms aim to predict future resource utilization in servers to increase the efficiency in the energy of cloud data centers by reducing unreasonable migrations and allocating resources that won't be used efficiently through an effective user management system. Hence, identifies overloaded servers that need VM migration and underutilized servers that can be switched to low-power mode for energy efficiency. VMCUP-M algorithm is a proactive algorithm based on a multi-purpose prediction scheme that employs historical use data and, therefore, chooses the virtual machines that consume most of the server's resources as the principal reason for its performance without increasing the power state changes. by leveraging historical data and forecasting future resource usage, VMCUP-M optimizes resource allocation, reduces energy consumption, and enhances SLA compliance. The algorithm's performance evaluation demonstrates its effectiveness in minimizing migrations, power state changes, and the number of active servers, highlighting its potential to improve energy efficiency in cloud environments significantly.

4.4 Kepler: A framework to calculate the energy consumption of containerized applications

The work in [17] proposed by Amaral et al. presents how to calculate the energy consumption of applications running in containerized environments with the core of the framework Kepler by using BPF (Berkeley Packet Filter)-based monitoring, that entails the monitoring of real-time data, which can be applied to different areas like CPU usage, memory consumption, network I/O, and other performance indicators that will directly or indirectly impact on energy consumption. Accurate data and analysis of container energy consumption patterns to predict future energy consumption might enable energy management and optimization of the cloud infrastructure and data center environments. Therefore, it provides valuable information for data center operators and cloud service providers. It allows them to identify which applications consume more energy, optimize resource allocations based on actual energy usage, and enhance their operation's sustainability [5].

5 Results and Discussion

In this paper, the proposed approach to analyze the current state of the art in energy prediction on the cloud is the comparison between VMs and containers. This section compares the key features of each methodology used in the analyzed core papers, focusing on energy prediction for Virtual Machines (VMs) and containers. Therefore, the main focus of this section is the differences between each architecture and how they influence the prediction methodology and final prediction result.

From the comparative table Figure 4, papers referring to VMs and others dealing with containers provide some major insight into energy consumption in cloud computing environments. Firstly, in the context of containers, Amaral et al. [17] and Zheng et al. [16] argued that this kind of architecture can help quite a lot in monitoring energy usage at a very fine accuracy level because of microservices utilization. This provides easy access and computation for real-time and historical data analysis at a container level, consequently delivering admirable granularity when calculating energy usage and demonstrating great potential for adaptive management strategies through container orchestration. This is why the predictive model's scalability, flexibility, and simplicity are born from the

Feature/Aspect	Amaral et al. (Containers)	Zheng et al. (Containers)	Nguyen Trung Hieu (VMs)	Merizig et al. (VMs)
Data Utilization	Real-time system metrics for container performance.	Real-time and historical data for container management.	Historical data and operational metrics for VMs.	Historical energy usage and operational metrics.
Detail level	Container-level energy consumption insights.	Container-level analysis with emphasis on scheduling and migration.	Aggregated VM-level energy usage and optimization.	Data center-wide energy consumption predictions.
Scalability and Flexibility	High, due to the lightweight nature of containers.	High, adaptable to dynamic container orchestration.	Moderate, depending on the VM management platform.	High, models can be retrained with new data.
Accuracy/Reliability	High accuracy in calculating current energy consumption.	Reliability in achieving energy savings through management strategies.	Predictive accuracy is dependent on model quality and data.	High predictive accuracy, subject to feature selection and model tuning.
Potential for Predictive Modeling	Indirect; real-time data can inform predictive models.	Indirect; focuses more on management strategies than prediction.	Direct; uses predictive models for VM management.	Direct; core focus on developing predictive models.
Detail Level of Prediction	High due to microservice architecture.	High, supported by container orchestration.	Lower due to monolithic applications.	Lower, due to VM abstraction layers.
Predictive Model Complexity	Potentially simpler due to direct resource usage mapping.	Simpler, aligning with container granularity.	More complex due to the hypervisor abstraction layer.	More complex, accommodating VM abstraction complexities.

Figure 4. Comparison table between VMs and containers core papers.

straightforward mapping of the resources consumed. In contrast, the approaches defined by Nguyen Trung Hieu et al. [18] and Merizig et al. [15] on VMs exhibit a complex predictive model due to the level of abstraction of VM layers of the whole infrastructure. Therefore, the detailed level of prediction may be lower because the prediction may only take into some of the components. Nevertheless, the potential for a predictive model is "direct" due to their methodology, which is already based on predictive models that can be retrained with new data and present a high predictive accuracy of machine learning models for energy management. Overall, these findings evidence the strengths and limitations of each virtualization technique and the application of energy management strategies. Containers exhibit flexibility and efficiency using real-time energy tracking, while VMs offer robust, though potentially more complex, predictive modeling capabilities. Furthermore, both architectures can obtain accurate energy calculation and prediction results to manage and possibly use prediction models that improve energy optimization [15][16][17][18].

6 Conclusions

In the context of cloud computing, where energy consumption is of greater importance, VMs and containers play crucial roles in the cloud's ecosys-

tem, as they would in energy prediction and optimization. The comparison of the techniques shows the necessity of using both paradigms strengths and not favoring one over the other in energy efficiency strategies. Although table Figure 4 shows containers have a less complex model to predict or calculate energy [16][17], this can be disputed because the VMs techniques need to consider the whole data center infrastructure [15][18]. Containerization techniques are less complex, only calculating or predicting the energy of their environment and therefore complicating the prediction of the whole infrastructure's energy consumption [16][17]. In summary, this study prompts us to examine the undisputed coexistence and interdependence of both VMs and container technologies for sophisticated power forecasting and control under cloud computing platforms. Combining the strengths of each of the technologies — detail-oriented energy management of containers and purposeful predictive models of VMs — it is possible to create a more robust and coherent energy optimization strategy in cloud infrastructures. In the end, even though not all the papers have a robust prediction model, the calculations used help develop a future predictive model that can accurately predict both ways of virtualization. For future work, a new architecture for energy management can be implemented using the that the strengths observed in both VMs and containers, covering both the micro-level operations and the macro-level infrastructure of the data center [15][16][17][18].

References

[1] T. Erl, R. Puttini, and Z. Mahmood, "3.2 Basic Concepts and Terminology," in Cloud Computing: Concepts, Technology Architecture, Prentice Hall, 2013, ch. 3, sec. 2.

[2] P. S. Kocher, "Docker Containers," in Microservices and Containers, Addison-Wesley Professional, Apr. 2018, ch. 5, ISBN: 9780134591728.

[3] International Energy Agency, "Electricity 2024: Analysis and Forecast to 2026," IEA Publications, Jan. 2024. [Online]. Available: www.iea.org

[4] E. Ahvar, A.-C. Orgerie, and A. Lebre, "Estimating Energy Consumption of Cloud, Fog, and Edge Computing Infrastructures," IEEE Transactions on Sustainable Computing, vol. 7, no. 2, April-June 2022.

[5] P. Singh and H. Chen, "Introducing Kepler: Efficient Power Monitoring for Kubernetes," Red Hat Emerging Technologies, 22 August 2023. [Online]. Available: https://next.redhat.com/2023/08/22/introducing-kepler-

efficient-power-monitoring-for-kubernetes/.
[6]S. Ghafouri, S. Abdipoor, and J. Doyle, "Smart-Kube: Energy-Aware and Fair Kubernetes Job Scheduler Using Deep Reinforcement Learning," in 2023 IEEE 8th International Conference on Smart Cloud (SmartCloud), Tokyo, Japan, 2023, pp. 154-163. doi: 10.1109/SmartCloud58862.2023.00035.
[7]S. M. Jain, "Linux Containers and Virtualization: A Kernel Perspective," 1st ed., Apress, Place of publication not identified, 2020. [Online]. Available: ISBN: 1-4842-6283-2, ISBN: 1-4842-6282-4
[8]T. Erl and E. Barceló Monroy, Cloud Computing: Concepts, Technology, Security, and Architecture, 2nd ed. Pearson, August 2023, 608 pages.
[9] M. Portnoy, Virtualization Essentials, 3rd ed. Sybex, May 2023, 336 pages.
[10] L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," Computer, vol. 40, no. 12, pp. 33-37, 2009.
[11] J. Keppler, O. Kennedy, and T. Olson, "Energy Efficient Data Centers: A Critical Review and Future Directions," Applied Sciences, vol. 9, no. 1, p. 43, 2019.
[12]A. Beloglazov and R. Buyya, "Optimization of Energy Consumption in Cloud Computing Environments," in Handbook on Data Centers, G. Folino, B. Di Martino, A. Giordano, and M. Calzarossa, Eds. Springer, 2013, pp. 871–895.
[13] R. Buyya, S. Ilager, and P. Arroba, "Energy-efficiency and sustainability in new generation cloud computing: A vision and directions for integrated management of data center resources and workloads," Softw. Pract. Exper., vol. 54, no. 1, pp. 24-38, Jan. 2024. [Online]. Available: https://doi.org/10.1002/spe.3248.
[14] B. A. Stradi-Granados, Cloud Computing for Engineering Applications, 1st ed., Springer Cham, 2020. [Online]. Available: https://doi-org.libproxy.aalto.fi/10.1007/978-3-030-40445-1.
[15] A. Merizig et al., "Machine Learning Approach for Energy Consumption Prediction in Datacenters," in Proc. 2020 Int. Conf. on Mathematics and Information Technology (ICMIT), Adrar, Algeria, Feb. 2020, pp. 142-148.
[16] R. Zheng, H. Wang, and H. Jin, "Energy Saving Strategy of Power System Cluster Based on Container Virtualization," 2020.
[17] M. Amaral et al., "Kepler: A framework to calculate the energy consumption of containerized applications," in Proc. 2023 IEEE 16th Int. Conf. on Cloud Computing (CLOUD), 2023.

[18]N. T. Hieu, M. D. Francesco, and A. Ylä-Jääski, "Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers," in IEEE Transactions on Services Computing, vol. 13, no. 1, pp. 186-199, Jan.-Feb. 2020, doi: 10.1109/TSC.2017.2648791.

Learning Hurdling Skills with Adversarial Motion Priors

Jinglin Yang
jinglin.yang@aalto.fi

Tutor: Nam Hee Kim

Abstract

We present a method based on Adversarial Motion Priors (AMP) that enables the discovery of natural-looking hurdling skills for physics-based characters. AMP offers an efficient approach to separate high-level task objectives from low-level motion styles, empowering physics-based agents to execute natural-looking motions. The naturalness of the motion is measured by a discriminator trained based on Generative Adversarial Imitation Learning (GAIL), which assigns a score to generated motions based on their style similarity to the reference motions sampled from the dataset. We introduce a method that incorporates random state initiation, early termination, and curriculum training into the vanilla AMP framework to facilitate the training process. We demonstrate the efficacy of properly designed early termination conditions in facilitating the character's understanding of hurdling rules, such as the need to cross a hurdle without contact. The introduction of random state initialization enhances policy robustness by enabling the character to tackle hurdling tasks from diverse starting motions and positions Curriculum training plays a pivotal role in supporting the character's skill acquisition, fostering perseverance in attempting higher hurdles rather than prematurely abandoning the challenge..

KEYWORDS: Animation; Physical simulation; Deep Reinforcement Learning

1 Introduction

Hurdling is a famous sport where athletes race down a straight or curved track, navigating a series of evenly spaced barriers known as hurdles, and reach the finish line. The objective of our study is to develop a humanoid agent that is capable of dynamically adjusting its movements to run and navigate hurdles with agility and efficiency under physics-based simulation.

To achieve this goal, we construct a simulated environment utilizing the NVIDIA Isaac Gym [5] with a humanoid and a hurdle bar positioned along the running track. We integrate adversarial motion priors (AMP) into our deep reinforcement learning framework, where AMP acts as an adversarial discriminator for imitating the reference motion clips provided. The motion priors is modeled as an adversarial discriminator utilizing a set of reference motions that constitute a desired motion style. It serves to evaluate the resemblance between simulated motions and the reference motion data. It Consequently, the agent undergoes training to accomplish specified high-level task objectives, while its low-level motions follow the behavioral style generalized from the reference motion data, thereby generating natural movements.

2 Related Work

This section provides some related work including character animation, physics-based simulation, and deep reinforcement learning. Currently, there are two prevailing methods for synthesizing natural motions for virtual characters: kinematics-based and Physics-based frameworks.

2.1 Kinematics-based Animation

In kinematics-based animation, motions refer to mathematical representations of the motion of objects or systems without consideration of the forces causing the motion [11]. These models describe the positions and velocities of objects over time based solely on their geometric and kinematic properties, such as shape, size, and constraints. By leveraging datasets of motion clips, controllers can effectively select the appropriate clip to play in a given scenario. Kinematic methods often yield superior motion quality compared when provided with high-quality data. How-

ever, kinematic models face constraints when tasked with synthesizing novel motions or strategies, particularly in the presence of complex environmental conditions. Additionally, the absence of physics considerations renders their outputs fundamentally unrealistic and unnatural in certain circumstances.

2.2 Physics-based Animation

Physics-based animation leverages fundamental principles of physics to simulate the interactions between objects and their environment, enabling realistic and accurate representations of dynamic phenomena. Animation based on this approach has been developed for decades, with the primary challenge persisting in the design of a robust controller. Many manually designed controllers have achieved great success such as locomotion [2]. However, manually designed controllers rely heavily on the insight related to the specific task, and the controller is difficult to generalize to different tasks. Tracking-based controllers have become a popular domain, where the controller learns to reproduce the motions by imitating the reference capture motion data. Furthermore, Deep reinforcement learning has been demonstrated to be capable of replicating diverse motions while accomplishing targeted tasks [7]. In our work, we leverage the Isaac Gym framework, which offers high-performance and high-fidelity simulation capabilities, for constructing the physics-based environments [5].

2.3 Deep Reinforcement Learning (DRL)

Reinforcement learning (RL) is a machine learning framework focused on iterative interactions between an agent and its environment to optimize cumulative rewards. By employing sequential decision-making processes, RL algorithms iteratively adjust the agent's actions based on received feedback, aiming to derive an optimal policy that maximizes long-term rewards. Deep reinforcement learning (DRL), a subfield of RL, incorporates deep neural networks into the solution, allowing the agent to take actions based on unstructured inputs instead of explicitly defining the state space. DRL has found success in implementing robust strategies for simulated agents in complex environments [4]). Noteworthy algorithms such as TRPO and PPO showcase the efficacy of this approach in attaining optimal solutions [3]. However, while many methods demonstrate the capability to generate physically plausible motions, achieving natu-

ralistic movements without reference data poses a challenge. The design of task rewards for natural movements relies heavily on human insights and can struggle to generalize to other motions or tasks. Addressing this challenge, the utilization of reference motion data in the training process has emerged, wherein the motion data informs the reward function in RL, aiming to minimize the discrepancy between the simulated character's motion and the reference motion [9]. Nonetheless, the impracticality of obtaining reference motion data across all applications limits the utility of this approach. Consequently, methods based on adversarial motion priors (AMP) have been developed to enable simulated characters to execute high-level tasks by combining multiple low-level motions that capture general behavioral features extracted from motion datasets [10].

3 System Overview

We model the character adapted from the Deepmimic [7], featuring 13 links and 34 degrees of freedom. To streamline the experimental setup, a simplified rectangular prism serves as the hurdle bar. Our dataset comprises one motion clip depicting running movements, serving as reference motion data. The primary methodology employed for training the simulated humanoid to execute the hurdling task is the Adversarial Motion Priors (AMP) framework. Specifically, the agent is trained not to precisely replicate provided motions but to adopt general styles extracted from the reference motion data. This approach enables the agent to perform complex tasks utilizing various motions that resemble the motion clips in the dataset.

Figure 1 demonstrates the general framework of our model. In the motion dataset, each motion is represented as $m^i = \{\hat{q}_t^i\}$, where $\hat{q}_t^i$ is the pose at the sequence t. A policy controls the agent by generating motions based on the given current states s_t and the goal of the task, which is represented as $\pi(a_t|s_t, g)$. The action would result in a new state s_{t+1} and a reward r_t for applying the actions. The objective of this agent is to learn a policy that maximizes the expected discounted rewards, consisting of the weighted task reward r_t^G and the weighted motion style reward r_t^S which is specified by the adversarial discriminator. The task reward function $r_t^G = r^G(s_t, a_t, s_{t+1}, g)$ defines the high-level objective for the agent to accomplish, in our case, running along the trace and jump over the hurdle. The style reward $r_t^S = r^S(s_t, s_{t+1})$, which measures the naturalness

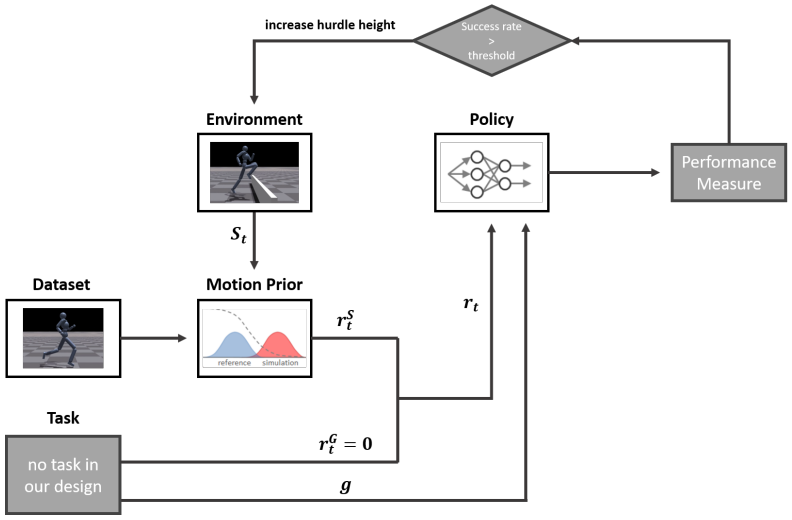


Figure 1. Overview of the system

of the motions generated by comparing with the given reference motion data. In conclusion, the task reward encourages the policy to complete the high-level task, while the style reward encourages the policy to produce motions in a way that resembles the given reference motions. In our method, we adopt a single-clip imitation, thus the task-rewards is set to 0. The system evaluates the performance of the current policy by measuring whether the character crosses the hurdle bar at a success rate that exceeds a predetermined threshold. If the success rate surpasses this threshold, the system adjusts by setting a new, higher hurdle bar for subsequent training.

The training procedure employs the proximal policy optimization (PPO) algorithm, which facilitates stable learning within the defined framework by avoiding to have too large policy updates. PPO updates the policy conservatively by measuring how much the current policy has changed compared to the previous one using a ratio calculation between them, which is then clipped within a range of $[1-\epsilon, 1+\epsilon]$, effectively removing the incentive for the current policy to deviate excessively from the old one.

3.1 Adversarial Motion Prior

In our design, we utilize the framework based on Adversarial Motion Priors (AMP), where the reward function is composed of two components:

$$r(s_t, a_t, s_{t+1}, g) = w^G r^G(s_t, a_t, s_t, g) + w^S r^S(s_t, s_{t+1}). \quad (1)$$

While designing the reward function r^G for the task in reinforcement learning is relatively intuitive, designing the style reward r^S to encourage natural movements poses a challenge. Instead of directly measuring

motion similarity between generated and reference motions, AMP adopts the Generative Adversarial Imitation Learning (GAIL) framework to discern whether the motion is a real sample from the dataset or a fake one generated by the policy. The AMP discriminator produces a general score indicating the similarity between generated and reference motions. AMP decouples motion style from task goals, enabling the accomplishment of tasks using various motion priors. Additionally, the same motion prior can be applied to perform different tasks. In our study, we employ running motion data as the motion prior, while the task is hurdling.

The discriminator is trained using sigmoid cross-entropy loss as the objective, which has been demonstrated to be effective in imitating a wide range of motions [10]. The discriminator evaluates input motion data and predicts a score, with a score of 1 indicating that the data is predicted to be sampled from the dataset, while a score of -1 suggests that the data is a synthetic sample generated by the policy. These scores serve as labels for the regression task, where 1 represents genuine data and -1 represents synthetic samples.

The observation spaces in our model encompass various aspects of the humanoid's state. These include the 3D positions of all body parts relative to the humanoid's local coordinate system, the velocity and angular velocity of the root (the pelvis of the humanoid), the local rotation and velocity of each joint, and the relative distance to the hurdle bar from the root. It's worth noting that not all features are utilized in training the discriminator. For instance, certain features such as the relative distance to the hurdle bar may have little relevance to assessing the similarity between generated and sampled motions. Hence, careful selection of relevant features is crucial to ensure the discriminator effectively evaluates motion similarity.

3.2 Network Architecture

In our design, we incorporate three networks: a policy network, a value function network, and a discriminator network, all having the same architecture. This architecture consists of two ReLU hidden fully-connected layers with 1024 and 512 units, respectively, followed by a linear output layer. For the policy network, the output value represents the mean of a Gaussian distribution $\mathcal{N}\left(\mu(s_t, g), \Sigma\right)$. The policy then generates actions sampled from this distribution given the current state s_t and goal g.

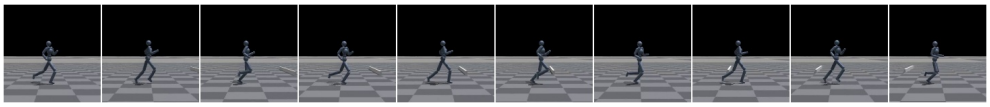


Figure 2. The character run away from the hurdle bar if early termination for detection of running away is not set

4 Training and Evaluation

4.1 Early Termination

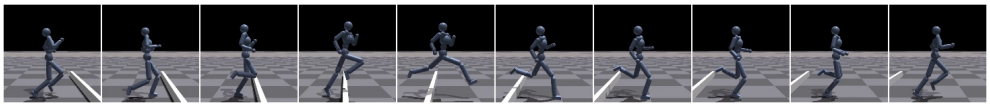
During the training, each episode is limited to a finite horizon, terminating either upon reaching the maximum horizon limit or when specific termination conditions are triggered. In locomotion tasks, a common early termination condition is fall detection, typically characterized by contact between the torso and the ground or when certain links fall below a predetermined threshold [8]. Once the early termination conditions are triggered, the rewards for the remaining episode are set to 0. Consequently, the policy is encouraged to avoid triggering these conditions. Another benefit of early termination is its ability to bias training in favor of samples that are more relevant to the ongoing tasks. For instance, when the humanoid is detected falling, it becomes challenging (and essentially another task) for it to stand back. Without early termination, the rest of the episode would be dominated by data where the humanoid is attempting to recover to its normal trajectory. In our model, in addition to the two existing termination conditions, we introduce two more early termination conditions. The first condition triggers termination if any body part makes contact with the hurdle bar, while the second condition initiates termination if the humanoid is detected deviating from the running trace. These additional conditions reflect the necessary action of cleanly leaping over hurdles in a hurdling race. Without the early termination condition for hurdle contact, the humanoid agent tends to exhibit behavior where it run normally and kick the hurdle bar away instead of cleanly jumping over it. In addition, without the early termination condition for detecting when the humanoid agent goes off track, it tended to bypass the hurdle bar rather than jumping over it.

4.2 States Initialization

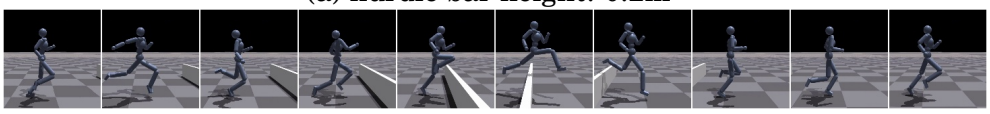
In our model, we employ random state initialization. Similar strategies have been employed and proven to be efficient in learning from demonstration samples [6]. At the beginning of each iteration, the initial state of the agent is randomly sampled from the motion clip dataset, rather than starting from a fixed state. This approach offers several advantages. Firstly, it allows the policy to learn later phases before mastering the previous ones, as opposed to learning the motion sequentially. Secondly, it encourages the exploration of challenging motions. Without this randomized initialization, the agent may become stuck in receiving rewards retrospectively until it reaches a high-reward state. Prior to encountering such a state, the policy has no way of knowing that this state is favorable, and potentially leading to convergence to local optima. Randomized initialization states address this issue by encouraging the exploration of potentially favorable states, even before the policy has learned the necessary strategies to reach them. This approach efficiently leverages information from the reference motion dataset, ensuring comprehensive exploration and utilization of the available data. Additionally, we implement randomization of the initial distance to the hurdle bar, which we demonstrate enhances the robustness of our policy. Without this setting, the humanoid is more prone to task failure when the hurdle bar is slightly adjusted closer to or further away from the starting position.

4.3 Curriculum Training

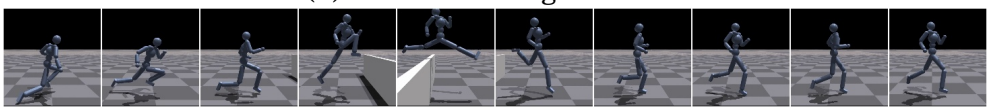
The key insight of curriculum training is to imitate the learning process of humans that follows the easy-to-hard sequence by allowing models to start learning with easy samples first and gradually progress to complex samples and knowledge [1]. It's easier for the model to learn easier tasks before it has the capability to complete the more difficult tasks. In our model, we refer to the insights of the curriculum training, to initially set a short hurdle bar first, and increase the hurdle bar height gradually when the success rate of the humanoid crossing the hurdle bar exceeds a certain predefined threshold. This trick is the key for the humanoid to master the techniques required to cross the hurdle bar. Similar approaches have already been employed in training skills such as high jump, where the height of the bar increases as the accumulated reward surpasses a certain threshold [12]. We have tested that if the initial hurdle bar is set too high,



(a) hurdle bar height: 0.2m



(b) hurdle bar height: 0.5m



(c) hurdle bar height: 0.8m

Figure 3. Snapshots of motions from the trained policies with increasing hurdling bar height

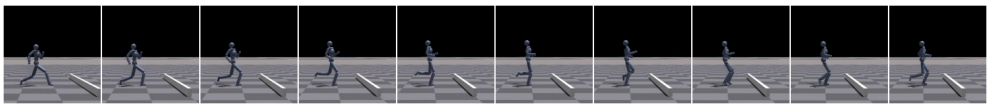


Figure 4. The character would stop moving forward when encountering a hurdle bar that is initialized too far away from its starting position

the humanoid would not learn or even attempt to cross the hurdle bar.

4.4 Task

In our approach, we employ single-clip imitation, where there is only style-rewards from the motion prior with the absence of task-rewards. Under this setting, the only task for the character is to imitate the motion from the dataset as long as possible. Consequently, the character earns higher rewards the longer it "survives" in the simulated environment. Despite the absence of explicitly defined tasks, the character remains motivated to overcome hurdles, as the presence of a hurdle bar obstructs the character's ability to accurately imitate the reference motion. However, we observed that if the hurdle bar is positioned too far from the character, it may tend to prioritize mimicking the running motion without making substantial forward progress, as depicted in Figure 4.

5 Conclusions

We present an architecture based on AMP for learning hurdling skills with a physically simulated character. We further demonstrate efficient training incorporating early termination, random states initialization, and

curriculum training. While our current implementation utilizes the motion data from running, we believe the incorporation of motion data from jumping could enhance the learning process and produce more natural-looking motions. AMP takes advantage of large datasets, and provide a mechanism for automatically selecting appropriate motion clips for imitation. Currently, our work has demonstrated that the character could cross one hurdling bar, We envision extending our approach to incorporate multiple hurdling bars and to enable the character to execute the entire hurdle sport process seamlessly.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[2] Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Generalized biped walking control. *ACM Trans. Graph.*, 29(4), jul 2010.

[3] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo, 2020.

[4] Libin Liu and Jessica Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Trans. Graph.*, 37(4), jul 2018.

[5] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[6] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations, 2018.

[7] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, July 2018.

[8] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.*, 35(4), jul 2016.

[9] Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.*, 36(4), jul 2017.

[10] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40(4):1–20, July 2021.

[11] H. Ruder, T. Ertl, K. Gruber, M. Günther, F. Hospach, M. Ruder, J. Subke, and K. Widmayer. Kinematics and dynamics for computer animation. In Sabine Coquillart, Wolfgang Straßer, and Peter Stucki, editors, *From Object Modelling to Advanced Visual Communication*, pages 76–117, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[12] Zhiqi Yin, Zeshi Yang, Michiel van de Panne, and KangKang Yin. Discovering diverse athletic jumping strategies, 2021.

Overview of different migration strategies from monolithic architecture to microservices architecture

Joona Munukka
joona.munukka@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

Monolithic architecture has been the default choice for many software systems for a long time. Recently, its place has been taken by microservices architecture. With the popularity of microservices architecture rising, migrations from monolithic architecture to microservices architecture have become a standard practice. This paper provides an overview of migration strategies from monolithic to microservices architecture. It introduces the concepts of monolithic and microservices architecture and covers several migration strategies, including Big Bang migration and Incremental migration. The conclusion emphasizes the importance of thorough analysis in assessing the benefits and challenges of the transition. Overall, this paper offers insights into the complexities and considerations in transitioning from monolithic to microservices architecture.

KEYWORDS: microservices architecture, monolithic architecture, migration strategies, big bang migration, incremental migration, strangler fig, parallel run

1 Introduction

Monolithic and microservices architectures offer contrasting approaches to software development and deployment. A microservice is a compact application capable of independent deployment, scaling, and testing, designed with a singular responsibility [21]. In contrast, a monolith is large in size, and its modules lack the capability for independent execution [6].

The popularity of microservices has surged in recent years. Microservices popularity extends beyond the adoption by new services, as even legacy monolithic services transform to align with the trend of microservices architecture [22]. However, migrating from a monolithic to a microservices architecture is not a simple task. Without proper guidance, migration can lead to significant challenges, potentially resulting in a microservices solution that is less effective than the original monolithic. An example of such a difficulty is the creation of "nanoservices", where an exaggerated approach introduces more overhead than benefits. As a result, integrating appropriate strategies and patterns is crucial for a successful migration.

Different migration strategies and patterns offer distinct benefits, making choosing the most suitable approach for a project challenging. Factors such as code base size, application uptime requirements, team size, and project deadlines may influence the selection of a migration strategy.

This paper aims to introduce several different migration strategies, highlighting their benefits and problems. Section 2 first introduces the microservices and monolith architectures, explaining their advantages and disadvantages to establish a foundational understanding. Section 3 presents an overview of several migration strategies. Section 4 compares the covered migration strategies and discusses the findings. Finally, Section 5 provides the conclusion.

2 Microservices vs. Monoliths

2.1 Microservices Architecture

Definition and Characteristics

Microservices architecture is structured around the collaboration of multiple microservices [16]. Figure 1 illustrates the microservice architecture. Microservices represent independently deployable services that encapsulate specific business functionalities, allowing for the construction of complex systems from modular building blocks [15]. Each microservice has a single responsibility, is self-contained, and exposes its functionality through Application Programming Interfaces (API) while abstracting implementation details [4]. These services, modeled around distinct business domains, operate autonomously and communicate via networks, providing a flexible approach to problem-solving [15]. The autonomy of each microservice increases its availability and enables it to be tolerant of failures [20]. Moreover, they are technology-agnostic, enabling teams to choose the most suitable tools and languages [15].

Microservices must be configurable to address different usage scenarios [7]. From an external perspective, each microservice is a black box, hosting business functionality accessible via networked endpoints [15]. Internal implementation details are concealed, and microservices avoid shared databases, encapsulating their own data storage. Updates and changes to the system only require modifying the component related to a specific microservice or container rather than the entire system [20].

Benefits and Challenges

Microservices offer numerous advantages, including evolutionary development, open standards compliance, high development velocity, reusability, flexibility, and versioning capabilities [7]. They facilitate agile development methodologies by allowing smaller, independent teams to work on individual components [4]. Additionally, microservices enable the creation of large, scalable applications, enhance fault tolerance, and foster a culture of automation and decentralized processes [20].

However, transitioning to microservices presents challenges such as deployment complexity, automation requirements, and data management complexities [4]. Organizational changes pose another hurdle when adopting microservices [7]. Furthermore, inter-process communication intro-

duces overhead that may affect the performance [4]. Overcoming these challenges demands a shift towards a DevOps culture, cross-functional teams, and investment in dynamic infrastructure [7].

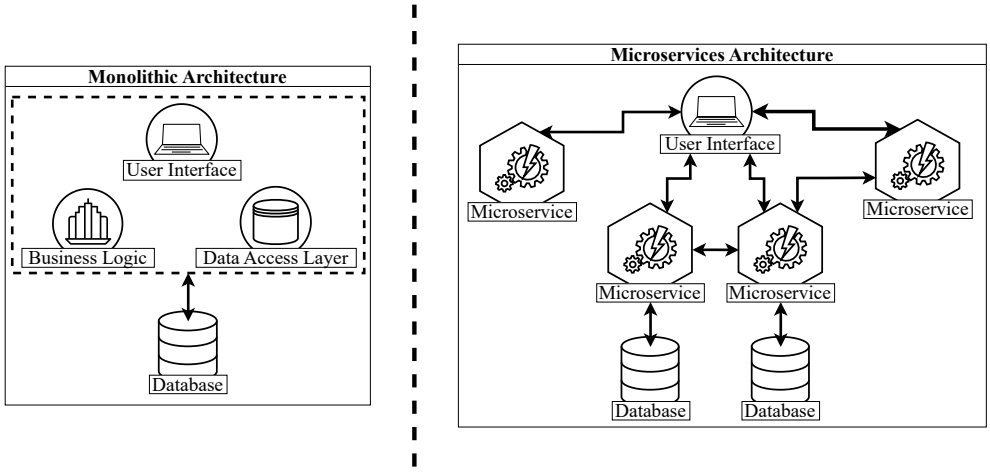


Figure 1. Visualization of Monolithic and Microservices Architectures [20]

2.2 Monolithic Architecture

Definition and Characteristics

A monolith encompasses all functionalities of a system deployed together, with variations including single-process, modular, and distributed monolithic architectures [15]. Visualization of monolithic architecture is shown in Figure 1. Typically reliant on a single development technology, monolithic architecture poses challenges in development and deployment efficiency, requiring rebuilding and deploying the entire system for any changes [20].

Benefits and Challenges

Monolithic architecture offers simplicity in deployment, intra-process communication, and familiarity in development practices [4]. It provides a stable platform for large organizations [20] and facilitates code reuse within the monolith itself [15]. However, monoliths are rigid systems that struggle to adapt to new requirements and incur high scaling costs [20]. They suffer from complexity, unintended consequences, longer start-up times, and scalability issues as the application grows [4]. Despite prevailing negative perceptions, monolithic architecture remains a valid and sensible choice, offering benefits often overlooked in favor of newer approaches [15].

3 Migration Strategies

3.1 Big Bang Migration

Big Bang migration involves a complete overhaul of an existing monolithic system, transitioning it into microservices in a single event [8]. Users continue to use the old system until the new one is fully developed, tested, and deployed [1]. This approach is complex and resource-intensive, demanding a code freeze on the legacy application during the re-architecting process [5]. Development efforts may span several years, with data migration occurring at the project's end [12].

Factors such as contractual requirements or the small size of the old system may justify waiting for the transformation to complete, although longer transformation periods pose challenges in maintaining the old system without changes [12]. It is important to note that the Big Bang migration does not deliver any benefit until it is complete [17]. Despite its challenges, the Big Bang migration is typically cheaper than alternative options and may be preferred by some organizations for project management purposes [12]. The Big Bang migration has potential benefits but is relatively uncommonly used [5].

3.2 Incremental Migration

Incremental migration offers a structured approach to transitioning to a microservices architecture by breaking the process into smaller, manageable steps, facilitating learning, and minimizing risks [16]. This strategy introduces microservices gradually, guided by the determination of whether re-architecting provides immediate value to the business and application users [5]. Business needs drive the selection of services for independent development, aiming for recognizable benefits while balancing transition risks.

The incremental migration process typically unfolds through four phases: defining transition goals, specifying the scope and level of architectural changes, preparing for resource readiness, and adjusting critical development practices [5]. This approach allows for a gradual learning curve, increased confidence, and immediate impact on targeted services due to active development. However, risks such as co-existing architectures, resource constraints, and maintaining quality standards amidst di-

verse technologies should be considered.

By incrementally splitting microservices, organizations unlock their value gradually, rather than waiting for a big-bang deployment [16]. This iterative process enables the identification of quick wins, builds momentum, and reduces the impact of mistakes, fostering a smoother transition to a microservices architecture.

3.3 Strangler Fig Pattern

The Strangler Fig Pattern offers a methodical approach to refactoring monolithic applications into microservices by gradually replacing functional domains [22]. Dividing the application into functional domains, each domain is replaced with a microservices-based implementation, allowing the old and new systems to coexist. Inspired by the growth pattern of a fig tree, the new system wraps around the existing one, incrementally replacing it over time [16].

Implementation of the pattern involves identifying parts of the existing system for migration, implementing the functionality in new microservices, and rerouting calls from the monolith to the new microservices [16]. The implementation can be divided into three phases shown in Table 1.

Phase	**Description**
Transform	A parallel site is created and incrementally transformed while coexisting with the original.
Coexist	Both the original and migrated sites coexist, with gradual redirection to the new site for newly implemented functionality.
Remove	Old functionality is removed, and traffic is redirected away from that part.

Table 1. Phases of the Strangler Fig Pattern [22]

The Strangler Fig Pattern promotes an incremental approach to microservices adoption, offering flexibility and reversibility [22]. This strategy enables the gradual migration from monolithic systems to microservices, allowing for the new system to grow and potentially replace the old one entirely [16]. Importantly, it enables pausing or even stopping the migration while benefiting from current implementations. It is particularly useful when migrating functionality without wanting to make significant

changes to the existing system, such as with black box systems or concurrently developed monoliths.

By monitoring progress and adjusting implementations with each new microservice, organizations can adapt the transition to specific environments or requirements [8]. Overall, the Strangler Fig Pattern provides a flexible and manageable pathway to transitioning from monolithic architectures to microservices [22].

3.4 Parallel Run

Parallel run strategy facilitates the simultaneous operation of both old and new implementation of the same functionality in the production environment, enabling a comparison of results to ensure equivalence [16]. This approach involves calling both implementations concurrently, with one considered the source of truth until verification confirms the reliability of the new one.

Utilized for verifying both functional equivalence and non-functional aspects of the new implementation, parallel run ensures that outputs from both implementations match given the same inputs [16]. It is particularly beneficial for high-risk changes and situations with limited testing time, allowing comprehensive verification.

In practice, requests are processed concurrently by the old monolith and the new system, with the system validating its response against the monolith's response to ensure consistency [18]. Metrics are generated to monitor consistency, aiding the gradual rollout per endpoint and facilitating feedback collection.

While parallel run offers advantages such as realistic testing scenarios, gradual rollout, and performance evaluation before full deployment, it also presents challenges like increased load, complex result comparisons, and GDRP compliance concerns [18]. Despite these drawbacks, it remains a valuable strategy for ensuring the reliability and equivalence of new implementations in complex migration scenarios.

More details on how the parallel run is used as part of a migration strategy can be found in the following sources. The parallel run is the key part of the migration strategy in [18]. The parallel run is employed in two distinct phases of the migration process outlined in [14].

3.5 Data Migration Strategies

Transitioning from a monolithic architecture to microservices requires careful consideration of data migration strategies [2]. Database migration is a critical aspect of transitioning to microservices, as monolithic systems typically rely on a single shared database, whereas microservices often have their own databases. Extracting and restructuring data to align with the requirements of individual microservices demands careful planning and robust migration strategies.

However, decomposing the data layer is notably difficult, with many preferring to adopt microservices while retaining a centralized legacy database [3]. The complexity of data migration often leads teams to avoid it altogether [23]. Yet, database migration significantly impacts the success probability of Microservices Architecture projects [2].

Addressing the migration of these databases requires careful consideration, with recommendations suggesting the splitting of data to allow each microservice access to its private database [19]. Failure to replicate the database can result in it becoming a single point of failure, thereby compromising the availability of the application [9].

There are three main options to execute this extraction: splitting the database first, then the code; splitting the code first, then the database; or splitting both simultaneously [16]. Each option has its benefits and challenges, influencing the overall success of the migration process.

The first approach involves splitting the databases first, creating separate schemas or databases to logically isolate data related to different services. While this enables early detection of issues and independent evaluation of changes, it may not yield immediate benefits and requires careful consideration of performance and consistency tradeoffs.

Alternatively, teams may choose to split the code first, allowing for early deployment of independently deployable code artifacts and a simplified understanding of data requirements. However, this approach risks persistent issues with shared databases and delays in identifying performance or data consistency issues.

Finally, teams have the option to split both the code and data together in one significant step. While this approach may take longer to assess the impact, it is generally advisable to avoid it and instead consider splitting either the database or code first.

4 Discussion

Migrating from a monolithic architecture to a microservices architecture is a complex task. It is crucial to thoroughly analyze the potential benefits and challenges of migrating to microservices. While the microservices architecture is often perceived as the superior option, there is a growing recognition that it may not always be the most suitable choice [5, 11, 16]. In a recent case, reverting to monolithic architecture helped to achieve higher scaling, resilience, and reduced costs [13]. Therefore, migrating to microservices should only be pursued with a clear rationale to avoid wasting resources. The first step of the migration should always be to evaluate whether the migration should even be done.

If the benefits of the microservices outweigh those of monolithic architecture, proper planning is required to capture the benefits. The Big Bang migration is usually not advocated [8] and sometimes even recommended to never be used [17]. That is because even though the idea of Big Bang, migrating the entire application in a single rewrite, appears simple, it is complex to implement. However, with extensive experience, Big Bang migration can become a desirable option and should be considered when planning the migration.

Incremental migration is often a more practical approach. Its goal is to minimize the needed resources, whereas the Big Bang migration minimizes the duration [10]. There are many strategies to implement incremental migration. Strategies, such as Strangler Fig and Parallel Run, can be utilized to facilitate the transition, and a combination of different incremental migration strategies may offer additional benefits. In the migration planning phase, it could be a good idea to experiment with the idea of picking the best of multiple migration strategies to form the strategy used for migration.

Regardless of the chosen strategy, data migration is always a challenging and critical aspect of the process. Failing to execute it successfully could lead to an inability to realize the full benefits of microservices. Therefore, careful planning and evaluation of different methods for splitting the database are essential for a successful data migration.

5 Conclusion

The transition from monolithic architecture to microservices architecture is a significant trend in software engineering. This paper has provided an overview of several migration strategies and introduced the concepts of monolithic and microservices architectures. The discussed migration strategies include the Big Bang migration, incremental migration, Strangler Fig pattern, and the parallel run strategy, along with a chapter focusing on data migration.

It is crucial to thoroughly evaluate the benefits and challenges of microservices compared to monoliths before initiating a migration. While microservices offer numerous advantages, they may not always be the best choice. If migration to microservices is deemed beneficial, it is essential to carefully assess different migration strategy options and select one that aligns with the goals, constraints, and risk tolerance.

The Big Bang migration may offer simplicity, but it also presents significant risks and challenges. In contrast, incremental migration allows for a more gradual and controlled approach, enabling learning and adaptation throughout the migration. The Strangler Fig pattern and parallel run strategy provide effective frameworks for implementing incremental migration. However, they also come with their own set of challenges. Data migration, an integral part of the process, requires meticulous planning regardless of the migration strategy used.

In conclusion, a successful migration to a microservices architecture necessitates comprehensive planning to address the technological and organizational challenges. By selecting the right migration strategy, creating a thorough plan, and prioritizing data migration, organizations can execute the migration successfully while reaping the benefits of microservices architecture.

In the future, further research is necessary to explore the transition from monolithic to microservices architecture. This research is crucial for the advancement of new and enhanced migration strategies and the identification of specific instances where migration yields significant benefits. Given the evolving nature of this field, it is critical to re-evaluate previous challenges in light of current research findings. With increased knowledge, migration strategies once deemed complex and unfeasible may become well understood and the preferred choice. This highlights the need for continuous re-evaluation and adaptation within the field.

References

[1] Mohamed Abouahmed and Omar Ahmed. *Machine Learning in Microservices: Productionizing microservices architecture for machine learning solutions*. 1st edition. S.l.: Packt Publishing, 2023. ISBN: 978-1-80461-214-9.

[2] Abdullah Alshammari et al. "High-performance computing-enabled probabilistic framework for migration from monolithic to microservices architecture using genetic algorithms". In: *Soft Computing* (31st Oct. 2023). DOI: 10.1007/s00500-023-09336-w.

[3] Saša Baškarada, Vivian Nguyen and Andy Koronios. "Architecting Microservices: Practical Opportunities and Challenges". In: *Journal of Computer Information Systems* 60.5 (2nd Sept. 2020), pp. 428–436. DOI: 10.1080/08874417.2018.1520056.

[4] Grzegorz Blinowski, Anna Ojdowska and Adam Przybylek. "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation". In: *IEEE Access* 10 (2022). DOI: 10.1109/ACCESS.2022.3152803.

[5] Karoly Bozan, Kalle Lyytinen and Gregory M. Rose. "How to transition incrementally to microservice architecture". In: *Communications of the ACM* 64.1 (Jan. 2021), pp. 79–85. DOI: 10.1145/3378064.

[6] Nicola Dragoni et al. "Microservices: yesterday, today, and tomorrow". In: (2016). Publisher: arXiv Version Number: 4. DOI: 10.48550/ARXIV.1606.04036.

[7] Bob Familiar. *Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions*. Microservices, Iot, and Azure. Berkeley, CA: Apress, 2015. ISBN: 978-1-4842-1275-2.

[8] Ken Finnigan. *Enterprise Java microservices*. Shelter Island, NY: Manning, 2019. 253 pp. ISBN: 978-1-61729-424-2.

[9] Augusto Flávio A. A. Freire et al. "Migrating production monolithic systems to microservices using aspect oriented programming". In: *Software: Practice and Experience* 51.6 (June 2021), pp. 1280–1307. DOI: 10.1002/spe.2956.

[10] Jonas Fritzsch et al. “Towards an Architecture-Centric Methodology for Migrating to Microservices”. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops*. Ed. by Philippe Kruchten and Peggy Gregory. Vol. 489. Series Title: Lecture Notes in Business Information Processing. Cham: Springer Nature Switzerland, 2024, pp. 39–47. ISBN: 978-3-031-48549-7 978-3-031-48550-3. DOI: 10.1007/978-3-031-48550-3_5.

[11] Dimitrios Gravanis, George Kakarontzas and Vassilis Gerogiannis. “You don’t need a Microservices Architecture (yet): Monoliths may do the trick”. In: *2021 2nd European Symposium on Software Engineering*. ESSE 2021: 2021 2nd European Symposium on Software Engineering. Larissa Greece: ACM, 19th Nov. 2021, pp. 39–44. DOI: 10.1145/3501774.3501780.

[12] Pavel Hruby and Christian Vibe Scheller. “Microservice Architecture Patterns for Enterprise Applications Supporting Business Agility”. In: *Proceedings of the 29th Conference on Pattern Languages of Programs*. PLoP ’22. USA: The Hillside Group, 2nd Nov. 2023, pp. 1–21. ISBN: 978-1-941652-18-3. URL: https://dl.acm.org/doi/10.5555/3631672.3631698.

[13] Marcin Kolny. *Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%*. Prime Video Tech. 22nd Mar. 2023. URL: https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90 (visited on 01/04/2024).

[14] Alan Megargel, Venky Shankararaman and David K. Walker. “Migrating from Monoliths to Cloud-Based Microservices: A Banking Industry Example”. In: *Software Engineering in the Era of Cloud Computing*. Ed. by Muthu Ramachandran and Zaigham Mahmood. Series Title: Computer Communications and Networks. Cham: Springer International Publishing, 2020, pp. 85–108. ISBN: 978-3-030-33623-3 978-3-030-33624-0. DOI: 10.1007/978-3-030-33624-0_4.

[15] Sam Newman. *Building microservices: designing fine-grained systems*. Second Edition. Beijing: O’Reilly Media, 2021. 586 pp. ISBN: 978-1-4920-3402-5.

[16] Sam Newman. *Monolith to microservices: evolutionary patterns to transform your monolith*. First edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2019. 255 pp. ISBN: 978-1-4920-4784-1.

[17] Chris Richardson. *Microservices patterns: with examples in Java*. Shelter Island, NY: Manning, 2019. 490 pp. ISBN: 978-1-61729-454-9.

[18] Ali Sebzevari and Francesco Sarracco. *Parallel Run Pattern - A Migration Technique in Microservices Architecture*. Zalando Engineering Blog. 4th Nov. 2021. URL: https://engineering.zalando.com/posts/2021/11/parallel-run.html (visited on 18/03/2024).

[19] Davide Taibi, Valentina Lenarduzzi and Claus Pahl. "Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation". In: *IEEE Cloud Computing* 4.5 (Sept. 2017), pp. 22–32. DOI: 10.1109/MCC.2017.4250931.

[20] Freddy Tapia et al. "From Monolithic Systems to Microservices: A Comparative Study of Performance". In: *Applied Sciences* 10.17 (21st Aug. 2020). DOI: 10.3390/app10175797.

[21] Johannes Thönes. "Microservices". In: *IEEE Software* 32.1 (Jan. 2015). DOI: 10.1109/MS.2015.11.

[22] Victor Velepucha and Pamela Flores. "A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges". In: *IEEE Access* 11 (2023). DOI: 10.1109/ACCESS.2023.3305687.

[23] Xin Zhou et al. "Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry". In: *Journal of Systems and Software* 195 (Jan. 2023). DOI: 10.1016/j.jss.2022.111521.

How to browse the web without leaving evidence

Joona Sauramäki
joona.sauramaki@aalto.fi

Tutor: Tuomas Aura

Abstract

There are multiple entities that want to track users on the internet, ranging from malicious parties to advertisement networks, that want to monetize user data for profit.

Browsing the internet relies on multiple technologies that each have different problems with privacy and security. Browsing leaves evidence of the session on the user's device and on the accessed servers. Different entities also try to fingerprint browsing sessions to identify users.

There are multiple solutions for anonymizing sessions. VPN hides client IP, private browsing modes leave less evidence on the device and the TOR network provides the most comprehensive protection for privacy.

Optimal privacy can be achieved by combining different solutions to provide the best protection. However, it's essential to acknowledge potential trade-offs in user-friendliness. Individuals must determine the level of protection they think is necessary.

KEYWORDS: Browser, forensics, incognito

1 Introduction

Users on the web are becoming more aware of tracking on the internet and how to protect against it. Now every major browser offers a private browsing mode, and some browsers are solely focused on protected and private browsing. VPNs are more popular than ever, but their benefits seem to be overestimated or misunderstood.

The problem with when browsing the web with the intention of not leaving any evidence is that the browsing session leaves evidence in multiple places without the user's knowledge. Even the private browsing modes often lead users to believe in a stronger protection than what is realistic. Private browsers also have had vulnerabilities and oversights that left data accessible after a browsing session. Evidence of the browsing session is stored by the browsers as client history, cache, cookies, and other places. Additionally, external entities can track generated DNS queries and activity on the web.

This paper goes through where browsers store information about a browsing session and how forensics methods could gather evidence to identify users and their activity on the web. Also, it discusses how to protect information leakage from DNS queries, ISP traffic logging, and website tracking.

Section 2 introduces different types of browsing artifacts that a browsing session can create. Section 3 presents different technologies that can be used to leave less evidence of the browsing session. Finally, section 4 provides the conclusion.

2 Browser Session Artifacts

Browsing the internet leaves behind traces of evidence about the session in multiple places. The browser stores information on the user's computer, every domain needs to be resolved with a Domain Name System (DNS) query, all of the data goes through an Internet Service Provider (ISP), and the websites deploy technologies to track clients. These can be split into evidence that is left on the user's computer and traces that are left on different cloud servers.

Digital forensics is a part of forensics that involves collecting, examining, and identifying digital evidence. Digital evidence comprises information that can be extracted from the user's computer and all artifacts

created by online activities to remote servers. Digital forensics is mostly used as a tool for law enforcement to gather evidence. The knowledge of forensic techniques also helps to understand how to protect against unauthorized snooping and tracking.

2.1 Local Files

All of the most common browsers leave behind local artifacts about a browsing session on the client machine. The most commonly known artifacts are the browsing history and user-downloaded files, but many other artifacts are generated in each session. For example, data typically stored by browsers on the file system includes, bookmarks, cookies, favicons, form autocomplete information, stored credentials, and cached files.

When the browser is running, additional information can be gathered through a RAM analysis. It is a method where computer RAM is dumped on a file and analyzed. The RAM can also be accessed if it is swapped to a pagefile on another storage medium which persists between system reboots. Ohana et al. [14] were able to extract information from RAM and pagefile about cached images, URL history, and usernames. They could do it with browsers running in private browsing modes.

Solomos et al. [16] proposed a method to track client sessions using the favicon cache. When the web server is first accessed the browser caches favicons of different attacker-controlled domain paths. After manipulating the cache, the server can track the client by testing the existence of the cached favicons.

There are multiple forensic tools [15] for collecting and interpreting information about browser activity. These tools make it easier to link, analyze, and combine evidence from different sources to get a better picture of the browser activity. Nalawade et al. [12] defined these advanced characteristics as integrated analysis, timeline analysis, analysis of search history, analysis of URL encoding, analysis of user activity, and recovery of deleted information.

When removing browser-created files they should be handled in a similar manner as when deleting any confidential data [18]. After deleting the browser files, there may still be methods to recover the data from the storage device. Hard drives only delete the indexes to the data, but it can still be recovered using forensic methods. When a file is deleted from a Solid State Drive, it runs a TRIM command for optimizing the drive that deletes the content but if the operation is disabled at the OS level,

recovery of data is much more likely.

Many browsers allow the installation of browser extensions. From a privacy perspective, the extensions are problematic because they often require access to the website information to be usable. They can also leave additional files to the client, which can leak information on user activity. Many browser extensions offer also synchronization and cloud-based services that can leak information. Starov et al. [17] analyzed browser extensions and found that 6.3% introduce privacy leaks accidentally or intentionally. There also has been research into extension security [5] and privacy [7, 9].

2.2 Web Server Tracking

Another way of tracking clients is tagging and fingerprinting [19], done by the accessed web servers. Tagging is where the web server puts something on the user's computer, like a cookie [10], that can be tracked across website access. In fingerprinting the webserver gather usually all available data about the browser connection and information to distinguish the connection from other clients. Especially fingerpring does not guarantee complete differentiability between all accesses, but is generally able to differentiate between most of the clients.

Tagging and fingerprinting do not directly expose any sensitive information but they can indirectly link data between different sessions to each other resulting in leaking information between sessions.

2.3 Name Service

Another privacy consideration is Domain Name System (DNS) [11]. It translates human-readable domain names to IP addresses. The DNS queries are usually unencrypted, although there are solutions like DNS over TLS and DNS over HTTPS.

The client's stub resolver is responsible for querying a DNS server. The DNS server gets access to many fields of data, most notably source IP address, port, and domain name. With the combination of source IP and port, the DNS server can even distinguish clients that share the same IP. This is common when using mobile data because they are often behind a Carried-Grade NAT. Some DNS servers may log queries, for example to provide intrusion detection systems.

For performance reasons, DNS relies heavily on caching at every level.

The stub resolver typically caches all of the queries. Anyone who has access to the computer has access to the information from the stub resolver. Even without direct read access to the stub resolvers cache, some fingerprinting and tagging techniques can query to cache to get information.

There are multiple methods of tracking clients based on cached DNS records. Kelin at Al. [8] proposed a technique where tracking users across websites is achieved by inserting a tracking snippet into a website that queries different name servers. Its goal is to put unique DNS cache data into the stub resolver by querying tracker-controlled DNS records. These records contain each time a randomly ordered set of A records of tracker-owned IP addresses. Websites can use this information to query the server to create statistically unique fingerprints. The fingerprint remains unchanged as long as the stub resolver keeps records cached.

DNS has also other privacy implications besides revealing information about accessed domain names. For example, the DNS system has other queriable fields such as MX records for email addresses. Based on the addresses one can deduct where the client is sending emails to. Although, you can only get information about domains and not specific users belonging to that domain.

2.4 Internet Service Providers

As all of the internet traffic passes through an Internet Service Provider (ISP) they can track it all. In many countries, it is mandated that the ISPs log the current owner of each connected IP, but they are not allowed to track any outgoing connections.

You could also assume that some ISP could be malicious and log all of the possible information which could be assumed of that any party could listen in on connections on the internet. As most of the internet traffic is encrypted it can not be accessed, but everybody can see every IP address that the client accesses and ISP can link this IP to the owner of the connection.

3 Hiding Browsing Artifacts

This section goes through different methods of hiding information about a browsing session. It is discussed how those methods work and what kind of data they can hide.

	FF	Safari	Chrome	IE
History	no	yes	no	no
Cookies	no	yes	no	no
HTML5 local storage	no	yes	no	no
Bookmarks	yes	yes	yes	yes
Password database	yes	yes	yes	yes
Form autocompletion	yes	yes	yes	no
User approved SSL self-signed cert	yes	yes	yes	yes
Downloaded items list	no	yes	yes	n/a
Downloaded items	yes	yes	yes	yes
Search box search terms	yes	yes	yes	yes
Browser's web cache	no	no	no	no
Client certs	yes	yes	yes	yes
Custom protocol handlers	yes	n/a	n/a	n/a
Per-site zoom level	no	n/a	yes	n/a

Figure 1. Public mode data access by Aggarwal et al. [1]

3.1 Private Browsing Modes

All of the major browsers provide some kind of private browsing mode. What complicates things more is that every browser has different implementations of private browsing modes with different names. The protections that the private browsing modes achieve and how they are represented to users often lead to misconceptions of their capabilities [20].

Aggarwal et al. [1] categorized the goals of private browsing modes into two categories, privacy from a local attacker and a web attacker. Local privacy means that the action of accessing a web server should not leave any traces on a local filesystem that can be later accessed by an attacker that has access to the client. This means that all of the local browsing artifacts and cache should be unchanged when using a private browsing mode.

Even though the browsing sessions in private mode should not change any local files many of the solutions violate this. Aggarwal et al. [1] found that many browsers violate this. The violations include modifying last accessed timestamps in SQLite databases, certificate authority certificates stored, and plugin-related data. Also, browsers generally use a cache for each incognito session that is deleted afterward. The cache could be analyzed when the session is open.

Privacy from web attackers is that it should be difficult for any web server to link activities in private mode the activities in public mode. This means the web servers accessed should not be able to fingerprint the browsers between these modes.

Private browsing modes can not protect against leaking the client IP

addresses as they are local-only solutions. These modes have solved the problem with browser extensions by generally disabling them by default but the user can opt-in to use them.

3.2 Virtual Private Network

VPN is a secure connection between a device and a network or two networks. This can be achieved in many different ways but generally in browser privacy VPN refers to a connection between the client and the network that is provided to the client as a service. The client connects to a VPN service that routes all of the internet traffic from the client over the VPN services network that forwards them to the internet. This means that the outgoing packet address belongs to the VPN service and not the client.

Virtual Private Networks (VPN) have become more popular in recent years partially because of the amount of marketing around them. These have been marketed to hide IP addresses and the ability to circumvent geoblocking. The problem with marketing these solutions is that they can contain misleading claims, overpromise, and exaggerate the information which can negatively affect the user's view of internet safety [2].

VPN hides the client IP address by rerouting the packets through the VPN provider's network. It makes it harder to identify between different VPN users as all of their packets are routed through the same IP addresses.

One of the problems with commercial VPNs is that all of the data goes through them and they have the same capability as ISPs to log information. VPN providers are also less regulated than ISPs on how they can operate.

There are also providers that offer free VPN services and some of them are malicious. There are services that market private and non-loggin policy but still log connections and sell the data to third parties making the benefits of a VPN non-existent [6].

3.3 The Onion Router

The Onion Router (TOR) [4] is open-source software that enables anonymous communication on the internet. TOR network consists of entry, middle, and exit nodes. The client's traffic enters through an entry node, is routed over multiple middle nodes, and then exits through an exit node.

TOR network relies on onion routing where the traffic is encrypted multiple times as it is passed over the nodes. Only the entry node knows the client's IP address and the exit node knows the destination address and nothing about the client. The middle nodes know only the previous and next hops in the chain. Exit nodes can eavesdrop on unencrypted traffic.

TOR relies on volunteers to operate these nodes comprising individuals, organizations, and non-profit entities. The volunteers enable the TOR network to be decentralized and have resilient infrastructure. As the network is run by volunteers there are malicious nodes present. TOR relies also on the fact that any entity should not be able to gain a significant portion of the network, which would enable the entity to trace the connection over the nodes.

Another problem with TOR is that there have been traffic analysis attacks [3]. In traffic analysis attacks the attacker typically controls the entry and exit nodes and based on the timing, packet size, and other information, correlates traffic flow on entry and exit nodes. Recently there have also been machine learning-based solutions for traffic correlations [13].

Even though TOR has many vulnerabilities, when everything works as expected and the attacker does not control any nodes, the network provides decent anonymity. Also, my TOR browser by default provides privacy-focused features. For example, it deletes cookies between sessions.

4 Conclusion

This paper reviewed different types of solutions for network privacy and how to leave as little evidence as possible of a browsing session. When browsing the internet evidence is left in multiple places to the user's device and the servers accessed. Browsing the internet is a complicated process involving multiple technologies is hard to maintain privacy as each of the technologies has the potential to leave evidence.

The most basic of protections is to use a private browsing mode that protects from others using the same computer. Private browsing modes do not provide any protection from leaking IP addresses or tracking done by third parties.

VPNs have many benefits but for privacy, they provide the ability to hide IP addresses. However, the VPN service provider still knows the IP

address and needs to be trusted to keep it secret.

TOR browser provides the most comprehensive protection against privacy. But it also has weaknesses that can be exploited to track users. However, most of these exploits are hard to execute.

Often privacy is the balance between convenience and privacy. By combining these solutions users can achieve good privacy on the internet even though most of them have some kind of flaws. Some of the flaws can be mitigated by using multiple solutions.

References

[1] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. An analysis of private browsing modes in modern browsers. In *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[2] Omer Akgul, Richard Roberts, Moses Namara, Dave Levin, and Michelle L. Mazurek. Investigating influencer vpn ads on youtube. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 876–892, 2022.

[3] Lamiaa Basyoni, Noora Fetais, Aiman Erbad, Amr Mohamed, and Mohsen Guizani. Traffic analysis attacks on tor: A survey. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 183–188, 2020.

[4] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. Tor: The second-generation onion router. In *USENIX security symposium*, volume 4, pages 303–320, 2004.

[5] Arjun Guha, Matthew Fredrikson, Benjamin Livshits, and Nikhil Swamy. Verified security for browser extensions. In *2011 IEEE Symposium on Security and Privacy*, pages 115–130, 2011.

[6] Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson. An analysis of the privacy and security risks of android vpn permission-enabled apps. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, page 349–364, New York, NY, USA, 2016. Association for Computing Machinery.

[7] Soroush Karami, Panagiotis Ilia, Konstantinos Solomos, and Jason Polakis. Carnus: Exploring the privacy threats of browser extension fingerprinting. In *In Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, 2020.

[8] Amit Klein and Benny Pinkas. DNS cache-based user tracking. In *NDSS*, 2019.

[9] David M. Martin, Richard M. Smith, Michael Brittain, Ivan Fetch, and Hailin Wu. The privacy practices of web browser extensions. *Commun. ACM*, 44(2):45–50, feb 2001.

[10] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. In *2012 IEEE Symposium on Security and Privacy*, pages 413–427, 2012.

[11] Paul V Mockapetris. Rfc1034: Domain names-concepts and facilities, 1987.

[12] Apurva Nalawade, Smita Bharne, and Vanita Mane. Forensic analysis and evidence collection for web browser activity. In *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pages 518–522, 2016.

[13] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1962–1976, New York, NY, USA, 2018. Association for Computing Machinery.

[14] Donny Jacob Ohana and Narasimha Shashidhar. Do private and portable web browsers leave incriminating evidence? a forensic analysis of residual artifacts from private and portable web browsing sessions. In *2013 IEEE Security and Privacy Workshops*, pages 135–142, 2013.

[15] Aamir Rasool and Zunera Jalil. A review of web browser forensic analysis tools and techniques. *Researchpedia Journal of Computing*, 1(1):15–21, 2020.

[16] Konstantinos Solomos, John Kristoff, Chris Kanich, and Jason Polakis. Tales of favicons and caches: Persistent tracking in modern browsers. In *Network and Distributed System Security Symposium*, 2021.

[17] Oleksii Starov and Nick Nikiforakis. Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1481–1490, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[18] Shashank Tomer, Aviral Apurva, Pranshu Ranakoti, Saurav Yadav, and Nihar Ranjan Roy. Data recovery in forensics. In *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, pages 188–192, 2017.

[19] Randika Upathilake, Yingkun Li, and Ashraf Matrawy. A classification of web browser fingerprinting techniques. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, 2015.

[20] Yuxi Wu, Panya Gupta, Miranda Wei, Yasemin Acar, Sascha Fahl, and Blase Ur. Your secrets are safe: How browsers' explanations impact misconceptions about private browsing mode. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 217–226, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

Optimisation of heating, ventilation, air-conditioning, and cooling (HVAC) with machine learning (ML) methods

Julius Mikala
julius.mikala@aalto.fi

Tutor: Matti Huotari

Abstract

Traditional HVAC control methods like rule-based control (RBC) have provided satisfactory indoor environments but lack optimisation in cost and responsiveness to external influences. Enhanced control methods, particularly those employing machine learning (ML) techniques, show significant potential for advancement in HVAC optimisation. This paper presents a overview of different approaches to HVAC optimisation with a focus on ML and Reinforcement Learning (RL) methods. HVAC optimisation is discussed by how various ML and simulation methods are employed to predict how the building responds to control. Next, the RL control models are discussed. The discussed literature suggests that ML controlled HVAC systems demonstrate improvements in temperature stability and energy expenditure.

KEYWORDS: HVAC control, artificial neural network, reinforcement learning

1 Introduction

Spending a substantial amount of time indoors has heightened the emphasis on the quality of indoor environments, with a direct impact on human health and productivity. Numerous studies [1] highlight the positive association between indoor environment enhancements and the well-being of occupants. Key to improving indoor conditions are Heating, Ventilation, Air-conditioning, and Cooling (HVAC) systems, which have been identified as major contributors to the overall indoor environmental factors affecting health and productivity. In the United States, HVAC systems constitute up to 50% of a building's energy budget [16]. Consequently, the pursuit for solutions that enhance HVAC energy efficiency, while simultaneously maintaining or improving indoor occupant comfort, becomes crucial for developing sustainable buildings.

Current control methods for HVAC include rule based control (RBC) and other basic automation systems, which are often created through expert knowledge [12]. As a result, the control afforded by the RBC method can provide satisfactory indoor environments, but is unoptimal when it comes to cost. Furthermore, RBC method is poor at responding to outside influences such as weather, and occupancy [12]. As a result, enhanced HVAC control methods exhibit significant potential for advancement. Moreover, the prevailing research trend underscores an inclination towards machine learning methods, as evidenced by several literature reviews [12, 5, 18]. The reason for utilising ML for building control stems from its ability to use data to learn the complex building environment, meaning that time and effort can be saved [18].

Other frequently used methods in HVAC optimisation include Model Predictive Control (MPC) and Fuzzy Logic [5]. These aforementioned control methods are not ML since the methods do not use data for learning. Instead non-ML methods may have the same drawback as RBC, requiring expert knowledge for controller design. As ML control methods arguably enable easier controller creation this paper will focus on them.

This will paper review will different approaches to optimising HVAC with ML methods. The goal is to give the reader an overview of the different methods used in HVAC optimisation. Therefore, the paper will cover the dynamic building models, what kind of buildings the control is employed in, and the ML control methods employed. As Artifical Neural Network (ANN) and Reinforcement Learning (RL) are currently one

of the most frequently employed methods for HVAC control optimisation this paper will focus on these.

2 Machine learning methods

ML is defined as a methods that use data to automate a solution to some difficult problem [15]. ML differs from other ways of solving such problems by involving data that is used to train a model capable of a solution. In contrast other methods often require expert knowledge to arrive at a solution. Machine learning problems can be divided into supervised, unsupervised, and reinforcement learning. ANN are a ML method, which have sparked interest in ML and are often used to solve complex problems [15]. Arguably, this also applies to HVAC optimisation which is why the rest of this chapter will focus on various ANN methods.

2.1 ANN methods

ANNs encompass a diverse array of methods, including Multilayer Perceptrons (MLPs), Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNN), and more, each tailored to address specific tasks and challenges in machine learning. Introducing Multilayer Perceptron (MLP) networks serves as an good starting point in exploring the capabilities of ANNs.

An ANN is designed to mimic biological neurons [11]. They function by summing together inputs and a bias, which is then passed through an activation function. The activation function is used to introduce nonlinearity to the network, and enables it to model more complex relationships. This signal is then sent to subsequent neurons through synapses which apply a weight to the signal. In a perceptron the activation function is such that the neuron output is 1 when the input sum is above a threshold and 0 otherwise. Perceptrons and multilayer perceptrons are an example of a feed-forward network where neurons are organised in layers and connections always feed forward to the next layer. ANN typically learn with a back-propagation algorithm, where the weights and biases of a network are adjusted layer by layer in order to bring the network output closer to a desired value. The learning is also usually done in several steps. This training can be done as supervised learning, where the desired learning goal is known. Unsupervised learning is the opposite

of this, but is less commonly used for heating optimisation as can be seen from a literature review [5]. Reinforcement learning is a learning setting where the desired output is not known, but the a reward can be calculated for it [11].

In recurrent neural networks the connections between neurons is not limited to a feed forward direction, but instead the neurons can connect backwards to create back-loops [11]. This enables the network to hold memory to exhibit dynamic temporal behaviour. As a result, RNNs are often used for time-series. However, RNNs suffer from vanishing or exploding errors with long input sequences [9]. Long short-term memory (LSTM) networks are one adaptation that solves this issue by keeping the long term memories constant. This enables LSTMs to forecast time series effectively even when there are long periods between important events [11].

Reinforcement learning techniques can be applied independently of ANNs, as demonstrated by Q-networks [11]. Nevertheless, ANNs are commonly employed in heating optimisation literature due to their capacity to handle continuous states and actions. RL is created on the concept that an agent performs an action in an environment when the environment is in a certain state [6]. During training the agent learns an optimal control policy that is a mapping between states and the probability of selecting an action. RL defines a state-value function that represents the future rewards when following the optimal policy. The value of an action is defined by action-value-function which indicates the value of an action in a state when following the policy. During RL model training there is a back and forth between exploiting already functional strategies, and exploring actions/states that have not yet been observed.

Convolutional neural networks are a variation on the way ANNs process connections between layers. CNNs solve the issue stemming from large amounts of inputs resulting in large amounts of parameters to tune in traditional feed forward networks [3]. CNNs consists of several layers, which include convolution, nonlinear, pooling, and fully-connected layers. The nonlinear layer is the layer in which an activation function is applied, whereas a fully-connected layer is an example of a simple ANN layer discussed previously. The convolutional layers use convolution mathematical operations instead of connections with weights as in traditional ANNs. Finally, pooling layers are a method of downsampling large inputs and are used after convolution layers.

3 Optimisation approaches

There are several different methods for optimising HVAC with ML. The most discussed methods include model predictive control (MPC), reinforcement learning (RL) as can be seen in literature reviews [5, 18, 2]. While several From the literature reviews, the algorithms are distinguished by how the dynamics of a building are modelled and the algorithms used for controlling. Other distinguishing environmental factors are the variables that are accounted for, the number of actions available for the controller, and the optimisation goal. As each building has different HVAC and building characteristics, the control problem requires identification of the system [5]. This also means that a universal model for control cannot be created [5].

3.1 Building model

The optimisation of a building requires some form of representation of a building, which is achieved with a dynamic model of the building [5]. Thus, HVAC optimisation control methods often use two different models, a dynamic for forecasting the behaviour of the building, and a control algorithm for the output. The dynamic model can be used either in the control loop, or to create the controller. When a dynamic controller is not used in the control loop, the control method is referred to as model-free [5]. Model-free methods do not necessarily need a dynamic building model for creating the model, but can instead be use historical data [12]. However, this limits the controller based on the historical data. As a result, training the control model on a real building is likely infeasible due to cost and time constraints. Most model-free implementations use RL or artificial neural networks, but fuzzy logic has also been used for the control task [12]. The reason for the lack of fuzzy logic implementations is the difficulty in creating them. Dynamic models are often used in the training of model-free controllers because because the often used methods, RL and ANN, require large amounts of training data [12]. In contrast, control methods used with dynamic building models in the control loop are more varied, and include ANN, SVM, as well as other supervised ML methods [5].

The building models can be divided into white-box, black-box, and grey box-models [5]. The white-box models are based on modelling the physics of the buildings. While this method ensures that the models are explain-

able, the models are also difficult and time-consuming to create. Examples of black-box models include the RL and ANN methods [5]. Such models require a lot of data from previous operation of the building. Finally, grey-box models combine both both of the previous types.

An example of a black-box building, can be seen in a study [14] where a convolutional neural network-long short term memory (CNN-LSTM) model is used to approximate the the real world behaviour of the building. CNN-LSTM models are often used for time-series prediction in applications that range from stock-prices to energy consumption. These models are used in time-series problems because the number of inputs that time-series problems require are often very large, which results in lengthy training periods [14]. Therefore, reducing the number of features is beneficial which can be achieved with a CNN model that reduces the number of features by detecting patterns in the data. These features are then used by the LSTM model to create a prediction of the next time-step. When comparing an LSTM and a CNN-LSTM model, the latter learns faster and produces more accurate results [14]. The CNN-LSTM energy consumption prediction model used compressor suction temperature, evaporator side water outlet temperature, condenser side water inlet temperature, and condenser side water inlet temperature as inputs. The study [14] is highly focused on the describing how the training of the DRL methods are improved, and omits to mention what the HVAC controller action space is. More information is provided on a building model in [17] where two models are used to predict the behaviour of the house. These are a model to predict the thermal environment of the room and a model to predict the energy consumption of the environment. The study [17] compares an LSTM model and a XGBoost model, and finds that the XGBoost model performs better based on R^2 score. The inputs for the models includes several indoor temperatures, outdoor temperatures, and the state of the HVAC system. These models are later used to train and validate RL models.

A of a white-box building model is used in [13] uses EnergyPlus software to model how the building indoor temperature responds to control. The modelling involves creating a 3D model of the building, where the insulating properties of facade and airflow parameters are included. Simulation is then performed with contrast transfer functions and finite difference methods. "*This algorithm discretises building enclosures such as walls, floors, and ceilings into various nodes and numerically solves the*

heat transfer equations using a Finite Difference Method" [13]. Other publications indicate that EnergyPlus is a frequently used model to forecast building temperature and energy usage. While these white-box models can not be considered ML, they are often used together with ML methods to solve the building control problem.

3.2 Building control with Reinforcement Learning

When ML is used for HVAC control one of the most frequently used methods is currently RL. As a result, this section will focus on how the control problem is solved with RL. Some often used reinforcement learning algorithms include Q-learning, policy-gradient, actor-critic, and value based [5, 18]. Several methods can be used to realise RL, when neural networks are employed for this purpose the algorithm is referred to as deep reinforcement learning (DRL) [6]. In most papers [6, 8, 17] employing RL for HVAC control the building model is only employed in the training phase. The building model is used to teach the model an optimal policy based on the building state As a result, RL can be considered a model free method.

When considering HVAC optimisation, the simplest approach is optimising the heating in HVAC. This can be seen in three studies [6, 8, 17]. While all the aforementioned studies approach the control problem from the perspective of improving temperature stability, the studies differ due to differences in hardware and choice of methods. For example, in [6] the heating system consists of water radiators, meaning that the control signal or action space is defined as the a discrete list of values that represent the radiator water temperature. In contrast, when the control hardware is a electric radiator the action can be defined as a boolean on/off [8]. Finally, in [17] the state space is again defined as a discrete list which represents the supply air setpoint temperature of a HVAC system. All three studies use variations on DRL. As a result, the resign of the controller reward function is important as it is used to indicate how well the control functions during training. In each case the reward functions are similar as two reward functions. The first reward functions measures temperature comfort, which is defined as how stable the indoor temperature is. The second function, measures energy consumption normalised according to outside temperature. Outside temperature must be accounted for in the energy consumption reward function because heating becomes increasingly more demanding as the outside temperature drops. The state-spaces for the DRL algorithms use indoor environment sensors, mainly temperature,

and various weather features. Weather features include outside temperature and sunshine. In cases where an office building is heated [6, 8] the building schedule is also include since the temperature is expected to be lowered when the building is not in use. The studies [6, 8, 17] reported energy expenditure reductions of around 10%.

HVAC devices are responsible for controlling other factors apart from the simple temperature use case, such as CO2 through ventilation [7, 19]. When considering several factors the control problem becomes more difficult since the action space becomes larger and different actions can affect both temperature and indoor air-quality. Furthermore, HVAC devices are often separated into zones, which means that control must be either implemented for each zone separately or as multi-zone control. The former introduces problems due to possible decreases efficiency, whereas the latter sees an increase in problem complexity. In [7] the problem with increased action space is handled by training several DRL models in parallel. Each model control a different part of the HVAC system. This approach saw an overall decrease in energy usage of around 11%. The problem of multi-zone HVAC control is solved in a similar manner in another study [19]. Here models for each zone are trained simultaneously so that the control models are trained to work together. Further methods are then employed to create a multi-zone controller, but due to their complexity these are outside the scope of this paper. Testing of this approach shows that indoor CO2 concentration, temperature stability, and energy usage are all improved significantly.

4 Discussion

Given that ML control of buildings has been shown to shown to provide a noticeable improvement in both simulated and real settings we might question why this control method is not already widely adopted. This lack of adoption in the building industry is evidenced by publications discussing ML control as possible improvements, but not as something that has been realised on a large scale. Such an example can be seen in a study [10], where the possibility of implementing ML control is buildings is consider fro Finland. In part this lack of adoption may be explained by the novelty of the subject. While there are publications as old as 2002, the subject has seen a significant increase in publications starting from 2018 [2]. As a result, the ML solutions may not have had time to be im-

plemented in the building industry. Another drawback with ML control of HVAC is the data requirement. Large office buildings or campuses may use thousands of sensors. For older buildings the investment in sensors might thus outweigh the improvement gained from ML control. Moreover, when employing a ML based controller for as the building dynamic model, this data must be stored in order for it to be usable. The same also applies to new or renovated buildings where there is a lack of data required for model training. In the case that the appropriate sensor are installed and the data stored, there might still arise problems from the data format. Currently, most buildings have data structures that vary between buildings [4], which means that data-points must be manually selected. As result, ML solutions may not be scalable enough for widespread adoption. Therefore, future research may need to tackle this problem of scalability before ML HVAC optimisation is widely adopted.

5 Conclusion

This paper sought to give the reader an overview of methods employed for HVAC optimisation. This was achieved by defining mL methods, such as ANNs, and RL which are often used in HVAC control. These methods leverage data to learn complex building environments, enabling more adaptive and responsive control strategies. Next, the two types of models employed in HVAC control were discussed. First, the models predicting how a building responds to control, meaning dynamic building models. ML methods employed for these include variations on LSTM networks, and XGBoost. However, building simulation software, which are not ML was also employed for this purpose. The second sort of model used for HVAC control is the model producing the control signal. These control models included ANN, and DRL methods.

When deployed the ML controlled HVAC systems showed improvements in both temperature stability and energy expenditure. However, the adoption of ML-based HVAC optimisation faces challenges. Data requirements, scalability issues, and variability in data structures may pose significant hurdles.

References

[1] Yousef Al Horr, Mohammed Arif, Amit Kaushik, Ahmed Mazroei, Martha Katafygiotou, and Esam Elsarrag. Occupant productivity and office indoor environment quality: A review of the literature. *Building and Environment*, 105:369–389, August 2016.

[2] Maher Ala'raj, Mohammed Radi, Maysam F. Abbod, Munir Majdalawieh, and Marianela Parodi. Data-driven based hvac optimisation approaches: A systematic literature review. *Journal of Building Engineering*, 46:103678, April 2022.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[4] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 41–50, 2016.

[5] Yasaman Balali, Adrian Chong, Andrew Busch, and Steven O'Keefe. Energy modelling and control of building heating and cooling systems with data-driven and hybrid models—a review. *Renewable and Sustainable Energy Reviews*, 183:113496, September 2023.

[6] Silvio Brandi, Marco Savino Piscitelli, Marco Martellacci, and Alfonso Capozzoli. Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings. *Energy and Buildings*, 224:110225, October 2020.

[7] Qiming Fu, Xiyao Chen, Shuai Ma, Nengwei Fang, Bin Xing, and Jianping Chen. Optimal control method of hvac based on multi-agent deep reinforcement learning. *Energy and Buildings*, 270:112284, September 2022.

[8] Anchal Gupta, Youakim Badr, Ashkan Negahban, and Robin G. Qiu. Energy-efficient heating control for smart buildings with deep reinforcement learning. *Journal of Building Engineering*, 34:101739, February 2021.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Lotta Kannari, Julia Kantorovitch, Kalevi Piira, and Jouko Piippo. Energy cost driven heating control with reinforcement learning. *Buildings*, 13(2):427, 2023.

[11] Andrej Krenker, Janez Bešter, and Andrej Kos. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pages 1–18, 2011.

[12] Panagiotis Michailidis, Iakovos Michailidis, Dimitrios Vamvakas, and Elias Kosmatopoulos. Model-free hvac control in buildings: A review. *Energies*, 16(20):7124, October 2023.

[13] SeyedehNiloufar Mousavi, Mohammad Gheibi, Stanisław Wacławek, Neale R. Smith, Mostafa Hajiaghaei-Keshteli, and Kourosh Behzadian. Low-energy residential building optimisation for energy and comfort enhancement in semi-arid climate conditions. *Energy Conversion and Management*, 291:117264, September 2023.

[14] Yi Peng, Haojun Shen, Xiaochang Tang, Sizhe Zhang, Jinxiao Zhao, Yuru Liu, and Yuming Nie. Energy consumption optimization for heating, ventilation and air conditioning systems based on deep reinforcement learning. *IEEE Access*, 11, 2023.

[15] Gopinath Rebala, Ajay Ravi, Sanjay Churiwala, Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. Machine learning definition and basics. *An introduction to machine learning*, pages 1–17, 2019.

[16] Vahid Vakiloroaya, Bijan Samali, Ahmad Fakhar, and Kambiz Pishghadam. A review of different strategies for hvac energy saving. *Energy conversion and management*, 77:738–754, 2014.

[17] Man Wang and Borong Lin. Mf^2: Model-free reinforcement learning for modeling-free building hvac control with data-driven environment construction in a residential building. *Building and Environment*, 244:110816, October 2023.

[18] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, July 2020.

[19] Liang Yu, Yi Sun, Zhanbo Xu, Chao Shen, Dong Yue, Tao Jiang, and Xiaohong Guan. Multi-agent deep reinforcement learning for hvac control in commercial buildings. *IEEE Transactions on Smart Grid*, 12(1):407–419, January 2021.

3D Gaussian Splatting

Junyuan Fang
junyuan.fang@aalto.fi

Tutor: Shuzhe Wang

Abstract

KEYWORDS: Novel View Synthesis, 3D-Gaussian Splatting, 3D Scene Representation, Volume Rendering, Data Visualization, Spatial Analysis

1 Introduction

Novel view synthesis (NVS) is the task of generating new images of a scene given single or multiple inputs of the same scene [7]. Novel view image synthesis typically involves two main steps: representing the scene and rendering the image. In recent years, NVS technology like Neural Radience Fields (NeRF) emerges as a prominent area of interest in computer vision and computer graphics research. This technology applies from creating detailed 3D models for architectural projects to its use in the film and gaming industries for producing photorealistic visuals and interactive experiences, as well as generating NVS in VR and AR, etc.

NeRF performs state-of-the-art complex scene creation, but its intensive computational demands, particularly during the ray-matching and sampling processes, lead to considerable overhead. Consequently, both the training and rendering phases in NeRFs are markedly slow. Thus, addressing the challenge of improving computational speed while achieving

high-fidelity 3D representation becomes imperative for practical applications in real-time rendering.

Recently, a groundbreaking non-neural network-based method, 3D Gaussian Splatting (3D GS) [8], is introduced to the computer vision and computer graphics community. This method leverages position, covariance, color, and opacity parameters to construct a comprehensive volumetric representation by using 3D Gaussians, significantly enhancing rendering speed and quality. It surpasses the performance of the current state-of-the-art in high-quality novel-view synthesis, Mip-NeRF360 [1], in a fraction of the training time [8]. In less than four years since the publication of the original NeRF paper, a substantial amount of research and numerous advancements emerges on the NeRF framework. This body of work includes various open word scene understanding tasks, such as the recent 3D visual and language model, LERF [9]. With its superior rendering capabilities, 3D GS sets to become a leading alternative, potentially replacing previous NeRF-based tasks.

This paper follows the trend and focuses on the recently released 3D GS method. This paper is organized as follows. Section 2 introduces the background related to 3D GS, like novel view synthesis, traditional 3D reconstruction and rendering and NeRFs. Section 3 describes how to represent the scene as a 3D GS with 3D Gaussians, spherical harmonic function (SH), and covariance matrices in world space and view space. Section 4 presents the optimization process of 3D GS. Section 5 discusses results and evaluation. Finally, the paper concludes with a summary of findings and contributions.

2 Background

In this section, we briefly review methods of NVS that rely on traditional reconstruction, as well as the NeRF approach that can achieve higher quality through implicit representation of 3D scenes.

2.1 Traditional 3D Scene Reconstruction and Rendering

Reconstructing a 3D scene from a collection of images and then rendering the 3D scene into a novel view camera constitutes the traditional NVS approach. Key early techniques, such as Structure-from-Motion (SfM) [18, 16], and multi-view stereo (MVS) [5, 17], are the groundworks for

converting 2D images into 3D information via stereo vision. SfM initially establishes a sparse point cloud during camera calibration, primarily for basic 3D space visualization. In subsequent advances, MVS algorithms produce a full 3D reconstruction. These foundational methods highlight the importance of explicit scene representation, as the 3D model enables the re-projection of new pixel-based images from a given target pose [15]. However, these traditional geometric methods [4, 16, 6] often struggle with occlusions, high computational costs, incomplete surface reconstruction, and also limitations in understanding light source changes and reflections.

2.2 Neral Radiance Fields

NVS through Neral Radiance Fields (NeRFs) [11] surpasses traditional algorithms in terms of performance issues mentioned above. As Figure 1 illustrates, NeRFs capture the propagation of light across the entire space, offering an implicit, continuous volumetric representation [19] of the scene. This approach effectively overcomes the challenges often encountered with traditional methods, including handling texture-less surfaces [14], occlusions [11], and incomplete surface reconstructions [11], as well as managing complex lighting and reflections [23]. Moreover, NeRFs ensure the continuity and completeness of the scene [11].

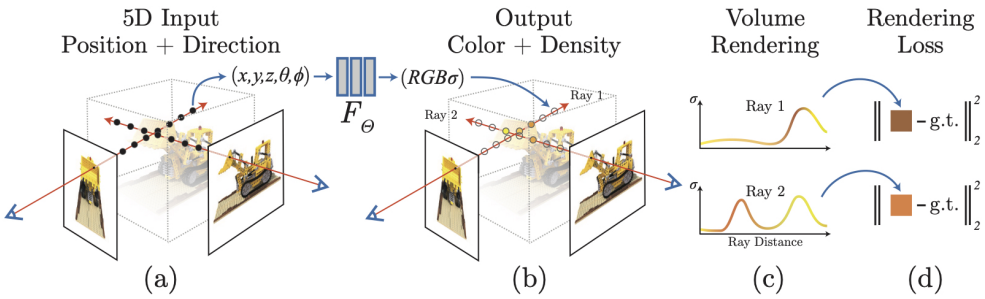


Figure 1. The description of NeRF's volume rendering and training process. Image sourced from [11]. (a) sends camera rays through the scene and generates a set of sampling points for each pixel in the image being synthesized. (b) applied MLP to query learned color and volume density for corresponding points. (c) uses volume rendering to turn colors and volume densities into pixel colors along camera rays. (d) compares colors to ground truth images to calculate the loss.

NeRFs adopt a different approach to scene representation, moving away from traditional methods that rely on points, meshes, or voxels for storing explicit radiance information. Instead, NeRFs utilize an implicit radiance field, denoted as $F_\theta : (x, y, z, \theta, \phi) \rightarrow (R, G, B, \sigma)$ for scene representation. Function $F_\theta(\cdot)$ processes spatial coordinates (x, y, z) along with

viewing direction's azimuthal and polar angles (θ, ϕ), producing the color information (R, G, B) and volume density σ. A deep learning model, often a Multilayer Perceptron (MLP), parameterizes this function, marking a significant advancement in how scenes are represented and visualized.

In NeRF's framework, each ray emanating from the target pose correlates directly with a pixel in the synthesized image. Through the use of a ray-marching technique, NeRF iteratively samples points along each ray to integrate color and density contributions, thereby determining the pixel's final color. This process effectively captures how light interacts with the scene, incorporating both absorption and scattering effects [11]. Enhancements such as importance sampling and positional encoding have significantly boosted NeRF's ability to reconstruct 3D scenes with remarkable detail and fidelity [11]. Despite these advances, the computational intensity of volumetric ray-marching poses challenges for real-time applications. Alternatives like 3D GS have emerged as more computationally efficient options for achieving real-time NVS.

3 3D Gaussian Splatting

3D GS [8] innovatively represents light distribution within a scene by shifting from implicit to explicit radiance fields. Unlike methods that rely on NeRF-like implicit radiance fields, which necessitate sampling dense points along each ray, including those in vacant spaces. 3D GS streamlines scene representation.

3D GS starts from a sparse set of (SfM) points without normal [8] to optimise the rendering process by circumventing unnecessary computations in areas devoid of content. This technique harnesses both the efficiency and adaptability of explicit radiance field representation by employing three-dimensional Gaussians, which are differentiable. 3D GS uses back-propagation-driven optimization with the density control of explicit 3D Gaussians, aiming for a balance that enhances visual quality, speeds up the learning phase, and supports real-time rendering. The mathematical formulation of the 3D Gaussian representation's idea is outlined as follows [2]:

$$F_{3D-Gaussian}(x, y, z, \theta, \phi) = \sum_i G(x, y, z, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \cdot c_i(\theta, \phi), \tag{1}$$

where G is the Gaussian function with mean $\boldsymbol{\mu}_i$ and anisotropic covariance $\boldsymbol{\Sigma}_i$, c_i represents the view-dependent color, and i indicates over-

lapped Gaussians' index. This section explains 3D Gaussian representation in 3D GS, and how an image is rendered based on this representation along with the covariance matrices in both world space and view space.

3.1 3D Gaussian

Reconstructed scene's surface are constructed by millions of Gaussian "ellipsoids" [8]. Each ellipsoid's shape is described by an unnormalized 3D Gaussian function, which consists of a mean value $\boldsymbol{\mu} \in \mathbb{R}^3$, an anisotropic covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$. Given a three-dimentional location $\boldsymbol{x} \in \mathbb{R}^3$, the probability density function of the 3D Gaussian function is defined as follows:

$$G(\boldsymbol{x}) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}. \tag{2}$$

Figure 2 provides a comprehensive understanding of a Gaussian function for the uncoloured ellipsoid's shape without applying a threshold[1] to the size of the 3D Gaussian.

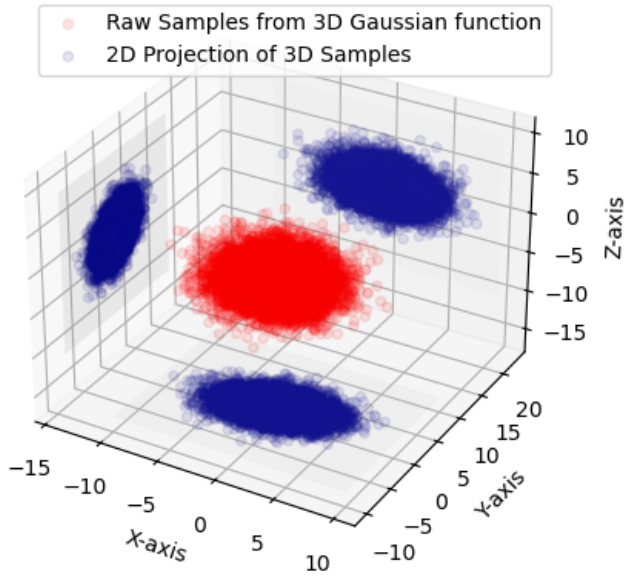


Figure 2. 3D Gaussian distribution "ellipsoid"'s visualization. Red-coloured points are points randomly sampled from 3D Gaussian, and dark blue-coloured points are points projections of points on X-Y, Y-Z, and X-Z planes.

In Equation 1, the radiance field is represented by Gaussian functions and the corresponding color [2]. Each Gaussian function encodes the color

[1] In gsplat [20], the Gaussians' size is set up to 3 standard deviations, thereby ensuring that the 3D Gaussian representations are clipped at the 99.8 % percentile.

$c_i \in \mathbb{R}^3$ using SH [3, 12]. SH allows the color to vary with the viewing direction. The expansion of Equation 1 incorporates α-blending on a point-based rendering, which enables the layering of N points according to their depth [8]. This process arranges points from the nearest to the furthest from the target pose, allows for the computation of the pixel color $C \in \mathbb{R}^{3\times3}$ by using the 3D Gaussian as a "point" :

$$C_{\text{pixel color}} = \sum_{i \in N} c_i \cdot T_i, \text{ where } T_i = \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j). \tag{3}$$

Here, T_i reflects each 3D Gaussian's opacity $\alpha_i \in [0,1]$, adjusted for the combined transparency of preceding Gaussians, influencing the depth-based color blending. Opacity α_i is given by a 2D Gaussian function multiplied by a learned opacity o_i obtained for each Gaussian:

$$\alpha_i = o_i \cdot e^{-\frac{1}{2}(\boldsymbol{x}'-\boldsymbol{\mu}_i')^\top \boldsymbol{\Sigma}_i'^{-1}(\boldsymbol{x}'-\boldsymbol{\mu}_i')}, \tag{4}$$

where $\boldsymbol{\Sigma}'$ is view space covariance of the 3D Gaussin, and $\boldsymbol{x}'$ and $\boldsymbol{\mu}_i'$ are 3D Gaussian position and mean position coordinates in the projected space.

3.2 World Space and View Space Covariance Matrices

Pixel's color C calculation uses a step to project 3D Gaussian to the 2D image plane. The view space covariance matrix $\boldsymbol{\Sigma}' \in \mathbb{R}^{2\times2}$ describes the covariance of the projected 2D Gaussian. Projection from 3D to 2D involves transforming the mean position $\boldsymbol{\mu}$ of the 3D Gaussian using point projection and the transformation of word space covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ according to the following equations:

$$\boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T\boldsymbol{J}^T, \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{3\times3}$ is the viewing transformation and $\boldsymbol{J} \in \mathbb{R}^{2\times3}$ is the Jacobian of the affine approximation of the projective transformation [8, 24]. The covariance matrix defines $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ an ellipsoid, its equivalent decomposed representation is used as follows [8]:

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T, \tag{6}$$

where $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{S} \in \mathbb{R}^{3\times3}$ are the rotation and scaling matrices, respectively. Finally, by substituting the formula from equation 6 into equation 5, then equation 5 into equation 4, and so on, the rendered pixel for the target pose is computed. Replicating these steps for all pixels allows us to render the entire image in the target pose.

4 3D Gaussian optimization step

The core of 3D GS is the optimization step [8], it offers a method to represent the scene accurately by optimizing 3D Gaussian's mean position $\boldsymbol{\mu}$, opacity α, covariance matrix $\boldsymbol{\Sigma}$, and SH coefficients. Controlling the density of 3D Gaussian in different areas also constitutes an optimization step after optimizing parameters for a single 3D Gaussian. This section introduces parameter optimization and density control for 3D GS.

4.1 Parameter Optimization

In order to enhance the quality of NVS images, 3D GS employs a weighted loss function that assesses both the absolute error and the similarity difference between synthesized and ground-truth images. It utilizes the well-known gradient descent method for optimizing the parameters. The weighted loss function is outlined as follows:

$$L = (1 - \lambda)L_1 + \lambda L_{D-SSIM}, \tag{7}$$

where term L_1 compares the absolute value difference, term L_{D-SSIM} [13, 21] compares the structural similarity of the images, and weighting factor λ balances these assessments for NVS images.

In the parameter optimization phase, particularly during gradient descent steps[2], directly optimizing the anisotropic covariance matrix $\boldsymbol{\Sigma}$ can lead to $\boldsymbol{\Sigma}$ becoming non-positive semi-definite, which complicates the loss function's search for a local minimum. Constraining and decomposing $\boldsymbol{\Sigma}$ into a rotation matrix $\boldsymbol{R}$ and a scaling matrix $\boldsymbol{S}$ separately as mentioned in equation 6 helps avoid the problem of non-positive semi-definiteness.

To further enhance the efficiency of parameter optimization and the rendering process, 3D GS designs a tile-based rasterizer [10, 8], which splits the screen into 16x16 tiles and then excludes 3D Gaussians that do not visually affect the current tile. This strategy reduces the number of Gaussians that need to be processed and accelerates the parameter optimization steps, enabling faster and more effective NVS image generation.

4.2 3D Gaussian Density Control

3D GS starts with sparse points obtained from SfM as starting points for initializing 3D Gaussians. The process transitions 3D Gaussians from

[2]For an in-depth discussion on parameter optimization, refer to the mathematical supplement [22].

sparse to dense ensembles by adaptively cloning, splitting, and pruning to adjust both the count and density of Gaussians per unit volume [8].

The process of moving from sparse to dense 3D Gaussian representation is based on filling information in empty areas. The area lacking in geometric detail is known as the "under-reconstructed" area, and the area has huge 3D Gaussians across vast portions of the scene known as the "over-reconstructed" area. Both "under-reconstructed" and "over-reconstructed" areas can be identified by the large view-space positional gradient. To overcome this, large Gaussians in the "over-reconstructed" area are split into smaller Gaussians. For small Gaussians in the "under-reconstructed" area, the new geometry can be covered by cloning, i.e., creating a Gaussian copy of the same size and moving it toward the positional gradient. In addition to obtaining precise scene representation by not only increasing 3D Gaussian density, 3D GS applied a thresholded hyperparameter ϵ_α to the opacity α of 3D Gaussians, thresholding the minimal visual contribution of 3D Gaussians, as a pruning step.

During the optimisation process, the system encounters floaters close to the input camera, which is common in volumetric representations [8]. In this case, the presence of these floaters leads to an unjustified increase in the 3D Gaussian density in an attempt to represent these irrelevant objects close to the camera. Therefore, the 3D GS sets the opacity α on the camera edges close to zero. This is because, as mentioned above, the 3D Gaussians will increase as needed, at the same time threshold ϵ_α can remove 3D Gaussians if they are not needed. This allows us to control the density of the 3D Gaussians.

5 Results

Table 1 uses structural similarity index measure (SSIM), peak signal-to-noise ratio (PNSR) and learned Perceptual Image Patch Similarity (LPIPS) error metrics to compare the NVS performance of Mip-NeRF360 and 3D GS from different aspects. Quote to the original 3D GS paper results, from Table 1, experiments of 3D GS use a single A6000 GPU, and Mip-NeRF360 experiments use a 4-GPU A100 node for 12 hours, which is approximately equivalent to 48 hours on a single A100 GPU. As Table 1 shows, sometimes 3D GS is even slightly better than the SOTA Mip-NeRF360 method with less than 1.5 percent of Mip-NeRF's training time.

Figure 4 shows nerfstudio [20] implemented reproduction of novel view

Table 1. Performance comparison of Mip-NeRF360, 3D GS with 3K iterations and 7K iterations on Mip-NeRF360 dataset. Results marked with dagger † have been directly adopted from the original Mip-NeRF 360 paper [8].

Dataset	Mip-NeRF360					
Method	SSIM↑	PSNR↑	LPIPS↓	Train	FPS	Mem
Mip-NeRF360	0.792†	27.69†	0.237†	48h	0.06	8.6MB
3D GS-7K	0.770	25.60	0.279	6m25s	160	523MB
3D GS-30K	0.815	27.21	0.214	41m33s	134	734MB

image syntheses with 3D GS. Figure 3 shows training images, which are a series of collected 360-degree view images of a uniform altitude taken around the aircraft. Based on these outcomes, we discover that 3D GS also like NeRF, both rely on training sources, and both give artefacts in under-observed parts of the scene.

Figure 3. Overlay of NVS image with different pose's training images

.

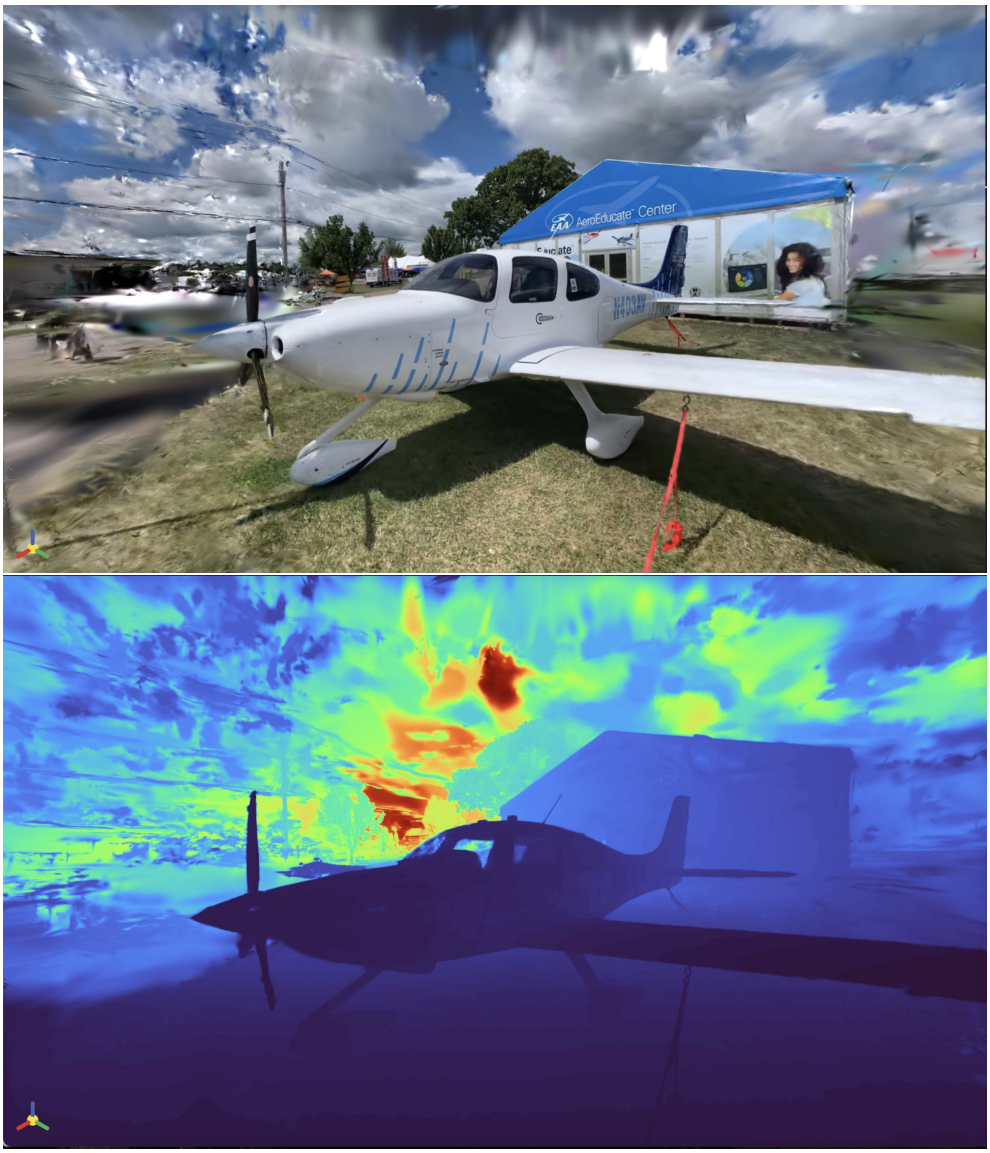


Figure 4. NVS of an RGB image and a depth map trained with 30K iterations. The model is trained on an Nvidia 2080Ti GPU in 30 minutes using the gsplat [20] method.

6 Conclusion

This paper reviews 3D GS and assesses the strengths as well as the weaknesses of this novel view synthesis method. 3D GS uses explicit 3D Gaussian representations and implements fast and accurate NVS tasks with significantly less training time than NeRFs. However, this method produces artefacts in under-observed parts of the scene, a problem that is not unique to 3D GS, but is also present in other methods such as Mip-NeRF360 [1].

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[2] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024.

[3] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[4] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.

[5] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[6] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[7] Yuxin Hou, Arno Solin, and Juho Kannala. Novel view synthesis via depth-guided skip connections. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3119–3128, 2021.

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[9] Justin* Kerr, Chung Min* Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

[10] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021.

[11] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[12] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[13] Jim Nilsson and Tomas Akenine-Möller. Understanding ssim. *arXiv preprint arXiv:2006.13846*, 2020.

[14] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12892–12901, 2022.

[15] Daniel Scharstein. *View synthesis using stereo vision*. Springer, 2003.

[16] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[17] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016.

[18] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006.

[19] Richard Szeliski. *Computer Vision: Algorithms and Applications*, chapter 13.5 Volumetric Representations, pages 830–833. Springer Nature, 2022.

[20] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[21] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[22] Vickie Ye and Angjoo Kanazawa. Mathematical supplement for the gsplat library. *arXiv preprint arXiv:2312.02121*, 2023.

[23] Junyi Zeng, Chong Bao, Rui Chen, Zilong Dong, Guofeng Zhang, Hujun Bao, and Zhaopeng Cui. Mirror-nerf: Learning neural radiance fields for mirrors with whitted-style ray tracing. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 4606–4615, 2023.

[24] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.

Scalability challenges of microservices

Kalle Korhonen
kalle.a.korhonen@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

Microservice architecture is an emerging software architecture, where software is divided into small, individually developed, and executable units. This paper investigates the scalability challenges of microservices compared to monoliths. While monoliths can be scaled by launching multiple instances of the same application, microservices offer the advantage of scaling individual parts of the application independently. However, the scalability advantages of microservices bring challenges in automatic scaling, load balancing, service discovery, and data consistency.

Automatic scaling is used for dynamically adjusting the number of instances of a microservice based on current workload demands. Load balancing is a technology used for distributing traffic across scaled microservice replicas. Service discovery is a solution to dynamically changing IP addresses, facilitating communication between microservices. The Kubernetes microservices platform includes solutions to these challenges.

Data consistency is a significant challenge in microservices due to the database-per-service approach used in microservice systems and the inability to guarantee ACID properties across distributed transactions. This paper looks at how to deal with this problem, suggesting the use of sagas to handle distributed transactions better.

KEYWORDS: microservices, software architectures, distributed systems

1 Introduction

Software has traditionally been developed using the monolithic approach [6, 11, 18], where all parts of the software are developed in a single codebase, and the software is compiled into a single executable which includes all parts of the software. The monolithic approach might become limiting with complex applications that receive a high amount of traffic because monolithic applications can be more difficult to maintain and scale.

Microservice architecture [6, 11, 18] is one solution to the challenges of monolithic architecture. In the microservices architecture, the software is divided into smaller, independently developed, and executable units called microservices. Microservices architecture has gained popularity since the beginning of the 2010s.

One of the biggest advantages of microservices architecture is the potential for better scalability. This is possible by the ability to scale individual parts of the software separately, addressing areas with the highest load. However, there are challenges in managing scalability, including issues with automation and ensuring consistency in the application's data and state.

This paper discusses the scalability of microservice systems by looking at current solutions and methods. Using related literature, it provides an analysis that highlights practical approaches to address scalability issues.

This paper is organized as follows. Section 2 explains the general concepts and background of microservices. Section 3 discusses scaling microservices and presents various concepts used in scaling microservices. Section 4 discusses the challenges encountered in scaling microservices introduced in Section 3. Section 5 concludes the findings of this paper.

2 Microservices

This section provides general concepts and background knowledge required for understanding microservice systems. Section 2.1 provides a definition of monoliths. Section 2.2 explains the general concept of microservices.

2.1 Monoliths

Monolithic architecture [6, 11, 18] has been a traditional way of building software for a long time. In monolithic architecture, the whole application is built from a single codebase and is commonly compiled into a single executable binary [4]. This means that the individual parts of the application cannot be executed independently, the whole application executes in a single process. An example of a monolithic architecture can be seen in Figure 1. It consists of a single application that contains everything from the user interface to business logic. Monolithic applications typically have a single database for storing data.

Monolith architecture can be a good design choice for many applications. When the codebase is small, the monolith can be a simple and effective choice [12]. Dragoni et al. [6] highlight several challenges associated with monoliths, including issues related to complexity, scalability, deployment and technology lock-in.

Large monolithic applications can become very complex and maintaining them can get difficult. Developing new features and fixing existing issues might need large changes in many different parts of the codebase [4]. There can be dependency conflicts if a certain dependency requires a specific version of some library, but another dependency requires another version of the same library. Compiling large monolithic applications may be time-consuming, it can cause a slowdown when the program is tested during the development [18].

It is possible to scale monolithic applications by creating replicas; increasing the number of running instances of the application. This increases the processing capacity of the application and allows processing of more requests at once. The traffic will be distributed uniformly among the replicas. The issue with scaling monolithic applications is that the parts of the application that are affected by the high load might be just a small part of the application. Scaling the whole application instead of specific parts of it can be wasteful and end up consuming unnecessary resources [6].

Technology lock-in means that the developer team must keep using the same programming language and technologies for the whole application [6]. Some programming languages are better suited for some tasks than others. With monolithic applications, the whole application is written in the same programming language. The developers working on the

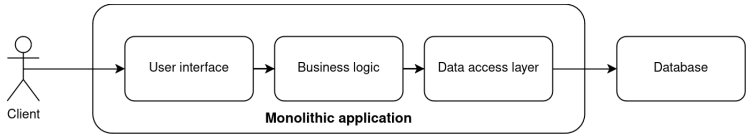


Figure 1. An example of a monolithic architecture

application must all agree to use the same language [6].

2.2 Microservices

Microservice architecture [6, 11, 12] is an emerging way to build software. In the microservices architecture, the application logic is split into small separate modules called microservices. Microservices are independent of each other and are developed individually. They follow the single responsibility principle (SRP) [12] which means that each microservice aims to serve only one functionality. Microservices started gaining popularity in the early 2010s as companies like Netflix [4] adopted this architecture.

Compared to monolithic applications, in microservice applications all application code doesn't run in the same process or thread, it might not even run on the same machine. In monolithic applications, the communication between the modules is done internally using function calls inside the same process [4]. Microservices rely on the network for communication. Communication between microservices should happen through well-defined interfaces that hide the internal workings of the microservices. Microservices shouldn't be able to access other services' internal components or function calls [18]. A typical communication method used in microservices is REST APIs over HTTP [4]. Other popular options are binary-based RPC protocols such as gRPC [16] which can offer better performance than HTTP. Message queue protocols such as MQTT are also popular choices for microservice communications [15].

Monolith systems usually use a single central database, but with microservices, the common practice is that each microservice has its own database [4]. This is called the database-per-service approach [13]. The approach provides flexibility and independence for microservice developers, because different database systems can be chosen for each microservice based on their needs [8]. For example, an online store with orders and product microservices could benefit from using a relational database for the orders microservice and a document database like MongoDB for the product information microservice. The separation of databases also

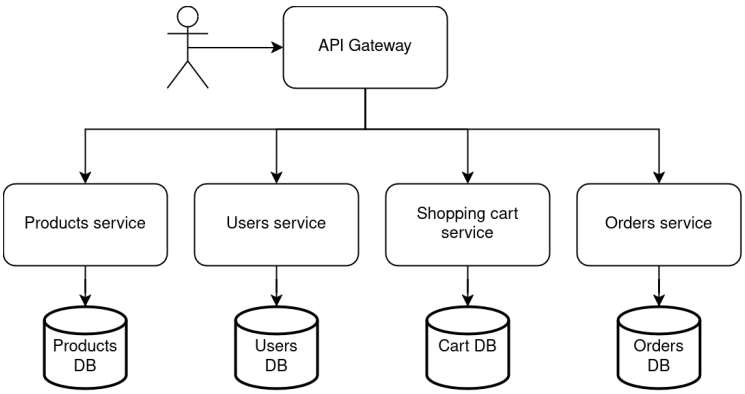


Figure 2. An example of a microservice architecture

makes it possible to alter database schemas without affecting other parts of the application [12].

Microservice architecture relies heavily on container technology. Each microservice runs in its own container which contains all its dependencies. Containerization tools such as Docker [9] and Kubernetes [2] have made the implementation and deployment of microservices more accessible [11]. Modern cloud platforms provide extensive facilities for running and managing containerized applications, for example, managed Kubernetes clusters and Docker container hosting services.

There are many benefits in microservice architecture when compared to traditional monolithic architecture. Dragoni et al. [6], Jamshidi et al. [11] and Laigner et al. [13] highlight several advantages of microservice architecture, including improved scalability, flexibility and independence.

One of the main benefits of microservices is the improved scalability. Because the system consists of independent microservices, each microservice can be scaled individually [11, 7]. This means that only the parts of the application that receive high load need to be scaled. This can improve resource utilization when the whole application doesn't have to be scaled. Different parts of the application can also have different performance requirements. Some parts might require a large amount of RAM while some parts might need a high single-threaded CPU performance [18].

Microservices are developed independently, making it possible to implement them in different programming languages. This flexibility enables development teams to adopt the most suitable technological choices for each microservice, and the independence to use their preferred technologies [11]. Independence also improves fault tolerance, if one microservice fails, other parts of the application can still stay functional.

Figure 2 shows a typical microservice architecture using a fictitious

online store as an example. The online store's functionality is split into individual microservices, products, users, shopping carts and orders that have their own services. In this example, the microservices are connected to an API gateway [10]. API gateway is a service that receives incoming requests and routes them to correct microservices.

3 Scalability in microservices

This section discusses the scalability of microservice systems and the challenges related to scalability. Each challenge is described, and it is explained how they have been solved in existing systems.

Section 3.1 explains scalability in general. Section 3.2 explains the automatic scaling of microservices. Section 3.3 introduces load balancing, which is one of the essential technologies used for scaling. Section 3.4 discusses service discovery, and how microservices can reliably find and connect to each other. Finally, section 3.5 discusses data management and consistency in microservice systems.

3.1 Scalability

Scalability means being able to adapt to changing workloads by increasing or decreasing the system's resources based on the demand [5]. Scaling is needed when the current processing power of the system is not enough to handle all user requests without significant delays or loss of service. The emergence of cloud computing has made scalability possible for larger audiences by providing computing resources that can be deployed instantly on demand.

There are two different ways of scaling: vertical and horizontal scaling [14]. In vertical scaling, a single server's computing resources such as CPU and memory are increased. On the other hand, in horizontal scaling, the number of servers is increased and the workload is distributed among them. Horizontal scaling is commonly used with microservices [4]. When a high load is detected on a microservice, more replicas of the microservice can be created and the traffic will be distributed among them.

3.2 Automatic scaling

The load that the microservice application receives is often dynamic. The load can be higher at certain times, for example when launching a new

product or a discount sale at an online store. When the system receives a high amount of traffic, the corresponding microservices need to be scaled. This is done by increasing the amount of replicas of each microservice. On the other hand, when the traffic is low, the system needs to be scaled down by removing replicas. Scaling microservices manually is not feasible if the amount of traffic is unpredictable.

Automating scaling aims to scale the microservice automatically depending on the load it receives. The load can be measured by measuring the individual microservices and how much processing power they are using, for example, CPU and RAM usage.

Kubernetes [2] is a container orchestration platform created by Google. Kubernetes can run containerized applications on clusters consisting of multiple nodes. Kubernetes can be used to scale the microservice horizontally by creating multiple replicas of it. The scaling can be done automatically using a feature called HorizontalPodAutoscaler [1]. HorizontalPodAutoscaler monitors the CPU and memory usage metrics of the microservice and either increases or decreases the number of replicas depending on the thresholds set by the developer.

3.3 Load balancing

Load balancer [17] is a service that distributes incoming traffic on multiple servers in order to distribute the load. The load balancer is typically an HTTP reverse proxy server, but there exist lower-lever load balancers for raw TCP/UDP protocols too. Load balancing is commonly used when scaling microservices. When a microservice has multiple replicas, the traffic is distributed among them using a load balancer.

A load balancer distributes the traffic to its backends using load-balancing algorithms. The most common load-balancing algorithm is round-robin [17], which distributes the traffic uniformly on all backends circularly. The load balancer can also route all traffic from a single IP address to the specific backend, for example, to preserve the session state. In geographically distributed systems, the load balancer can distribute the traffic to the backend nearest to the user.

The disadvantage of using a load balancer is that it adds a single point of failure to the system. If the load balancer is down, the whole system will be inaccessible. Adding load balancers to a microservice system also increases the complexity of the system.

3.4 Service discovery

Microservices need to be able to communicate with each other. To make communication possible, microservices have to know the IP addresses and ports of the other microservices they need to communicate with. Microservices are usually running in environments where they are created and destroyed dynamically, such as container platforms or virtual machines [18]. Because of this, the IP addresses of the microservices are dynamic. Because of the changing addresses, it is not feasible to hard-code the IP addresses in the application code or configuration files.

Microservice systems use service discovery mechanisms to locate other microservices [19, 18]. Service discovery is commonly done using a service directory [18]. The service directory keeps track of the running microservices and their addresses.

There are two common ways for service discovery: server-side service discovery and client-side service discovery [18]. In server-side service discovery, a dedicated request routing server or load balancer component is used. When a microservice needs to communicate with another microservice, the request is sent to the routing server. The routing server communicates with the service directory and will direct the request to the target microservice. Usually, the routing server is a load balancer that can distribute the incoming requests over multiple scaled replicas of the microservice. This routing server component is usually integrated into the platform, such as in systems like Docker and Kubernetes. The benefit of this approach is that it doesn't need changes to the client's code. The disadvantage is that it adds additional network hop.

In client-side service discovery, the microservice has code that communicates with the service directory. The microservice uses this code to query the target microservices' address from the service directory. The microservice then makes a request directly to the destination. The benefit of this approach is that the connection to the microservice is direct, there are no additional hops needed like in the server-side approach. The disadvantage is that it requires custom code for connecting and communicating with the service directory.

There are existing solutions that provide service discovery. Container platforms such as Kubernetes [3] and Docker [9] offer service discovery features. Kubernetes uses the server-side service discovery method. Kubernetes has a Service type which acts as a global endpoint for accessing

the microservice associated with the Service. Kubernetes uses its internal DNS system to give the Service a DNS name which can be used by other microservices running in the cluster. Kubernetes routes the request to a pod defined in the Service. If the microservice is scaled using a ReplicaSet, Kubernetes provides load-balancing and distributes the incoming requests to the pods in the ReplicaSet.

3.5 Data consistency

One of the challenges of the microservice architecture is data management and consistency. Microservice architectures typically use a database-per-service approach [13], where each microservice has its own database. The approach of having multiple databases can cause data consistency problems when data needs to be updated or queried from multiple microservices in a single query [18].

Requests to a microservice application typically involve multiple microservices; the request could go to an API gateway, which will call the required microservices and combine their output. With monolithic applications using a single database, ACID (atomic, consistent, isolated, and durable) transactions can be used. ACID principles ensure that multiple tables can be modified safely.

In microservice systems with multiple databases, it is not possible to use transactions. The database operation must be executed on each microservice's database individually. Due to the lack of proper transactions, the ACID property atomicity is lost. The state of one microservice may then change in the middle of an update, resulting in an inconsistent state. Microservice systems generally aim for eventual consistency, which means that data might not be consistent at all times.

Saga [18] is a pattern used in microservice systems to help with executing transactions over multiple microservices. They are a way to maintain data consistency across microservices without distributed transactions. In sagas, the transactions are divided into steps divided to each microservice part of the saga. Sagas are implemented using message-passing between the microservices. In sagas, each microservice passes a message indicating if the transaction step was successful. After a successful step, the next step can proceed. If some step fails, a compensating transaction is created to undo the other steps.

The saga pattern can be used to ensure data consistency in microservice systems, but it has its challenges. Implementing sagas increases the

complexity of the application. Development and debugging of the application becomes more challenging.

4 Discussion

From the literature reviewed in this paper, the main challenges of microservice scalability are automatic scaling, load balancing, service discovery and data consistency. These challenges are mostly caused by the distributed nature of microservice architecture and the required network communications between the microservices.

It seems that modern microservice platforms such as Kubernetes can solve many of the discussed challenges in scalability. Kubernetes has an internal load balancer, service discovery system and capabilities for automatic horizontal scaling. But Kubernetes doesn't solve the data management challenges of the microservices. There has been a lot of research on microservices, but more research is needed on data management and consistency.

5 Conclusion

This paper explored microservices and their scalability challenges. The literature review conducted in this paper shows that microservice architecture offers more scalability than monolithic architecture. While the monolith application can be scaled simply by increasing copies of the application, it can be wasteful because the load usually targets specific parts of the application. On the other hand, microservices offer more flexible scalability possibilities, when only the parts of the application that are receiving the high load need to be scaled. This increases the efficiency of the system.

The improved scalability brought by microservices also brings challenges. The biggest challenges in scalability are automatic scaling, load balancing, service discovery and data consistency. Different solutions for the challenges were found. It was noted that microservice platforms such as Kubernetes have existing solutions for many of the challenges, including automatic scaling, load balancing and service discovery.

References

[1] The Kubernetes Authors. *Horizontal Pod Autoscaling*, 2024. url: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale.

[2] The Kubernetes Authors. *Overview*, 2024. url: https://kubernetes.io/docs/concepts/overview/.

[3] The Kubernetes Authors. *Service*, 2024. url: https://kubernetes.io/docs/concepts/services-networking/service/.

[4] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 10:20357–20374, 2022. doi: 10.1109/ACCESS.2022.3152803.

[5] André B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd International Workshop on Software and Performance*, WOSP '00, page 195–203, New York, NY, USA, 2000. Association for Computing Machinery. doi: 10.1145/350391.350432.

[6] 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. doi: 10.1007/978-3-319-67425-4_12.

[7] Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, and Larisa Safina. Microservices: How to make your application scale. In Alexander K. Petrenko and Andrei Voronkov, editors, *Perspectives of System Informatics*, pages 95–104, Cham, 2018. Springer International Publishing. isbn: 978-3-319-74313-4.

[8] Wilhelm Hasselbring. Microservices for scalability: Keynote talk abstract. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, ICPE '16, page 133–134, New York, NY, USA, 2016. Association for Computing Machinery. doi: 10.1145/2851553.2858659.

[9] Docker Inc. *Docker overview*, 2024. url: https://docs.docker.com/get-started/overview/.

[10] F5 Inc. *What Is an API Gateway?*, 2024. url: https://www.nginx.com/learn/api-gateway/.

[11] 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. doi: 10.1109/MS.2018.2141039.

[12] Miika Kalske, Niko Mäkitalo, and Tommi Mikkonen. Challenges when moving from monolith to microservice architecture. In Irene Garrigós and Manuel Wimmer, editors, *Current Trends in Web Engineering*, pages 32–47, Cham, 2018. Springer International Publishing. isbn: 978-3-319-74433-9.

[13] Rodrigo Laigner, Yongluan Zhou, Marcos Antonio Vaz Salles, Yijian Liu, and Marcos Kalinowski. Data management in microservices: state of the practice, challenges, and research directions. *Proc. VLDB Endow.*, 14(13):3348–3361, sep 2021. doi: 10.14778/3484224.3484232.

[14] Dan C. Marinescu. *Cloud Computing: Theory and Practice*. Morgan Kaufmann, Boston, 2013. doi: 10.1016/B978-0-12-404627-6.00010-5.

[15] Milica MATIC, Marija ANTIC, Istvan PAPP, and Sandra IVANOVIC. Optimization of mqtt communication between microservices in the iot cloud. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–3, 2021. doi: 10.1109/ICCE50685.2021.9427602.

[16] Katherine Nieman and Sayeed Sajal. A comparative analysis on load balancing and grpc microservices in kubernetes. In *2023 Intermountain Engineering, Technology and Computing (IETC)*, pages 322–327, 2023. doi: 10.1109/IETC57902.2023.10152023.

[17] Mazedur Rahman, Samira Iqbal, and Jerry Gao. Load balancer as a service in cloud computing. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pages 204–211, 2014. doi: 10.1109/SOSE.2014.31.

[18] C. Richardson. *Microservices Patterns: With examples in Java*. Manning, 2018. isbn: 9781617294549.

[19] Yuwei Wang. Towards service discovery and autonomic version management in self-healing microservices architecture. In *Proceedings of the 13th European Conference on Software Architecture - Volume 2*, ECSA '19, page 63–66, New York, NY, USA, 2019. Association for Computing Machinery. doi: 10.1145/3344948.3344952.

Weaknesses in the Tor network

Kalle Saarinen
kalle.saarinen@aalto.fi

Tutor: Tuomas Aura

Abstract

This paper investigates attacks on the Tor network and its users. The paper looks at weaknesses in the Tor network and attacks exploiting those weaknesses. This paper shows that even if Tor has its flaws it can still be used anonymously if enough people use it.

KEYWORDS: security, anonymity, onion routing, tor

1 Introduction

The Onion Router or Tor is a circuit-based network designed for anonymous communication via the Internet.

The Tor network is decentralized and consists of nodes also called relays, run by volunteers, through which a connection is made to the wanted website or service. This connection through the nodes is called a circuit. The circuit consists of three nodes with the first node being either a guard node or a bridge node, the second node being the middle node, and the last node being the exit node.

This paper is a literature review focusing on different kinds of attacks against the Tor network which either deanonymize its users or render the

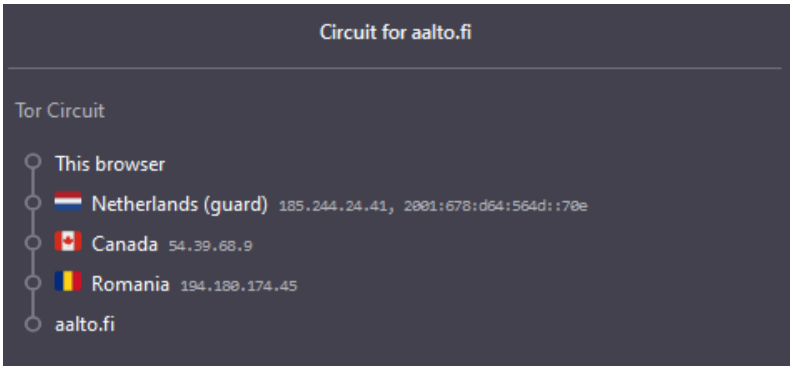


Figure 1. A circuit as shown in the Tor browser.

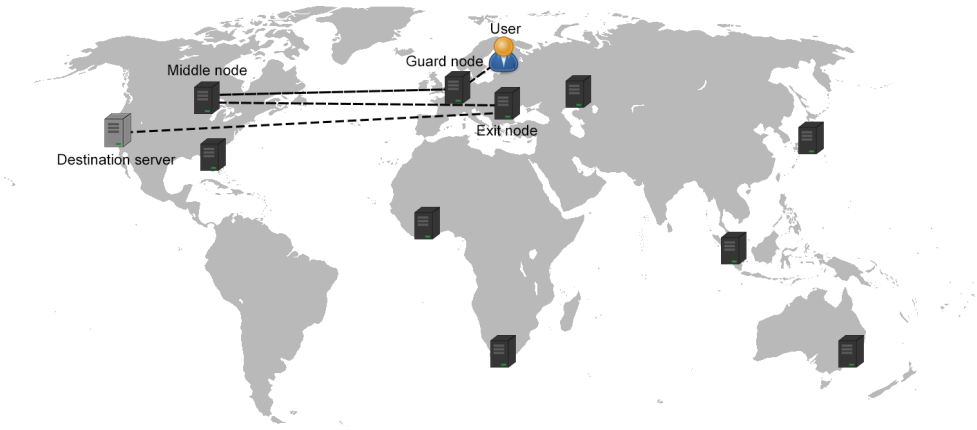


Figure 2. A map showing the circuit of figure 1. The number of nodes shown on the map does not reflect the number of nodes in the real world.

network unusable.

The rest of this paper is structured as follows. Section 2 will cover the different types of nodes in the Tor network and their purpose. Section 3 covers weaknesses in the network and what kind of attacks have been designed exploiting those weaknesses. Section 4 covers suggested and implemented defenses against these attacks. Section 5 covers the hosting requirements of nodes and the challenges that come with it. Section 6 is the discussion section and finally section 7 is the conclusion.

2 Types of nodes and their purpose

The network consists of four different kinds of nodes: guard nodes, middle nodes, exit nodes, and bridge nodes. The IP addresses of all non-bridge Tor nodes are public information, making it trivial to block all incoming or outgoing traffic to and from them. This is where the bridge nodes come in. A bridge node is not necessary for forming a circuit, but since their

IP addresses are not public information one can connect to a bridge node through which the user connect to the Tor network [5].

Figure 1 shows a circuit as it is shown in the Tor browser. The figure shows that the guard node is located in the Netherlands, the middle node is located in Canada and the exit node is located in Romania. Figure 2 shows that same circuit on a world map.

2.1 Guard nodes

Guard nodes, also called entry nodes, are usually the first node in a circuit, the other possible first node in a circuit being a bridge node. The IP address of guard nodes is public information meaning, it is possible to block access to them. In these cases, a bridge node is used instead of a guard node. When forming a new circuit the guard node does not always change. This is due to the fact that upon running the Tor browser for the first time, the browser chooses a few of nodes and uses those as its guard nodes [5]. These guard nodes are used for two to three months after which a new set of guard nodes is chosen [10].

2.2 Bridge nodes

Bridge nodes can be the first part of a circuit. Bridge nodes are usually only used when someone or something is preventing the user from directly connecting to a guard node. Since the IP addresses of all guard nodes are known an ISP or a government can block access to all of the guard nodes, a bridge node bypasses this since the IP addresses of bridge nodes are not public information meaning bridge nodes can be thought as more private guard nodes.

2.3 Middle nodes

Middle nodes, as the name suggests, are located in the middle, between the guard or bridge node and the exit node.

2.4 Exit nodes

Exit nodes are the last nodes on a circuit, these nodes connect to the destination server or service. When a Tor user connects to a service it looks like their connection is coming out of the exit node, this can cause issues for the operator of the node. This will be explained in section 5.1.

3 Weaknesses in the network

When using Tor there are three different parts which can be attacked. The client, the server, and finally the network itself [3]. Each of these parts have different weaknesses and are used differently depending on what an attacker wants to accomplish. Usual goals for an attacker would be to deanonymize a certain user, deanonymize users of a certain service, or take down parts of the network making in inaccessible. The attacker itself could be in any part of the network. They could be a client trying to take down nodes. They could be a server trying to deanonymize users of the service or they could be operating a malicious node trying to deanonymize users.

3.1 Deanonymizing users in the network

The simplest way to deanonymize a user of the Tor network is for the user's circuit to have the guard node and the exit node to be controlled by the same attacker [5]. The attacker can then correlate the traffic between the guard node and the exit node to determine the user's actual IP address, the IP address of the destination, and the content of the traffic. With Tor having thousands of nodes the chance of this happening naturally is very small. Other users are a crucial part in making Tor anonymous. The more traffic there is the harder it is to correlate the traffic in entry and exit nodes. Also the more nodes there are in the network the smaller the chance of getting a circuit with an attacker controlled guard node and exit node.

3.2 Super nodes

An attacker would need a huge resources to attack the network. This is where the so called *super nodes* come in. Super nodes are high bandwidth, high availability and long-lived nodes on the network. When creating a circuit, the circuit creation algorithm prefers these nodes over others due to their reliability.

Li et al. [8] show that the existence of super nodes creates weakness in the network and the authors design a new theoretical attack on the Tor network. In the paper, the authors discovered that 21 percent of the nodes in the network, are super nodes through which 66 percent of the traffic flows through. If an attacker were to control a small number of

super nodes they would have a much higher chance of deanonymizing users than if they were controlling the same number of regular nodes.

3.3 Centralization

Decentralization is an important factor in the Tor network. The more centralized the network is, the less anonymous it is. Abbott et al. [1] describe a method of deanonymizing users by having one of the user's circuits have an attacker controlled exit node and a later one having an attacker controlled guard node. It is important to note however that this attack requires the destination to use HTTP instead of HTTPS since it modifies the HTTP traffic.

The paper by Johnson et al. [7] came to the conclusion that if an attacker operates nodes with a total of 100 MiBps they have a 80 percent chance of deanonymizing a user in 6 months. If the attacker controls an Internet exhange point (IXP), they will can have a 95 percent chance to deanonymize a user in three months.

3.4 Denial-of-service

Denial of service attacks, can drive away users by making the network slow. The less users there are the harder it is to maintain anonymity within the network. As the traffic amount becomes smaller, it is easier to identify certain users using patterns or correlating their activity. Also with users leaving the network the number of legitimate nodes would also decrease. This would make it easier for an attacker to have bigger proportion of the total nodes under their control.

Barbera et al. [2] created a denial-of-service attack against Tor nodes called *CellFlood*. This attack sends specially crafted packets or cells to Tor nodes which the node has to process. This attack works because processing the cell takes four times longer than generating it. This is due to the cryptography used in these cells.

Jansen et al. designed a new attack called *the sniper attack* [6]. This attack could be used to disable arbitary Tor nodes. It was estimated by the authors that "a strategic adversary could disable all of the top 20 exit relays in only 29 minutes, thereby reducing Tor's bandwidth capacity by 35 percent". With the existence of super nodes an attack like the sniper attack is concerning. This attack can not only cause damage to the availability of the network but it can force its users to use Tor nodes in control

of the attacker, deanonymizing them that way.

Li et al. [8] discussed a new theoretical denial-of-service attack called the *loop attack*. This attack takes advantage of the fact that super nodes, which were discussed in chapter 4.1, exist in the Tor network. In this attack, the attacker needs to have a sizeable part of the network under their control. The attacker then blocks access to all the known super nodes, making the network extremely slow for the users. The loop in the name of this attack comes from the fact that, with a slow network, users will stop using it since it is not reliable or fast enough. With users leaving the network, it becomes even less secure. Node operators stop operating their nodes, making the network slower since the number of legitimate nodes decrease. The strong suit of this attack is the fact that it would be almost impossible to detect in the beginning and become detectable only after it has made users and operators leave the network due to its unreliability.

4 On the defense

According to Dingledine et al. [5] there is nothing that can be done about the small chance of a created circuit being compromised. As mentioned in section 2.1 the guard node is always chosen from a small subset of all guard nodes. This makes it so that if none of the chosen guard nodes are controlled by an attacker then it is impossible for the circuit to be fully compromised by an attacker. This lessens the chance of a user having a compromised circuit but this chance is never zero. The same thing is with *super nodes*. It is a design flaw that is basically impossible to fix. If there were no *super nodes* and every node essentially had a weight of one, an attacker could simply create an huge number of slow nodes. This would make the network unusable for its users.

Jansen et al. [6] propose three ways to counter *the sniper attack*, *Authenticated SENDMEs*, *Queue Length Limit*, and *Adaptive Circuit Killing*. The most effective out of these three is the *Adaptive Circuit Killing* method. This defense mechanism has been added to Tor [9].

Johnson et al. [7] discovered that increasing the time for choosing a new set of guard nodes significantly increased the time it took for an attacker to compromise a user. At the time of the paper that time was 30 days. As mentioned in section 2.1, this has been increased to over 60 days.

Barbera et al. [2] propose client puzzles as a solution to the *CellFlood* attack.

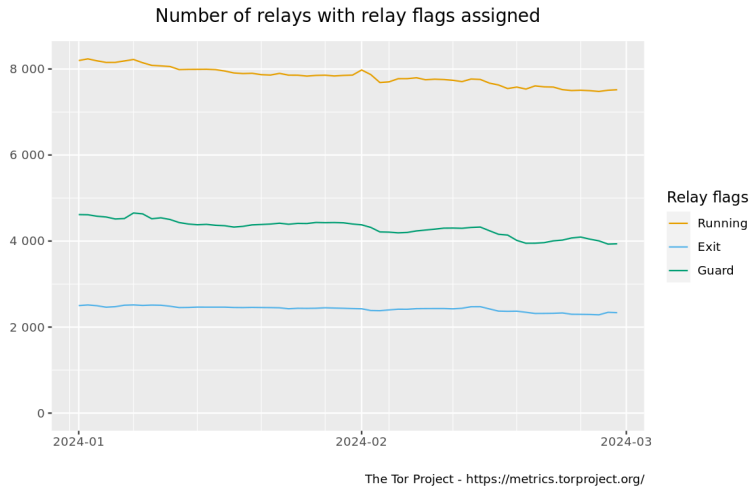


Figure 3. A chart showing the number of exit and guard nodes compared to the total number of nodes from 1.1.2024 to 29.2.2024 [11]

5 Operating different types of nodes

When someone wants to host a Tor node there are some choices the operator can make regarding their hosted node. The operator can set the exit policy of their node. When the exit policy is set to *reject *:**, the node will only act as a middle or guard node. The operator cannot directly decide to run a guard node. The reason as to why is explained in the next subsection.

5.1 Hosting requirements and challenges

The easiest type of node to operate is a bridge node since the IP address of a bridge node is not made public, services will not likely block the IP address of a bridge node from accessing their service.

Guard and middle nodes are nearly identical in terms of hosting. When operating a non-exit node, the node will act as a guard node if it meets a requirement to be one. This requirement is a 2 MB/s bandwidth both up and down. If this requirement is not met, the node will only act as a middle node [4].

Exit nodes are the most demanding to operate due to the fact that these nodes connect to the outside world. When a Tor user connects to a website, to the outside world it looks like the exit node operator is connecting to that website. If the website the user is connecting to hosts illegal content for, example, it looks like the node operator is accessing the illegal content. This can cause issues for the operator of the exit node. For this reason, most nodes on the network do not allow exiting on the node and

are middle or guard nodes. This can be seen in figure 3.

Because hosting a Tor node, especially an exit node, can affect the operators own internet usage not everyone wants to host a node. Especially users with a metered connection since a Tor node is required to be able to handle at least 100 Gigabytes of traffic up and down a month [4].

6 Discussion

Not every attack needs a countermeasure against it implemented in Tor itself. For example the attack by Abbott et al. [1] requires the server to be using HTTP instead of HTTPS. In 2007, when the paper was published, this was a bigger issue but nowadays the vast majority of websites use HTTPS making this attack very niche today.

An attacker with unlimited resources could take down the network with a denial-of-service attack. There is nothing that can be done in order to completely prevent this. If an attacker were to have more resources available than the enterity of the network it could take down the whole network. Even if an attacker could not take down the current network it could first try to weaken it by making operators not want to host their nodes anymore. With this in mind it would be interesting to know what percentage of exit nodes are operated by ordinary individuals since operating exit nodes can impact the operator's daily life. If one were to assume that a good number of exit nodes were operated by an attacker what would be the chances of having a compromised circuit.

7 Conclusion

Over the years different kinds of vulnerabilities have been discovered in Tor. Attacks have been designed using these vulnerabilities. These attacks are closely monitored, and the vulnerabilities fixed. The biggest issue with Tor is the fact that there is always a small chance that their circuit has been compromised. The only counter to this flaw is to have more nodes to make the selection of a compromised node slimmer and to have more users to make traffic correlation harder. Tor is actively maintained and upon the discovery of new vulnerabilities they are fixed as fast as possible.

Bibliography

[1] Timothy G Abbott, Katherine J Lai, Michael R Lieberman, and Eric C Price. Browser-based attacks on Tor. In *International Workshop on Privacy Enhancing Technologies*, pages 184–199. Springer, 2007.

[2] Marco Valerio Barbera, Vasileios P Kemerlis, Vasilis Pappas, and Angelos D Keromytis. Cellflood: Attacking tor onion routers on the cheap. In *Computer Security–ESORICS 2013: 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings 18*, pages 664–681. Springer, 2013.

[3] Enrico Cambiaso, Ivan Vaccari, Luca Patti, and Maurizio Aiello. Darknet security: A categorization of attacks to the Tor network. In *ITASEC*, pages 1–12, 2019.

[4] Tor Community. https://community.torproject.org/, 2024.

[5] Roger Dingledine, Nick Mathewson, Steven Murdoch, and Paul Syverson. Tor: The second-generation onion router (2014 draft v1). 2014.

[6] Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. The sniper attack: Anonymously deanonymizing and disabling the Tor network. In *NDSS*. Citeseer, 2014.

[7] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 337–348, 2013.

[8] Chenglong Li, Yibo Xue, Yingfei Dong, and Dongsheng Wang. "Super nodes" in Tor: existence and security implication. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 217–226, 2011.

[9] Nick Mathewson. https://blog.torproject.org/tor-02510-released-and-tor-023x-deprecated/, 2014.

[10] The Tor Project. Tor browser user manual. https://tb-manual.torproject.org/, 2024.

[11] The Tor Project. Tor metrics. https://www.metrics.torproject.org/, 2024.

Analysis of mental health discourse on social media

Kiira Karonen
kiira.karonen@aalto.fi

Tutor: Yunhao Yuan

Abstract

Mental health issues are a growing threat to public health globally. As a result, more people are turning to online communities and social media for support. This paper reviews natural language processing (NLP) methods and techniques employed in the study of mental health discourses on social media. Results of previous studies on the topic are summarized in order to provide an analysis on the nature and effects of mental health discussion online. The findings of the review suggest that online interactions can have a positive impact on individuals mental health. Analysing the linguistics, cognitive and behavioral features of online mental health discourse shows that online social support can have similar benefits as traditional treatment methods. Receiving emotional support and engaging with coping stories were found to result in psychosocial benefits for individual users. Thus, this paper concludes that social media and online communities can potentially be leveraged into useful tools in the battle against the global mental health crisis.

KEYWORDS: mental health, social media, NLP

1 Introduction

Mental health problems are a growing threat to public health globally. According to the World Health Organization (WHO), one in every eight people in the world were living with a mental disorder in 2019 [24]. Furthermore, there exists growing concern that mental health problems affect youth and young adults disproportionately. In 2021 WHO estimated that one in seven 10-19 year-olds experience mental health issues [23].

As a result of the rising prevalence of mental health problems, concern regarding the accessibility and adequacy of treatment grows. According to WHO the mental health conditions of adolescent "remain largely unrecognized and untreated" [23]. Concurrently, more people are turning to the Internet for support. Social support is an important factor in mental well-being [5], and online communities present a novel way to achieve the psychosocial benefits of social support that help combat mental health problems [18].

Anonymity, accessibility, interaction management and social distance are perceived as some of the advantages of online social support compared to traditional face-to-face interaction [22]. Internet forums are seen as particularly desirable sources of social support for people struggling with illnesses associated with stigma, such as depression. Affordability and accessibility for people living in rural areas are also considerable advantages of online support compared to traditional approaches. [16]

Seeking and providing social support online is a relatively novel phenomenon resulting from the emergence of the Internet. To better understand the phenomenon and its effects on individuals and public health, approaches to measure and analyze conversations surrounding mental health on social media are being developed. This literature review examines the methods and techniques that are applied to study mental health discourses on social media computationally. The second objective of this paper is to summarize the results of previous studies on the topic and thereby analyze the nature and effects of mental health discussion online.

The structure of the paper is as follows. Section 2 explores previous research on the topics of social support in the context of mental well being as well as characteristics of mental health discourse online. Section 3 discusses the methods and techniques that can computationally be applied to study mental health discourse on social media. Some interesting findings from previous research into online mental health discussion are

examined in section 4. Section 5 discusses the limitations and other considerations in the research of mental health discussions online. Finally, section 6 concludes the review.

2 Background

Social support can be seen as one of the key factors promoting health and well-being [12]. The benefits of traditional face-to-face social support extend to the context of social media as well. The results of a 2013 study by Oh et al. [12] reveal that social networking sites such as Facebook can function as effective spaces for supportive interactions, and therefore contribute to enhancing users health self-efficacy and capability to manage ones own health. The potential of online communities as health promoting resource is recognized especially among younger generations who live increasingly digital lives [11].

Social support is recognized as a particularly significant element in coping with mental health struggles as it can reportedly have a protective effect against the negative consequences of mental disorders [5]. Through the adoption of social media, dedicated online mental health communities, where individuals who suffer from mental disorders seek information, advice and support, have increasingly appeared in online spaces [5] [10]. These communities provide support seekers the psychosocial benefits of social support that in turn can aid in coping with mental health issues [18].

Individuals who struggle with stigmatized health conditions can purposely avoid situations where the state of their health may be revealed to others [2]. Thus online discourse may appear more appealing for people suffering from issues associated with stigma and shame than traditional face-to-face interaction. Consequently, anonymity is recognized as an important property of online mental health communities, as it helps individuals to make self-disclosures on stigmatized topics less hesitantly [18] [10]. On the other hand anonymity in online contexts is linked to reduced accountability, which in turn can result in negative behavior such as bullying and harassment [1] [21].

In addition to anonymity, a 2002 study by Walther and Boyd [22] recognized three other attractive features of computer-mediated social support first of which is accessibility. Accessibility allows support seekers to apply for support from anywhere regardless of the time of day, which

is a particularly significant advantage for those who reside in geographically remote areas [16]. The constant availability of informational and emotional support may lower the threshold for interaction, compared to making an appointment with a healthcare professional. Because of this, the accessibility of online support could also be seen as a factor that eases the burden on the healthcare sector. Accessibility also relates to the affordability of social support provided by online forums [16], which can be considered a significant factor on the global scale.

The other factors identified by Walter and Boyd include social distance and interaction management [22]. Interaction management in the context of mental health discourse means that both those seeking support and those giving it have the time to reflect on their messages and manage the situation according to their preferences. Additionally, social distance in computer-mediated relationships allows improves stigma management and candor in contrast to face-to-face communication, according to the study.[22]

Online mental health communities rely on peer to peer interaction for support. Peer support in the traditional sense is successfully applied by many well-functioning mental health services such as Alcoholics Anonymous [21]. However, one of the challenges of online social support is that in the online context supporters are expected to temporarily assume the role of a psychological counselor when helping those seeking support. [21] As a result, the quality of interactions can vary and there is no guarantee that support-seekers receive appropriate assistance.

Some of the proposed solutions to improve the quality of mental health discourse online include moderation and guided chats [21] [13]. A 2021 study by Wadden et al. found that moderation in online mental health conversations "improved civility, supportiveness, and coherence" [21]. The results of the study indicate that moderation could potentially serve as a useful and scalable tool in the global mental health crisis [21]. Another proposed solution by O'Leary et al. are chats guided by prompts based on psychotherapy skills. The study conducted in 2018 discovered that both guided and unguided chats succeeded to reduce symptoms of anxiety. However, guided chats were found to perform better in providing informational support and insights, while unguided chats directed towards distractions from troubles rather than solutions and emotional support. [13]

3 Methods

Natural Language Processing (NLP) is a central concept in the research into computer-mediated mental health discourse. NLP is a field of machine learning that employs computers to develop models of how people understand and use language [4]. The availability of natural language use enabled by social media as well as the rapid evolution of computational resources has transformed the language analysis field [19]. As a result, researchers are increasingly able to connect everyday language use with social, cognitive, and behavioral phenomena [19]. This chapter introduces relevant NLP techniques and models in the study of text-based online mental health discourse. The choice of a suitable method for analyzing online mental health conversations depends on several factors, such as the characteristics of the data, the specifications of the NLP task and the computational resources available.

3.1 LIWC

Linguistic Inquiry and Word Count (LIWC) is a text analysis software created by Pennebaker et al. [14] and a well-established tool in the research into text based online interactions. Its operation builds upon the LIWC psycholinguistics lexicon, based on which words can be classified into linguistic, psychological and content categories validated by independent judges [16]. In the context of online mental health discussion LIWC can be applied in analysing both linguistics and sentiment features. According to the 2010 analysis by Tausczik and Pennebaker, LIWC can accurately identify emotion in text and rate positive and negative emotion words at a level similar to human evaluations [19].

3.2 LDA

Topic modeling is an NLP technique often employed in the study of mental health discourse on social media. Latent Dirichlet Allocation (LDA) is a probabilistic model used to represent a corpus [3]. The basic idea of LDA is that documents are treated as random mixtures of various topics, and each topic is represented by a distribution over words [3] [17]. Topic modeling applied to social media analytics helps create an understanding of discussions and reactions between users in online communities. It is useful in uncovering patterns that explain peoples posting behavior on

platforms such as Facebook and Twitter. [9]

3.3 VADER

Sentiment analysis is another valuable tool in the analysis of online mental health discussion. According to Tausczik and Pennebaker "the degree to which people express emotion, how they express emotion, and the valence of that emotion can tell us how people are experiencing the world" [19]. Analyzing the sentiment of social media content presents challenges due to the high volume of data, contextual sparseness, and abbreviated language. The LIWC lexicon described in part 3.1 is widely used but may not always be suitable for social media. VADER (Valence Aware Dictionary for sEntiment Reasoning) has been developed to address these challenges, offering a sentiment lexicon attuned to microblog-like contexts and incorporating grammatical rules for improved accuracy. VADER performs exceptionally well in social media sentiment analysis, outperforming individual human raters and retaining the benefits of LIWC while being more sensitive to social media expressions. [8]

3.4 GloVe

GloVe (Global Vectors for Word Representation) is a word embedding technique that learns a vector representations of words in a continuous vector space with relevant substructure. GloVe captures fine-grained semantic relationships between words based on their co-occurrence statistics in a text with an accuracy up to 75 percent. The paper by Pennington et al. published in 2014 found that GloVe outperforms other word embedding methods other on several NLP tasks. [15]

3.5 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a pretrained fine-tunable model, which has quickly become a state-of-the-art tool for varying NLP tasks [7]. BERT is a deep learning model that, According to Tenney et al. (2019), can "represent the types of syntactic and semantic abstractions traditionally believed necessary for language processing, and moreover that they can model complex interactions between different levels of hierarchical information." [20]

4 Findings

The methods presented in section 3 have been applied in several studies investigating the effects of mental health discussion on social media. This paragraph aims to present some important findings from relevant research and describe the methods used to obtain the results.

The findings of a 2023 study by Yuan et al. [25] indicate that individuals interacting with suicide-related coping stories on social media can gain beneficial psychosocial outcomes. The research employs the LIWC lexicon combined with SVM classifiers to investigate the existence of the Papageno effect on social platforms such as Twitter. The Papageno effect refers to the phenomenon where media can have positive effects on individuals who are struggling with suicidal ideation. The results of the study demonstrate that the benefits of the Papageno effect carry over to the online context as well: "we observe statistically significant psychosocial (affective, behavioral, cognitive) shifts in individuals after engaging with coping story posts". Although the study did not measure the change in suicidal thoughts directly, but the findings are based on the interpretation of other measures, the results are nonetheless indicative of the promising role of social media in mitigating suicide ideation. [25]

Another 2023 paper by Kim et. al [10] examines social media as a potentially effective space for social support on mental health related concerns. The study applies fined-tuned pre-trained BERT models as well as topic modeling techniques such as LDA to explore the characteristics and behavior of both support-seekers and supporters in online mental health communities. The results of the analysis provide indications of what mental health support in Internet communities is like and how it can be leveraged effectively. These findings show a direction for mental health research in the online context and can serve as a guideline for designing safer and more efficient virtual spaces in the future. For instance one of the main findings of the paper states that the users in online mental health communities prefer emotional support over informational support, from which Kim et al. conclude that dedicated mental health groups can serve as active spaces of support with an emotional atmosphere. [10]

A 2017 study by De Choudhury and Kiciman [5] found similar results regarding the characteristics of social support in online mental health communities. The study employed a token-based machine learning model to analyze the language of comments in mental health communities on

Reddit. The paper found that counter-beneficially associated tokens occur in comments that human raters had coded to consist of informative or instrumental support, while comments classified as emotional by the raters were more likely to contain positively associated tokens. [5]

Findings from research suggest that emotional support in online communities can lead to psychosocial benefits. However, online support for mental health problems relies mostly on peer-to-peer interaction, so investigating methods to help supporters provide helpful responses is crucial. A 2020 study by Saha and Sharma [18] focuses on factors that contribute to effective support in online mental health communities. By applying the LIWC lexicon and SVM classifiers to data collected from TalkLife the study found that positivity, dynamicity, adaptability and diversity are important features of responses associated with beneficial outcomes. Interestingly immediacy and quantity of responses were shown to have insignificant effects in the online context. [18]

Naturally, the characteristics of online mental health discourse are strongly influenced by user-specific features. According to a 2017 paper by De Choudhury et al. [6] both gender and cultural background contribute considerably to the mental health related content shared by users. The study observed differences in self disclosures through data collected from Twitter via linguistics analysis employing LIWC and topic modeling using LDA. The results state that males are more likely to convey higher negativity while also expressing lower desire for social support than females. On the other hand, females tend to express higher sadness and anxiety according to the paper. Cultural differences were found to be most prominent in the users' desire to communicate negative emotions or distress. The research into the relationship of demographic features of users and mental health discourse on the Internet helps guide the design of gender and culture-aware online spaces. [6]

5 Discussion

The findings from research into mental health discourse on social media suggest that online interactions can have a positive impact on individuals mental health. Thus social media can potentially be leveraged into a useful tool in the battle against the global mental health crisis.

However, the text-based analysis methods referenced in this paper can quickly become insufficient in the context of social media, as content

on many online platforms extends over traditional text based messaging. Digital content often combines text with pictures, videos, emojis, emoticons and memes, which is why solely text-based analysis cannot always produce a comprehensive interpretation of interactions. Moreover, the online context presents additional challenges when analyzing mental health discourse, as the language and context on social media platforms can be complex and their evolution is rapid. Interpreting nuances, slang and abbreviations commonly used in online communication adds difficulty to the analysis process.

Another limitation of this paper is that most of the reviewed research focuses only on mental health discourse in English. Language and culture undoubtedly influence discussion on the Internet, so the methods and findings described in this paper may not be applicable to other linguistic contexts. As a result, future research into mental health discourse online should aim to include a wider range of languages and cultural dimensions.

Furthermore, the studies reviewed in this paper focus their analysis on direct communication on social media, as a result of which passive engagement is overlooked. Users may seek support through consuming mental health related online media without directly taking part in conversations. The research methods considered in this article are mainly applied to text-based data. Combining linguistics and cognitive techniques with behavioral analysis could provide more thorough insights into the impacts of mental health discourse on passive users as well.

Another limitation in the study of mental health discourse online is the huge amount of data on social media platforms. The sheer amount of information imposes practical challenges, as social media data often contains noise and bots, as a result of which preprocessing requires considerable resources and effort. In addition, ethical and social considerations must be taken into account due to the nature of social media posts, as they may contain sensitive and personal information. Ensuring privacy and adhering to ethical guidelines are indispensable, although they may increase the complexity of the research process.

6 Conclusion

The increase in computational power and resources has allowed for more sophisticated and efficient methods in the field of NLP to emerge, en-

abling deeper understanding of social media data. Research employing techniques for analysing the linguistics, cognitive and behavioral features of online mental health discourse shows that online social support can have similar benefits as traditional treatment methods. However, these promising results from research on mental health discussion in social media cannot yet be generalized to all users due to linguistic and cultural diversity. Passive engagement is another phenomenon within the online mental health discourse context that requires further research. Nonetheless, the studies have demonstrated that social media has undeniable potential as a tool for mental health related social support. The psychosocial benefits provided by online mental health communities allow the groups to be harnessed in the battle against the mental health crisis alongside traditional treatment.

References

[1] Nazanin Andalibi, Oliver L Haimson, Munmun De Choudhury, and Andrea Forte. Understanding social media disclosures of sexual abuse through the lenses of support seeking and anonymity. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 3906–3918, 2016.

[2] Magdalena Berger, Todd H Wagner, and Laurence C Baker. Internet use and stigmatized illness. *Social science & medicine*, 61(8):1821–1827, 2005.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[5] Munmun De Choudhury and Emre Kiciman. The language of social support in social media and its effect on suicidal ideation risk. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 32–41, 2017.

[6] Munmun De Choudhury, Sanket S Sharma, Tomaz Logar, Wouter Eekhout, and René Clausen Nielsen. Gender and cross-cultural differences in social media disclosures of mental illness. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*, pages 353–369, 2017.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[9] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia tools and applications*, 78:15169–15211, 2019.

[10] Meeyun Kim, Koustuv Saha, Munmun De Choudhury, and Daejin Choi. Supporters first: Understanding online social support on mental health from a supporter perspective. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW1):1–28, 2023.

[11] Brad Love, Brittani Crook, Charee M Thompson, Sarah Zaitchik, Jessica Knapp, Leah LeFebvre, Barbara Jones, Erin Donovan-Kicken, Emily Eargle, and Ruth Rechis. Exploring psychosocial support online: a content analysis of messages in an adolescent and young adult cancer community. *Cyberpsychology, Behavior, and Social Networking*, 15(10):555–559, 2012.

[12] Hyun Jung Oh, Carolyn Lauckner, Jan Boehmer, Ryan Fewins-Bliss, and Kang Li. Facebooking for health: An examination into the solicitation and effects of health-related social support on social networking sites. *Computers in human behavior*, 29(5):2072–2080, 2013.

[13] Kathleen O'Leary, Stephen M Schueller, Jacob O Wobbrock, and Wanda Pratt. "suddenly, we got to become therapists for each other" designing peer support chats for mental health. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–14, 2018.

[14] James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[15] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[16] Nairan Ramirez-Esparza, Cindy Chung, Ewa Kacewic, and James Pennebaker. The psychology of word use in depression forums in english and in spanish: Testing two text analytic approaches. In *Proceedings of the international AAAI conference on web and social media*, volume 2, pages 102–108, 2008.

[17] Philip Resnik, William Armstrong, Leonardo Claudino, Thang Nguyen, Viet-An Nguyen, and Jordan Boyd-Graber. Beyond lda: exploring supervised topic modeling for depression-related language in twitter. In *Proceedings of the 2nd workshop on computational linguistics and clinical psychology: from linguistic signal to clinical reality*, pages 99–107, 2015.

[18] Koustuv Saha and Amit Sharma. Causal factors of effective psychosocial outcomes in online mental health communities. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 590–601, 2020.

[19] Yla R Tausczik and James W Pennebaker. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54, 2010.

[20] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[21] David Wadden, Tal August, Qisheng Li, and Tim Althoff. The effect of moderation on online mental health conversations. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 15, pages 751–763, 2021.

[22] Joseph B Walther and Shawn Boyd. Attraction to computer-mediated social support. *Communication technology and society: Audience adoption and uses*, 153188(2), 2002.

[23] WHO. Mental health of adolescents, fact sheet. https://www.who.int/news-room/fact-sheets/detail/adolescent-mental-health, nov 2021. Accessed: 2024-03-01.

[24] WHO. Mental disorders, fact sheet. https://www.who.int/news-room/fact-sheets/detail/mental-disorders, jun 2022. Accessed: 2024-03-01.

[25] Yunhao Yuan, Koustuv Saha, Barbara Keller, Erkki Tapio Isometsä, and Talayeh Aledavood. Mental health coping stories on social media: A causal-inference study of papageno effect. In *Proceedings of the ACM Web Conference 2023*, pages 2677–2685, 2023.

Games for cognitive abilities of elderly people

Kimi Kuru
kimi.kuru@aalto.fi

Tutor: Sanna Suoranta

Abstract

This paper focuses on identifying the key design principles of digital games aimed at enhancing cognitive abilities, in the elderly. This paper researches how digital games can be tailored to the needs of elderly people while enhancing their cognitive abilities. The paper reviews existing literature to identify key principles and potential areas for improvement in the design of cognitive games specifically intended for elderly users. The paper emphasized the importance of a user-centered design approach, considering the physical and cognitive capabilities of the elderly. Key aspects such as game design principles, gamification of cognitive activities, technological and development considerations are discussed. The paper also discussed the balance between engagement and accessibility in game design, ensuring that games are engaging without being overwhelming. With elements from technology and psychology, this paper contributes to the growing field of digital games aimed at sustaining and improving the cognitive health of the elderly. The paper integrates basic concepts for from different fields for further understanding on how to develop effective games for cognitive abilities of elderly people. The literature review and discussion in this paper are needed by developers, designers, and researchers working towards creating impactful cognitive games for an aging population.

KEYWORDS: Gamification, Cognitive Improvement, Elderly, Digital Games

1 Introduction

As people are aging, maintaining their cognitive abilities, such as reasoning, learning and remembering, becomes a concern. According to research, one effective way to prevent decline of cognitive abilities is by continuously learning new skills [16]. Notably, dancing is known to be one of the most effective activities in maintaining these abilities [18]. Moreover, games can be entertaining but they can also help improve cognitive abilities, especially after events like strokes [15].

However, implementing games that would benefit aging people comes with challenges. These challenges include making the games suit different cognitive levels [13], overcome issues with using technology [9], and ensure that older adults stay interested [12], [4]. As the number of older people increases, it is important to understand different ways that can help with maintaining their cognitive health.

The primary aim of this paper is to explore the essential components of non-entertaining games aimed at enhancing cognitive abilities in elderly individuals. This paper reviews existing literature to identify key principles and potential areas for improvement in implementation of cognitive games for the elderly with a special focus on game design considerations for successful cognitive improvement. This paper is organized as follows. Section 2 provides background to the topic by reviewing existing literature on main methods to prevent cognitive decline and the role of games in cognitive improvement. Section 3 discusses gamification and important game design factors when designing games for cognitive improvement. Section 4 provides discussion around the findings. Finally section 5 provides a conclusion.

2 Prevention of cognitive impairment

This section provides background about current methods and principles that are used for preventing cognitive decline and improve cognitive abilities. Section 2.1 explains the importance and challenges of sustained engagement for cognitive improvement. Section 2.2 provides examples of some known interventions for cognitive decline. Finally, Section 2.3 discusses the role of digital games in cognitive intervention.

2.1 Sustained mental engagement

Aging rises concerns about cognitive decline of elderly people thus creating a need for research focused on understanding how to prevent the cognitive decline. A research shows the importance of continuous cognitive engagement as a method to better maintaining cognitive abilities of the elderly [16]. The research also suggests that sustained mental effort improved cognitive abilities of elderly people by, for example, learning a new skill.

Elderly people usually are not very engaged with new technologies and can have difficulties learning them [3] which is a problem when trying to keep sustained engagement with digital games. Furthermore, digital platforms nowadays have almost endless possibilities for different types of games. Thus, it is important for elderly people to be able to utilize the possibilities of digital games. However, research suggests that external support with technology is essential for new technology adoption among elderly people [3]. This means that additional help is needed for the elderly to achieve sustained engagement with digital games.

2.2 Effective cognitive interventions

Research shows improvement of cognitive abilities among elderly people in traditional activities not involving the use of technology. Research focused on leisure activities suggests that activities such as dancing can reduce the risk of dementia when actively exercised [18]. In addition, such activities can be easy to maintain as they might not be too serious or difficult for the elderly. In contrast, playing digital games requires the use of technology which can be more difficult for elderly people.

As mentioned in section 2.1 sustaining cognitively demanding activities improve cognitive abilities for older people. Notably, Gallo and Abutalebi [7] suggest that learning a new language is one good way of improving cognitive abilities.

2.3 Role of games in cognitive intervention

Digital games offer many possibilities for developing games for improving cognitive abilities. Studies have shown that both entertaining and non-entertaining video games can enhance various cognitive functions in elderly people [14], [5], [2]. Entertaining games can keep players en-

gaged due to their entertainment value, while also improving cognitive functions [5]. However, entertaining do not usually focus on improving specific cognitive skills. Non-entertaining games are specifically designed to improve skills such as memory and problem solving. Thus the focus in design of such non-entertaining games requires consideration from different points of views such as psychology, technology and game design.

When designing games to improve cognitive functions, it is important to consider factors such as age-appropriate design, accessibility, personalization for different level players, engagement and motivation. Most of the considerations in the technical design of the game requires considering the diverse possibly declined functions of elderly people. In addition to considering functions of elderly people in non-entertaining games, it is important to consider how the actual game would improve cognitive functions of the players. An early example of a game designed for cognitive improvement is Brain Age. Brain Age is a digital game aiming to improve cognitive function that was developed by utilizing research on effective cognitive training [14]. Brain Age includes different games that give the user challenges like math problems [14]. Game design and technology considerations need to take into account how the game improves cognitive functions and build functionality around that.

3 Desigining games for cognitive improvement

This section discusses gamification in learning and focuses on essential factors in game design that need to be considered when creating games for cognitive improvement of elderly people. Section 3.1 discusses gamification of cognitive activities. Section 3.2 provides high-level principles for designing games for cognitive improvement. Section 3.3 discusses implications of the design principles to technology and development of the games.

3.1 Gamification of cognitive activities

Gamification refers to the process of adding game-like elements to activities or tasks in non-game contexts to encourage participation and engagement. In the context of gamification, Deterding et al. [6] separates the game-like elements as follows. Game interface design patterns are common components in successful interaction design such as badges, leader-

boards and levels. Game design patterns are common parts of game design concerning gameplay such as time constraints and resources. Finally, game design principles and heuristics are guidelines to analyzing design problems.

The gamification of cognitive activities could include, for example, gamification of learning a new language, since it is known to improve cognitive abilities as mentioned in 2.2. Accordingly a study suggests that Duolingo, a language learning application could benefit cognitive functions of older adults [10]. Also, gamification of an activity could make activities more engaging and accessible, which is important since sustained mental engagement is essential for improving cognitive functions. A systematic review by Lumsden et al. [10] suggests that gamified cognitive training improves player motivation in many instances.

3.2 Game design principles for cognitive improvement

Game design of non-entertaining games for cognitive improvement of elderly should focus on cognitive stimulation meaning that games should replicate traditional activities proven to be beneficial or create new stimulating activities with the help of technology [13]. In addition, games can include functionality to assess and record player performance to detect cognitive levels of players and adapt difficulty. This approach provides an experience that continuously challenges and engages the user, thus maximizing cognitive benefits the game potentially gives. Games like "Lumosity", a brain training application, demonstrate this principle by adapting to the user's performance to provide personalized brain-training experiences [1].

Furthermore, game design for elderly people should focus on tailoring a game to the specific needs and abilities of elderly players [13]. This involves creating games with good accessibility, usability, and gradual difficulty levels to account for different cognitive levels and technological familiarity [17]. Games for cognitive intervention should be accessible on multiple platforms, including smartphones, tablets, and computers. The use of touch-screen interfaces can be beneficial for the elderly, who may find them easier and more intuitive to use than traditional mouse and keyboard controls [17]. Also, given possible physical and sensory limitations, games must be easy to use for all players.

Meza-Kubo and Morán [13] emphasize that Games should be designed to give immediate and constructive feedback. Instant feedback makes el-

derly aware of their results and can possibly notify caregivers of any need for assistance. In a systematic literature review by Martinho et al. [11] most of the studies that included serious games, widely used game design elements such as feedback, progression, time constraints and scores. It was observed in the review that these feedback elements provided effective indicator of real time performance to the elderly person during a session of playing a game.

Game design should also focus on making the games motivating. The game should include enough cognitive activities to select appropriate cognitive activity for the player according to the player's preferences and needs [13]. Also including motivators in the game according to player's preferences can help staying motivated [13]. Additionally, introducing novel stimuli and new environments can be beneficial for cognitive health and keep the game interesting [2].

According to the systematic review by Martinho et al. [11], many gamified activities use game elements like badges, rewards and social interaction mechanisms. The review states that these elements were included as they improve the entertaining features of the game, which could also contribute to making the game more motivating.

3.3 Technology and development considerations

As stated in section 3.2, games should be designed to fit specific needs of the elderly. Developers need to prioritize user-centered and personalization design principles [11]. This could involve understanding the physical and cognitive capabilities of the elderly, including potential limitations such as reduced motor skills and weaker cognitive abilities. In addition, features like larger buttons and text, simplified menus, and voice assistance can make the games more accessible [17]. User interface designers and developers need to comply to basic accessibility requirements and consider the additional limitations of players.

The software should be highly customizable to take into account varying cognitive abilities and preferences of the elderly [13]. This might involve creating techniques that automatically adjust the difficulty level of the game based on the player's performance. The software architecture should allow for flexibility in adding features that serve different kinds of players. For example, the use of machine learning and artificial intelligence could be used to adapt the game for different players.

Implementing feedback and analytics mechanisms can help in moni-

toring the players progress and providing feedback [13]. Developers and designers can develop the software to allow for sharing of the progress of the player to possible caregivers.

To increase accessibility, developers could consider cross-platform development. Cross-platform development allows the game to be used on various devices, taking into account the preferences and accessibility of different users. Thus, game designers and developers should consider utilizing tools that allow developing cross-platform compatible software.

In a paper, Gerling et al. [8] point out that the use of different technology may be limited for due to age related difficulties so different options for input devices need to be available. According to Gerling et al. features of input devices such as small buttons or devices allowing parallel interaction may be hard for the elderly to use. In addition, elderly people can have difficulties understanding and accepting new technologies and their benefits [11].

4 Discussion

This paper has reviewed various aspects of non-entertaining games designed for cognitive improvement in elderly individuals. The key considerations in designing such games include examining gamification of cognitive activities, game design principles for cognitive improvement, technology and development considerations.

Gamified applications have shown to improve cognitive abilities in the elderly [10]. The improvement could be the result of the characteristics of the underlying activity. However, gamification also improves player motivation and engagement [10]. With the improvement in motivation, gamification could help elderly people keep sustained mental engagement in the cognitive activity, which is a key part in cognitive improvement as mentioned in section 2.1.

Essential game design principles in developing games for the cognitive improvement of the elderly include selecting or creating an activity for cognitive improvement, tailoring the game to fit the physical and cognitive needs of the elderly, creating feedback mechanisms and making the game engaging to the user [13]. These considerations can be transformed into elements of the game interface and game element design. These elements could include mechanisms to assess player performance and adjust difficulty, intuitive and accessible interface elements such as big buttons,

mechanisms for receiving feedback and adjustments for different preferences.

Feedback mechanisms and progress analytics are other important aspects, allowing for real-time monitoring and adjustments based on individual performance. This not only helps in personalizing the experience but also provides valuable insights for caregivers.

The design principles challenge the possibilities in using technology and developing the games. Input devices need to be customizable and simple to use, for example, a touchscreen tablet with a simple interface. To account for accessibility, different types of devices should be available and the game should be developed for multiple platforms. In addition, the game software should allow for flexibility because of the varying cognitive needs, which could mean, for example, developing a platform of multiple cognitive games aimed at different groups of people.

5 Conclusion

This paper has discussed the essential parts of non-entertaining games for cognitive improvement in elderly people. The essential parts were discussed by gathering game design principles that were found to be important for cognitive improvement of elderly players.

Maintaining the cognitive health of the older population is important. Finding new ways of incorporating technology in helping maintain the abilities is important as today technology is increasingly used for different applications. Digital games can improve accessibility, engagement and motivation of traditional cognitive activities that are proven to improve cognitive functions in the elderly. Furthermore, digital games can create new activities that would not have been possible traditionally.

In conclusion, the development of non-entertaining games for cognitive improvement in elderly people is a important area of research that includes technology, game design and psychology. there is a need for more extensive research to understand the long term impacts and effectiveness of these games, and aligning game design to cognitive needs of the elderly people. The success of these games depends on their ability to engage users in meaningful activities that are both enjoyable and beneficial to their cognitive health. As the elderly population continues to grow, the importance of improving cognitive abilities becomes increasingly apparent, offering a potential to improved cognitive abilities for elderly people.

References

[1] A. Al-Thaqib, F. Al-Sultan, A. Al-Zahrani, F. Al-Kahtani, K. Al-Regaiey, M. Iqbal, and S. Bashir. Brain training games enhance cognitive function in healthy subjects. *Medical science monitor basic research*, 24:63–69, 2018. doi: 10.12659/msmbr.909022.

[2] J. A. Anguera, J. Boccanfuso, J. L. Rintoul, O. Al-Hashimi, F. Faraji, J. Janowich, E. Kong, Y. Larraburo, C. Rolle, E. Johnston, and A. Gazzaley. Video game training enhances cognitive control in older adults. *Nature*, 501(7465):97–101, 2013. doi: 10.1038/nature12486.

[3] Yvonne Barnard, Mike D. Bradley, Frances Hodgson, and Ashley D. Lloyd. Learning to use new technologies by older adults: Perceived difficulties, experimentation behaviour and usability. *Computers in Human Behavior*, 29(4):1715–1724, 2013. doi: 10.1016/j.chb.2013.02.006.

[4] Johnny Salazar Cardona, Jeferson Arango Lopez, Francisco Luis Gutiérrez Vela, and Fernando Moreira. Meaningful learning: motivations of older adults in serious games. *Universal Access in the Information Society*, pages 1–16, 2023. doi: 10.1007/s10209-023-00987-y.

[5] G. D. Clemenson, S. M. Stark, S. M. Rutledge, and C. E. L. Stark. Enriching hippocampal memory function in older adults through video games. *PloS one*, 390, 2020. doi: 10.1016/j.bbr.2020.112667.

[6] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining gamification. volume 11, pages 9–15, 2011. doi: 10.1145/2181037.2181040.

[7] Federico Gallo and Jubin Abutalebi. The unique role of bilingualism among cognitive reserve-enhancing factors. *Bilingualism: Language and Cognition*, page 1–8, 2023. doi: 10.1017/S1366728923000317.

[8] Kathrin Maria Gerling, Frank Paul Schulte, Jan Smeddinck, and Maic Masuch. Game design for older adults: Effects of age-related changes on structural elements of digital games. In *Entertainment Computing - ICEC 2012*, pages 235–242. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-33542-6_20.

[9] Wijnand Ijsselsteijn, Henk Herman Nap, Yvonne de Kort, and Karolien Poels. Digital game design for elderly users. In *Proceedings of the 2007 Conference on Future Play*, Future Play '07, page 17–22. Association for Computing Machinery, 2007. doi: 10.1145/1328202.1328206.

[10] J. Lumsden, E. Edwards, N. Lawrence, D. Coyle, and M. Munafò. Gamification of cognitive assessment and cognitive training: A systematic review of applications and efficacy. *JMIR Serious Games*, 4(2):11, 2016. doi: 10.2196/games.5888.

[11] Diogo Martinho, João Carneiro, Juan M. Corchado, and Goreti Marreiros. A systematic review of gamification techniques applied to elderly care. *Artificial Intelligence Review*, 53(7):4863–4901, 2020. doi: 10.1007/s10462-020-09809-6.

[12] A. McLaughlin, M. Gandy, J. Allaire, and L. Whitlock. Putting fun into video games for older adults. *Ergonomics in Design*, 20(2):13–22, 2012. doi: 10.1177/1064804611435654.

[13] Victoria Meza-Kubo and Alberto L. Morán. Ucsa: a design framework for usable cognitive systems for the worried-well. *Personal and Ubiquitous Computing*, 17(6):1135–1145, Aug 2013. doi: 10.1007/s00779-012-0554-x.

[14] R. Nouchi, Y. Taki, H. Takeuchi, H. Hashizume, Y. Akitsuki, Y. Shigemune, A. Sekiguchi, Y. Kotozaki, T. Tsukiura, Y. Yomogida, and R. Kawashima. Brain training game improves executive functions and processing speed in the elderly: a randomized controlled trial. *PloS one*, 7(1), 2012. doi: 10.1371/journal.pone.0029676.

[15] Rui Nouchi, Yasuyuki Taki, Hikaru Takeuchi, Hiroshi Hashizume, Yuko Akitsuki, Yayoi Shigemune, Atsushi Sekiguchi, Yuka Kotozaki, Takashi Tsukiura, Yukihito Yomogida, and Ryuta Kawashima. Brain training game improves executive functions and processing speed in the elderly: a randomized controlled trial. *PloS one*, 7(1), 2012. doi: 10.1371/journal.pone.0029676.

[16] Denise C. Park, Jennifer Lodi-Smith, Linda Drew, Sara Haber, Andrew Hebrank, Gérard N. Bischof, and Whitley Aamodt. The Impact of Sustained Engagement on Cognitive Function in Older Adults: The Synapse Project. *Psychological science*, 25(1):103–112, 2014. doi: 10.1177/0956797613499592.

[17] Shadi Tahmassebi. Digital game design for elderly people. Master's thesis, Faculty of Technology and Society, Department of Computer Science and Media Technology, 2018.

[18] Joe Verghese, Richard B Lipton, Mindy J Katz, Charles B Hall, Carol A Derby, Gail Kuslansky, Anne F Ambrose, Martin Sliwinski, and Herman Buschke. Leisure activities and the risk of dementia in the elderly. *The New England journal of medicine*, 348(25):2508–2516, 2003. doi: 10.1056/NEJMoa022252.

Evaluation of React Hooks Against Signal-Based Approaches

Leevi Pulkkinen
leevi.a.pulkkinen@aalto.fi

Tutor: Juho Vepsäläinen

Abstract

In the dynamic landscape of front-end web development the creation of new and more efficient ways of building applications remains as a constant interest of the community. For a long time React hooks have been the most used way of managing application state. However, recently the use of signal based methods has been on the rise. In this paper the advantages of signal based methods over React hooks are examined. Furthermore, this paper discusses whether signals could be the new standard way of managing the state of web applications. It was concluded that signals offer several advantages, such as fine grained UI updates, increased potential for performance and improved flexibility for state management.

KEYWORDS: React, hooks, signals, Solidjs, Preact, Qwik

1 Introduction

Recently, the concept of signals has been a hot topic as a state management solution in web applications. Signals are reactive primitives that can be used to manage application state [1]. Signals emit a notification when they change and they can be observed by subscribers [8]. This allows subscribers of a signal to automatically perform re-calculations and

updates when the signal value changes.

React [13] is currently the most popular front-end development library [4] and it uses hooks instead of signals for state management. Even though React is widely used, it has been criticized for poor performance [9]. Many signal based solutions have been proposed as a replacement for React, such as Solid [3], Preact [1], Qwik [2] and many more. In addition, the standardisation of signal in JavaScript has been proposed [14].

In this paper React Hooks are compared to signal based solutions, with a focus on scaling state management, developer experience, and performance. In addition the possibility of signals becoming the standard way of managing state in web applications is discussed. These observations are conducted through the following research questions:

RQ1. What are the advantages of signal based solutions over React hooks?

RQ2. Could signal based solutions be the new standard for managing state in web applications?

The structure of this paper is as follows. Section 2 presents the essential concepts of reactive programming and how it relates to signals. Section 3 introduces the basics of React, Solid, Preact and Qwik. In section 4 the strengths and weaknesses of these frameworks are discussed. Section 5 provides final remarks.

2 Reactive programming explained

Reactive Programming (RP) is a way of programming that is based on continuously changing variables and propagation of changes [7]. When changes in state are detected, all dependent computations are recalculated automatically by the RP language used [7]. This allows developers to program in a declarative way, focusing on what the application should do, instead of when or how to do it [7].

Consider the following example of reactive programming:

```
1 var1 = 1
2 var2 = 2
3 var3 = var1 + var2
```

Listing 1. Example of reactive programming

In traditional sequential programming the value of `var3` would be always equal to 3 if not manually modified elsewhere in the code, even if the values of `var1` and `var2` would change. In RP, however, the value of `var3` would always be automatically recomputed if the value of either `var1` or `var2` changes. [7]

Users Interfaces (UI) are highly reactive due to the need of reacting to external event such as mouse clicks and button presses [7]. Due to this UIs are difficult to program using traditional sequential programming techniques. This is because it is impossible to predict the order of external events and changes in application state need to be tracked manually. This approach is highly prone to bugs and requires large amounts of effort from developers. Reactive programming aims to solve these problems by automatically propagating changes and allowing the developer to write applications in a declarative way. [7]

Many JavaScript libraries use the concepts of reactive programming for managing application state. Signals are similar to the example listing 1 due to the fact that signals can be derived from other signals and they can be automatically updated when changes occur.

3 Introduction to JavaScript front-end libraries

React [13], SolidJs [3], Preact [1] and Qwik [2] are all JavaScript libraries used for creating front-end web applications. All of these libraries model parts of the User Interface (UI) using components. Components are small and reusable, each of them modeling a part of the UI, such as a button or a search bar. Components are implemented as JavaScript functions that return JSX, a JavaScript syntax extension that allows writing markup similar to HTML [13]. Components can be declared inside other components, forming a tree structure for the UI. This component tree is displayed to the user through the Document Object Model (DOM).

3.1 React

In React, component specific memory is called state. State can be used to remember how the user has interacted with the component. State can be implemented using React hooks, of which the `useState` hook is the most simple.

```
1 const [amount, setAmount] = useState(0)
```

Listing 2. useState hook in React

The `useState` hook takes an initial state as input and returns a state variable (`amount`) and a setter function (`setAmount`). When the component is rendered for the first time, the state variable is equal to the default value provided. The state value can be changed by calling the setter function, and this causes component to re-render. [13]

```
1  function Counter() {
2    const [count, setCount] = useState(0)
3
4    // All code inside component
5    // is re-run when count is changed
6    console.log(count)
7
8    return (
9      <button onClick={() => setCount(count + 1)}>
10       Increment {count}
11     </button>
12   )
13 }
```

Listing 3. Example component in React

React utilizes a virtual DOM (VDOM), which is a representative copy of the DOM saved in application memory [16]. To goal of the VDOM is to improve rendering performance by avoiding unnecessary and inefficient updates being made to the DOM. A snapshot of the VDOM is created before each state change. Changes are then applied to the VDOM and this new state is compared to the snapshot using a reconciliation diffing algorithm. Finally after the comparison the changes are made to the DOM and shown to the user. [16]

3.2 SolidJs

Solid is a JavaScript UI library that focuses on developer experience and high performance through fine grained reactivity [15]. Fine grained reactivity means that only the parts of the page that are changing are updated, instead of updating the whole page or component [15]. Code inside components is only run once on initialization, which is possible due to only

updating parts of the DOM that need updating [15]. In contrast to React, Solid does not use a virtual DOM.

In Solid, application state and data is managed using reactive signals [15]. Similarly to React's `useState` hook, signals return a setter function that can be used to modify state. However, instead of returning a state variable containing the value of the signal, a getter function is returned. This getter function can be called to gain access to the current value of the signal.

```
// create a new signal
const [amount, setAmount] = createSignal(0)

// access initial signal value
console.log(amount())
```

Listing 4. Creating a signal in Solid

Changes in signal values are monitored by subscribers [15]. Subscribers can be used for updating the UI or running side effects [15]. Subscribers can be created by calling the `createEffect` functions, or by calling the getter function of a signal in the return statement of the component [15].

```
function Counter() {
  const [count, setCount] = createSignal(0)
  const increment = () => setCount(count() + 1)

  // not tracked - only runs once during initialization.
  console.log("Count:", count())

  createEffect(() => {
    // will update whenever `count()` changes.
    console.log(count())
  })

  return (
    <div>
      <span>Count: {count()}</span>{" "}
      {/* will update whenever `count()` changes. */}
      <button type="button" onClick={increment}>
```

```
18          Increment
19        </button>
20      </div>
21    )
22  }
```

Listing 5. Example component in Solid [15])

3.3 Preact

Preact is described as an alternative to React, with a similar API, smaller size and higher performance [1]. Due to the similar API, Preact supports hooks such as `useState`, `useEffect` and `useReducer`. Additionally, Preact supports the use of signals.

In Preact, a signal can be created by calling the `signal()` function **outside** of a component [1]. To model local component state, the `useSignal` hook can be used, which creates a signal on the components initial render. Preact signal is an object, and its value can be accessed through the `.value` property [1]. A signal value can be changed by directly changing the value held in the `.value field`.

```
1 // create signal
2 const amount = signal(0)
3
4 // access signal value
5 console.log(amount.value)
6
7 // modify signal value
8 amount.value = 1
```

Listing 6. Creating a signal in Preact

When a signal's value changes, any components that use the .value property of the signal are re-rendered [1]. This limits the amount of re-renders needed if the signal is passed through many components that do not use its value. Preact utilizes a more lightweight version of the virtual DOM than React, and it can be bypassed in some cases to increase performance. Therefore, if a signal is passed to JSX without accessing its value, it can be updated in the DOM directly, without re-rendering the component [1].

3.4 Qwik

Qwik is a JavaScript library that promises instant loading for applications regardless of size [2]. Qwik aims to do this by creating resumable, lazy loaded components instead of using hydration [2]. Hydration is the process of making a static page interactive, by downloading and running JavaScript associated with the rendered components [2]. Instead of hydration, Qwik introduces the concept of "resumability". In Qwik, event handlers and application state are serialized into the HMTL returned from the server, and required JavaScript is only downloaded when it is needed [2].

Qwik models reactive state using signals [2]. Qwik signals are created inside components by calling the `useSignal()` function. Similarly to Preact, Qwik signals are objects that have a `.value` field, which holds the value of a signal. Additionally changing a signals `.value` field automatically updates any components using the value of the signal.

```
1 export default component$(() => {
2     const count = useSignal(0);
3
4     return (
5         <button onClick$={() => count.value++}>
6           Increment {count.value}
7         </button>
8     );
9 });
```

Listing 7. Example component using Qwik [2]

Additionally Qwik supports the use of a `useStore()` hook [2]. `useStore()` works similarly to normal signals, but allows the use of deeply reactive objects, that essentially consist of multiple signals.

Qwik uses a combination of VDOM diffing and direct DOM updates [2], aiming to combine the best features of both approaches. Qwik uses the virtual DOM when structural changes to the DOM are needed, but otherwise aims to update the DOM directly [2] .

3.5 Summary

Overall, these libraries have a lot in common, but they also contain some key differences. All of these libraries create UIs using nested components and JSX. Hooks can be used in a similar way in both React and Preact and signals are supported by Solid, Preact and Qwik. Solid is the only library to not use a virtual DOM to any extent, and Qwik introduces resumability instead of hydration.

	React	SolidJs	Preact	Qwik
Hooks	Yes	No	Yes	No
Signals	No	Yes	Yes	Yes
VDOM	Yes	No	Yes *	Yes**
JSX	Yes	Yes	Yes	Yes
Components	Yes	Yes	Yes	Yes
Hydration	Yes	Yes	Yes	No

Table 1. *A more lightweight VDOM than in React
**VDOM used only for structural updates

4 Comparison of signal-based libraries

This section discusses the scalability of React hooks and signals in large applications. Additionally the differences in developer experience between these frameworks are discussed. Finally it is explored how these solutions differ in terms of performance.

4.1 Scaling state management

In React, state often needs to be shared between components [13]. This can be achieved by storing state in parent components and passing it down to child components using props, which is called lifting state up [13]. However lifting state up can lead to state being passed through many components that do not depend on it [13]. This is called prop drilling.

In the React documentation [13], usage of the `useContext` hook is suggested for passing application state deep into the component tree. Components can be wrapped into a context provider, that allows all wrapped components and their children to access provided state, without it being passed as a prop to each component. However, usage of the React con-

text API can possibly lead to reduced performance, since each component that uses the state provided in the context is re-rendered when the state changes [12].

In React, it is common to use third party state management solutions such as Redux [6] or MobX [5]. Redux stores application state in a single store [6], which allows React components to access the state from anywhere in the component tree, without prop drilling. MobX is a signal based approach, that allows making data observable and updates to the UI are made automatically when this data is changed [5].

When scaling application size, Solid, Preact and Qwik can suffer from prop drilling, similarly to React. Because of this, they support the use of context, which allows state to be accessed by components, without it being provided with props [1, 2, 3]. Additionally, Solid and Qwik support the use of stores, that allow storing multiple signals in a centralized place [3, 2]. Due to stores consisting of signals, they can be an efficient way of managing global or complex state. Furthermore, when a property of a state object is changed, only components that use that specific property are modified.

4.2 Developer experience

As demonstrated by the examples 3, 5, and 7, the syntax of using signals and hooks can be quite similar. However, the exact usage of signals differs between implementations. In spite of this it could be argued that the learning curve for each of these libraries is similar, since all of them use the component architecture and JSX for building the UI.

In React hooks must be created in the top level of components [13]. Thus, hooks cannot be created inside conditional statements or loops, after conditional return statements or in event handlers [13]. This can be confusing for new developers trying to learn React. In Solid and Preact, signals can be defined outside of components [15, 1]. This makes using signals more flexible, since they can be imported or exported from anywhere in the codebase.

All of React, Solid, Preact and Qwik offer official documentation of good quality. However, due to React being overwhelmingly more commonly used [4], it naturally has more community support in the form of blogs, guides and answers to common problems.

4.3 Performance

React's use of the virtual DOM has been criticized as being inefficient, notably by Rich Harris, the creator of Svelte, who said "Virtual DOM is pure overhead" [9]. Harris argues that VDOM diffing calculations, inefficient component re-rendering and defaulting to doing unnecessary work will make the application slower in the long term, even if there is no clear bottlenecks [9]. However, Harris also acknowledges that the existence of the VDOM allows developers to handle state in a manageable way, while offering performance that is often "good enough" [9].

Despite the crititisim of React's VDOM, Preact and Qwik also use a virtual DOM. However due to the usage of signals, these libraries are less reliant on the VDOM and can sometimes update the DOM directly [1, 2]. This reduces the overhead created by the VDOM and improves performance.

In React it easy to build an application that has sub-optimal performance. This is evident due to the large number of guides on how to optimize your React code, such as [11] and [10]. Signal based methods aim to be perfomant by default [1], thus reducing the need for optimization in later phases of development.

5 Conclusion

In this paper the usage of React hooks was compared to signal based approaches in the form of Solid, Preact and Qwik. Due to this comparison it can be concluded that the usage of signals possesses several advantages over React hooks, such as fine grained UI updates, increased potential for performance, and improved flexibility for state management. Furthermore, decreased reliance on the virtual DOM and the concept of resumability offered by Qwik make these libraries interesting alternatives to React.

In the world of front-end web development, the number of libraries to choose from is large and new ones are created frequently. Many of these libraries already support signals in some form. Despite of this React continues to be the most commonly used library and it is difficult see its usage rapidly declining in the near future. This is due to the large community of developers and huge amounts of production code using React. In addition, even though usage of signals offers many benefits, React is often

"good enough" for most projects. Therefore it is hard to see signals being the new standard for managing state in web development, at least in the near future. However this could change due to the standardization of signals being proposed as a part of JavaScript [14], which would allow libraries to use signals in a standardized way.

References

[1] Preact - fast 3kb alternative to react with the same modern api. https://preactjs.com/. Accessed: 2024-01-31.

[2] Qwik - the html-first framework. https://qwik.dev/. Accessed: 2024-01-31.

[3] Solidjs - a declarative, efficient, and flexible javascript library for building user interfaces. https://www.solidjs.com/. Accessed: 2024-01-31.

[4] Stack overflow developer survey 2023. https://survey.stackoverflow.co/2023/. Accessed: 2024-02-01.

[5] Mobx - simple, scalable state management. https://mobx.js.org/README.html, 2024. Accessed: 2024-02-28.

[6] Redux - a predictable state container for js apps. https://redux.js.org/, 2024. Accessed: 2024-02-28.

[7] Engineer Bainomugisha, Andoni Lombide Carreton, Tom van Cutsem, Stijn Mostinckx, and Wolfgang de Meuter. A survey on reactive programming. *ACM Computing Surveys (CSUR)*, 45(4):1–34, 2013. doi: 10.1145/2501654.2501666.

[8] Ryan Carniato. A hands-on introduction to fine-grained reactivity. https://dev.to/ryansolid/a-hands-on-introduction-to-fine-grained-reactivity-3ndf, 2021. Accessed: 2024-04-05.

[9] Rich Harris. Virtual dom is pure overhead. https://svelte.dev/blog/virtual-dom-is-pure-overhead, 2018. Accessed: 2024-03-27.

[10] Ibadehin Mojeed. Optimizing performance in a react app. https://blog.logrocket.com/optimizing-performance-react-app/, 2023. Accessed: 2024-04-05.

[11] Temitope Oyedele. React optimization techniques to help you write more performant code. https://www.freecodecamp.org/news/react-performance-optimization-techniques/, 2024. Accessed: 2024-04-05.

[12] André Rabold. Scaling react state management part 2. https://medium.com/@andre.rabold/scaling-react-state-management-part-2-5a9bcda83b55/, 2021. Accessed: 2024-03-01.

[13] React Team. React - a javascript library for building user interfaces. https://react.dev/, 2024. Accessed: 2024-01-30.

[14] Daniel Ehrenberg Rob Eisenberg. Javascript signals standard proposal. https://github.com/proposal-signals/proposal-signals, 2024. Accessed: 2024-04-04.

[15] SolidJS Team. Solidjs documentation. https://docs.solidjs.com/, 2024. Accessed: 2024-02-21.

[16] Prashant Yadav. The power of react's virtual dom: A comprehensive explanation. https://www.syncfusion.com/blogs/post/react-virtual-dom, 2023. Accessed: 2024-03-25.

Measuring cybersecurity risk of industrial control / OT systems

Lija Bista
lija.bista@aalto.fi

Tutor: Mikko Kiviharju

Abstract

The paper is aimed at revealing the cyber security risk that emerges due to the application of industrial control and operational technology systems (OT), but considers the potential of defining a process based on IEC 62443 Foundational Requirements. These requirements highlight the need for a variety of strategies to successfully lessen cyber threats and increase the safety of vital infrastructure assets. By the end of the research, the evaluation of secondary data and function definition highlights the critical importance of industrial cybersecurity controls. The study focuses on how accurate cybersecurity measurements can only be ensured by using high-level, standardized procedures and processes. It also examines how these measures stack up against an ever-evolving set of industry best practices. Thus, critical infrastructure assets are made safe from different cyber threats.

KEYWORDS: Industrial Control System (ICS), Operational Technology (OT), Industrial Automation and Control Systems (IACS)

1 Introduction

When various control elements are integrated, they work together to achieve industrial objectives, such as production targets and energy flow within industrial control systems [11]. Operational technology (OT) refers to the hardware and software employed within industrial environments to monitor, manage, and regulate devices, processes, and infrastructure, exerting control over the tangible realm [2]. In OT, industrial control systems (ICS) serve as a vital element, incorporating diverse devices, controls, systems, and networks to oversee a range of industrial processes [5].

Protecting industrial control/OT systems from cyberthreats necessitates comprehensive awareness of prevalent cybersecurity risks, such as malware attacks, supply chain vulnerabilities, and human errors, which can compromise system integrity and sensitive data [12]. Addressing these challenges requires proactive measures and robust cybersecurity strategies to mitigate the potential impact of disruptions and breaches on ICS/OT environments.

This study examines the elements impacting the effectiveness of security protocols in lowering cyber risks by analyzing the impact of IEC 62443 Foundational Requirements on the deployment and prioritization of security measures in industrial control/OT systems. Furthermore, the aim is to assess the evidence backing the effectiveness of IEC 62443 Foundational Requirements in evaluating cybersecurity controls within industrial control/OT systems, and their impact on the development of organizational security postures.

The paper is structured as follows. The IEC 62443 Foundational Requirements are presented in Section 2. The study objectives are explained in Section 3, and supporting data for the IEC 62443 Foundational Requirements' effectiveness is presented in Section 4. The Impact of IEC 62443 Foundational Requirements on Security Measures is covered in Section 5. In conclusion, Section 6 provides some final thoughts.

2 IEC 62443 Foundational Requirements

Attack vectors are used by threats to carry out attacks against an industrial control system (ICS). These threats can be directed towards the safety instrumented system, aiming to damage when safety functions are activated, or the control command system, which is in charge of oversee-

ing the physical system [3]. Malware is a serious danger to industrial control systems (ICS) since it is specifically designed to interfere with operations or harm machinery. Cyberattacks on ICS can result in physical damage to equipment, interruption of vital infrastructure, and even fatalities [6].

The IEC 62443 refers to the standards that constitute a framework aimed at ensuring the secure evolution of Industrial Automation and Control Systems (IACS), furnishing detailed and organized recommendations for cybersecurity. IEC 62443 is a multifaceted standard that applies to all industrial cybersecurity stakeholders. Its implementation process starts with a risk analysis and includes recommendations for a cyber risk reduction factor calculation approach as well as requirements for creating, implementing, maintaining, and enhancing a security program for an IACS [4]. An Industrial Automation and Control System (IACS) is guaranteed to have the necessary security and safety safeguards if it complies with the seven criteria specified by IEC 62443 for each security level. Let's delve into each of these foundational requirements for industrial automation security in detail as per IEC 62443 [1]:

Foundational Requirement 1 (FR 1) – Identification and Authentication Control

A strong and well-designed identification and authentication control mechanisms is necessary for the OT and industrial control subsystems, because these critical domains are a potential target for a number of cyber threats [10]. FR1 is a wide scope of sub-requirements (SRs) that focuses on a secure framework which authenticate user, device management, and access. With this control, the system will ensure authorized person of accessing only separated devices or information. This would be achieved through the development of techniques such as user authentication and entity authorization as a means of protecting the system from common hacking attacks. Through the adoption of access control mechanism, the system thwarts intruders or unauthorized individuals/entities, affording necessary protection against data intrusion or malicious activities. [10] FR 1 in the context of identification and authentication controls of the OT systems that are used in industrial control and operational technology illustrates the crucial area which must be addressed. Such activities comprise user identity verification, software applications and devices authen-

tication and securing username and password, etc. [10] The other way is that the application of strong password authentication, Public Key Infrastructure (PKI) certificates and authenticated feedback systems makes up for strong security resilience. On top of that, controls like the wireless access management and the monitoring of wrong attempts are the factors that help reduce possible issues. The measures taken, like displaying notifications on system use and limitations towards network access through untrusted networks do safeguard transparency, accountability and they do also protect against unauthorized access or manipulation. In summary, compliance with FR 1 strengthens both the willingness to avoid cyberattacks and the competence to resist a myriad of cybersecurity-related risks and ensure protection of vital infrastructure assets. [10]

Foundational Requirement 2 (FR 2) – Use Control

An appropriate use control techniques must be utilised to decrease cyber security risks and to guarantee the integrity, confidentiality, and availability of critical infrastructure's assets. [10] In the case of FR 2, there are a number of generic technical requirements (SRs) which are meant to provide for a secure system environment, with effective resource management, security based on the user device and using the software application. [10]

Sub-Requirement 2.1 indicates that the rigorous user authentication and the prohibition of unauthorized actions and data mutation are both the essential measures in this control system. SR 2.2 specifies the surveillance of wireless activity that is observed by the implementation of access policies and encryption of communication channels. Sub-Requirement 2.3 acknowledges the significance of implementing measures such as mobile and portable device authentication and data encryption as the equivalent of precautions considering the security risks of unauthorized access. Sub-Requirement 2.4 is dedicated to the installation of security mechanisms for safe mobile code execution, which aims to prevent and handle the risks accordingly. [10] Among Sub-Requirements 2.5-2.7, are the session management controls which include session locking, remote termination, and concurrent session management in order to ensure no unauthorized access and avoid data breaches Sub-Requirement 2.8 requires the organization to capture auditable events to enable control monitoring and forensic study, whereas Sub-Requirement 2.9 stresses dealing with control capac-

ity in terms of storage. Requirement 2.10 addresses issue of quick response to close the audit gap and requirement 2.11 – deals with the problem of time stamps accuracy for forensic analysis. However, under Point 2.12, Sub-Requirement, controls are being set up to get rid of the issue of non-repudiation, thereby making each and every action accountable and hindering malicious activities. [10]

Foundational Requirement 3 (FR 3) – System Integrity

For the FR 3, sub-level requisites (SRs) are grouped in order to improve the crucial process of protecting system integrity through reliable controls, checkpoints and mechanisms. [10]

Sub-Requirement 3.1 is all about the integrity of communication channels in particular via an encryption and data validation so that the access to any unauthorized tampering or interception is blocked. Sub-Requirement 3.2, which is one of technical features, represents the countermeasure to malicious code attacks by suggesting steps like antivirus software and code verification practices. In the framework of the sub-requirement 3.3, we would examining the functionalities of the security system using testing methodologies. [10]This would help identify potential sophisticated vulnerabilities and remedies them. Sub requirement 3.4 facilitates, above all, the implementation of control tools to ensure the completeness and accuracy of software and information, including file monitoring and access control. Part 3.5 points out that input validation is necessary in order to neutralize the use of vulnerabilities for exploitation purposes. Sub-Requirement 3.6 underscores the necessity of a stable and predictable system's output that could be potentially used for undesirable behavior. Sub-Requirement 3.7 lays emphasis on agile error handling methodologies to classify faults and avert their proliferation. Addressing session integrity issue by encrypting the transmission and managing the Keys is described in Sub Requirement 3.8. Sub-Requirement 3.9 states that audit information should be kept secure from any external modification or access which should secure its reliability for forensic analyses and compliance. [10]

Foundational Requirement 4 (FR 4) – Data Confidentiality

Data confidentiality must be ensured in industrial control and OT systems for protection of sensitive data from unauthorized access, disclosure, which may lead to loss of data as well as reputational crisis. SR4 which is the collection of some sub-requirements are designed to make sure that confidentiality of data within system is maintained. [10]

Sub-Requirement 4.1 underlines the implementation of access controls and encryption to prevent data loss as a result of leaking or misuse (i.e., unauthorized disclosure or loss of information). As Sub Requirement 4.2 postulates, sensitive data protection throughout it's lifecycle with the help of secure storage mechanisms and data retention policy would be kept on standby. Cryptography techniques will be used to carry out encoding to ensure that the confidentiality of information is not compromised by unauthorized access. Eventually, compliance with FR 4 of the highest level of protection concerning the confidentiality of data assets between industrial control and OT systems, reduces cyber risks. [10]

Foundational Requirement 5 (FR 5) – Restricted Data Flow

Keeping up with the arrangement of data work is the prime concern of industrial control system to avoid unauthorized entry, losing, or manipulating of valuable data. The FR 5 consist of several requirements (SR) so that the data flow is within the system is limited. [10]

Sub-Requirement 5.1 requires to execute network segmentation controls including firewalls, VLAN, and so on to disrupt the connections and shape the data movement among different sections. Sub-Requirement 5.2 of the standard details perimeter defense of network zones by implementing state-full inspection firewalls and intrusion detection systems to stop unapproved access from one environment to the other. Sub-Requirement 5.3 entails mechanisms to confine or track emails and files exchange, thus to prevent data outflow. [10] In Sub-Requirement 5.4, partitioning of applications is advised to secure communication as well as prevent unexpected and unauthorized access through utilization of approaches like containerization and access controls. FR 5 compliance guarantees anonymity and network traffic control in the industrial control system and operational activities that help decrease the number of cyber risks and prevent national infrastructures from being harmed. [10]

Foundational Requirement 6 (FR 6) – Timely Response to Events

Adherence to FR 6 constitutes an opportunity for timely preventive, mitigating and restorative measures towards cyber threats. If appropriate action is taken on time after any events or intrusion then there will be less impact of cyber threats and no operation will be compromised, whereas the business continuity will be secured. [10]

Sub-Requirement 6.1 places the accent on the availability of audit logs as a mechanism for rapid detection and response to the security incidents with the industrial control systems through control measures including in the installation of centralized logging systems and real-time monitoring tools for analysis. The Sub-Requirement 6.2's goal is about adopting monitoring controls in continually and resolving security threats which includes the application of the intrusion detection system and anomaly detection tools. [10]

Foundational Requirement 7 (FR 7) – Resource Availability

Efficient provision of resources within control and operational technology infrastructures is a critical aspect in the undertaking of continuous operation of critical infrastructural assets with the aim of minimizing the impact resulting from cyber threats. [10]

Sub-Requirement 7.1 puts forth the necessity of countermeasures against Denial of Service (DoS) assaults to keep system accessibility. This is to be attained by network traffic filtering and intrusion prevention system among other controls. It is highlighted in Sub-Requirement 7.2 that success is possible only when effective controls on resource management such as resource monitoring and electronic scheduling are put in place to ensure availability of resources. [10] Sub-Requirement 7.3 sets a 24-hour mechanism to restore control system power backups, while Sub-Requirement 7.4 is responsible to deploy recovery mechanisms efficiently post disruption. If not only that, Sub-Requirement 7.5 points out the very important role of emergency power systems as well as Sub-Requirement 7.6 being responsible about how the network is configured and secured. Sub-Requirement 7.7 states that the general (minimum) work system (minimal operation) concept should be followed to decrease the operational risks with Sub-Requirement 7.8 advocating for the accuracy of inventory of control system components. [10]

3 Research Questions

This paper is centered around the following research questions.

1. How do IEC 62443 Foundational Requirements influence the prioritization and implementation of security measures within industrial control/OT systems, and what factors impact their effectiveness in mitigating cyber risks?

2. What evidence is there to support the effectiveness of IEC 62443 Foundational Requirements in assessing the cybersecurity controls within industrial control/OT systems?

4 Supporting Evidence for Effectiveness of IEC 62443 Foundational Requirements

Ting et al. [13] stresses the significance of IoT devices in manufacturing, highlighting the cybersecurity risks posed by patching challenges. To mitigate these risks, assets are identified and categorized based on patch status, leading to the establishment of a comprehensive patch management list. The presented patch management procedure facilitates supervised patching to meet IEC 62443-2-4 requirements, resulting in fewer cybersecurity threats and guaranteed output availability. Ultimately, the received IEC 62443-2-4 certification validates the efficiency of the same patch management solution, leaving a trail of error-free and dependable service delivery among other cost-cutting and productivity enhancement benefits. [13].

In order to perform a comprehensive risk assessment of an industrial facility, the evaluation given in [7] follows the requirements of standard IEC 62443, with a particular focus on FR1: Identification and Authentication Control (IAC). The authors mention what might be the most vulnerable points in the model and apply zone-based methodology taking into consideration the probability and consequences of the future threats. They emphasize the need to assess the impact analysis aspects like data sensitivity, mission criticality, and system purpose using organizational docs as the source of information. Countermeasures are recommended to reduce threats, mainly regarding intrazonal connections, for instance, authenti-

cation procedures and the ban on files extraction. The conclusion highlights the need for more research and development in the assessment of cybersecurity threats, also recommending features, like attack frequency and cost [7].

5 Influence of IEC 62443 Foundational Requirements on Security Measures

An effective framework for evaluating cybersecurity tool performance and alignment with industry benchmarks is provided by the validation approach described in [9]. It employs the sections of IEC 62443 standard, namely segment 4-1 and 4-2. The framework results in a clear picture of a given tool's performance as well as its compliance with the already set criteria. This is possible through the methodical mapping and assessment approach that has been further enhanced with heatmap visualization approaches. The practicality of the methodology is also supported by the inclusion of a case study in real life, particularly when the software is adapted or a vulnerability scan compared with a broader infrastructure. This structure, in industrial context, becomes an important tool by which enterprises ensure that their cybersecurity systems are updated and pertinent to new industry standards [9]. This makes the necessity of accepting similar validation approaches to promote trustworthiness and reliability of security tools evident, therefore cybersecurity tools can significantly contribute to strengthening critical infrastructure resilience.

The paper [8] provides a study of cyber security in industrial setups, especially the power plants for hydropower as critical infrastructure. In other words, the research utilizes an IEC 62443-2-1-based cyber security management system (CSMS) which has six integral elements: from the initial risk assessment to the maintenance of CSMS. The authors implemented the practical application of the CSMS concept with the theoretical case study that was based on the actual hydroelectric power plants (HPP), demonstrating the conversion of safety risk analysis into security risk analysis using the cyber-HAZOP technique. The article identifies the principal ICS vulnerabilities within HPPs and offers the following: updating security patches and implementing physical access control to name but a few. [8]

6 Conclusion

In a nutshell, the cybersecurity risks hazards evaluation in industrial control and operational technology (OT) systems has high significance in preventing the weakening up of the critical infrastructure systems by a swarm of cyber vulnerabilities. This study have considered the IEC 62443 Foundational Requirements' role in the installation and the provision of security solutions. Thus, this paper has exposed the thinking process of substituting each core control with the necessary sub-requirements in order to help better support the cyber security risk mitigation by increasing the cyber resilience of industrial control systems and overall OT ecosystems. The effectiveness of IEC 62443 standard as a security solution is emphasized through points like patch management, risk assessment, and auditing tools usage which harmonizes system performance. Moreover, the report highlights the necessity of the frameworks that conform to IEC 62443 regulations so that the security defenses can be relied on and they can always keep updating industry's best practices. The whole picture will be strengthened by improving the cybersecurity posture if organizations are going to take up the principles and guidelines outlines in the IEC 62443 standard, these will help in reducing cyber risks and securing the critical infrastructure assets in industrial control and operation technology.

References

[1] Tareq Ahram and Christianne Falcao, editors. *Human-Centered Design and User Experience*. AHFE 2023 Hawaii Edition, Honolulu, Hawaii, USA, 2023. doi: 10.54941/ahfe1004214.

[2] Cisco. How do ot and it differ? Accessed on 29 January 2024, https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-ot-vs-it.html: :text=Operational

[3] Jean-Marie Flaus. *The Approach Proposed by Standard 62443*. ISTE :, London, England ; Hoboken, New Jersey :, 1st edition edition, 2019.

[4] Jean-Marie Flaus. *Methods and Tools to Secure ICS*. 2019. doi: 10.1002/9781119644538.ch10.

[5] Fortinet. What is ot security? Accessed on 29 January 2024, https://www.fortinet.com/solutions/industries/scada-industrial-control-systems/what-is-ot-security: :text=Industrial

[6] Andrew Ginter, editor. *Secure Operations Technology*. Abterra Technologies Inc. Calgary, Calgary,Alberta, Canada, 2018.

[7] Hicham Lalaoui Hassani, Ayoub Bahnasse, Eric Martin, Christian Roland, Omar Bouattane, and Mohammed El Mehdi Diouri. Vulnerability and security risk assessment in a iiot environment in compliance with standard iec 62443. *Procedia Computer Science*, 191:33–40, 2021. The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 16th International Conference on Future Networks and Communications (FNC), The 11th International Conference on Sustainable Energy Information Technology.

[8] Jessica Heluany and Ricardo Galvão. Iec 62443 standard for hydro power plants. *Energies*, 16:1452, 02 2023.

[9] Anas Husseis and Jose Luis Flores. A practical framework for evaluating cybersecurity tools leveraging the iec62443 standard. 10 2023.

[10] International Electrotechnical Commission. *IEC 62443-3-3:2013 Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels*. IEC, Geneva, Switzerland, 2013.

[11] NIST. Industrial control system (ics). Accessed on 29 January 2024,https://csrc.nist.gov/glossary/term/industrial$_{c}ontrol_{s}ystem$: : $text = An$

[12] Michael Artemio Go Rebultan. Introduction to ics/ot systems and their role in critical infrastructure, 12 June 2023. Accessed on 29 January 2024, https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2023/common-cybersecurity-risks-to-ics-ot-systems: :text=Some

[13] Vipin Ting, Hsiao-Yu Chou, and Jung-Hsing Wang. Securing manufacturing through patch management for iot devices. In *2023 IEEE 3rd International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, pages 479–482, 2023. doi: 10.1109/ICEIB57887.2023.10170074.

Microservices - Navigating Benefits and Challenges in Modern Software Architecture

Linh Ngo
linh.l.ngo@aalto.fi

Tutor: Antti Ylä-Jääski

Abstract

The forefront of modern practices in designing and deploying scalable and modular software systems is exemplified by the rise of microservices architecture. This paper thoroughly presents the benefits and drawbacks associated with the adoption of microservices architecture, while also making comparisons with the traditional monolithic architecture. Through a comprehensive analysis, it demonstrates the trade-offs and considerations involving in transiting to microservices, which offers valuabe insights for organizations, architects, and developers to navigate the landscape of software design.

KEYWORDS: Microservices Architecture, Monolithic Architecture

1 Introduction

At the core of software design, software architecture serves as a fundamental structure that defines technical and operational requirements for quality attributes that the system has to meet, hence making substantial influence over the robustness, scalability, and overall efficacy of software systems. Among software architectural frameworks, Monolithic Architecture (MA) and Microservices Architecture (MSA) are two popular

paradigms that represent contrasting approaches for structuring applications. While MA consolidates all components into a unified application, MSA fragments the application into independently deployable services. As the demand for scalable and maintainable software system increase, it becomes important to grasp the trade-offs between different architectural paradigms. Using knowledge published in recent scholarly literature, this paper aims to provide a comprehensive review of MSA by evaluating its present advantages and challenges, while also draws a comparison with MA. This paper is organized into five sections. Section 2 introduces essential background information about MA and MSA . Section 3 critically analyzes the advantages, disadvantages, challenges, and when to use associated with microservices. Section 4 encapsulates the paper with a conclusion. The references used are presented in the last section.

2 Background

2.1 Monolithic Architecture

Monolithic Architecture (MA) is a traditional method in software development, in which multiple components are encapsulated into a single program from a single platform [9]. When the application is small in scale, MA offers benefits such as streamlined process in development, testing, debug, and deployment. However, the drawbacks of MA become pronounced as software systems grow in size and functional complexity, rendering it unsuitable for the rapid delivery pace and high requirement volatility prevalent in recent years. The challenges stem from high coupling characteristics in large MA applications, resulting in scalability issues and elevated deployment costs. Additionally, traditional monolithic architectures face hurdles such as expanding difficulties, increased coupling, and deployment complexities due to their unified and tightly-integrated structure, posing obstacles like "dependency hell" and technology lock-in for developers [5].

2.2 Microservices

2.2.1 Characteristics

Recently, microservices has emerged as an alternative architectural paradigm to overcome the challenges of MA. This is achieved by decomposing mono-

liths into multiple dependent deployable services [2]. MSA operates on distributed model, with each service models a specific business domain, manages to run in their own container with its own private database, and can be deployed, tested, and maintained separately [7]. Therefore, microservices enhance modularity in software systems by emphasizing loose coupling and high cohesion [5]. This decentralized nature of MSA facilitates superior scalability and deployment, empowering services to scale autonomously, isolate faults, deploy flexibly, and adapt to specific technological needs.

As microservices can be viewed as breaking down monolithic systems into individual services, the communication and coordination between services is needed. In monolithic applications, components are invoked via functional calls, while microservice-based architectures interact using an Inter-Process Communication (IPC) mechanism. It is vital to take into account services interaction, for example, whether the interaction is synchronous or asynchronous when choosing IPC mechanism. In synchronous interactions, the client expects a timely response from the service and may even halt its operations while awaiting it. Synchronous request/response-based IPC mechanisms involve clients sending requests to services, which process them and send back responses to server. The most popular protocols for synchronous include HTTP-based REST Protocols. In contrast, in asynchronous interactions, clients send requests or messages to services without expecting to receive replies immediately. Responses are handled separately at some point in the future, allowing clients to continue their tasks without blocking and services to concurrently handle multiple requests without blocking. The most popular mechanism for asynchronous communication protocols is message-based communication protocols such as the Advanced Message Queuing Protocol (AMQP) [3].

2.2.2 Microservices Architectural Topologies

Although there exists various approaches to implement MSA, three primary topologies are particularly prominent and widely adopted: the API REST-based topology, application REST-based topology, and centralized messaging topology [10].

The API REST-based topology comprises microservices, which are fine-grained service components accessible through a REST-based interface facilitated by a separately deployed web-based API layer [10]. This topology is suitable for websites offering small, self-contained services accessible

through an application programming interface (API).

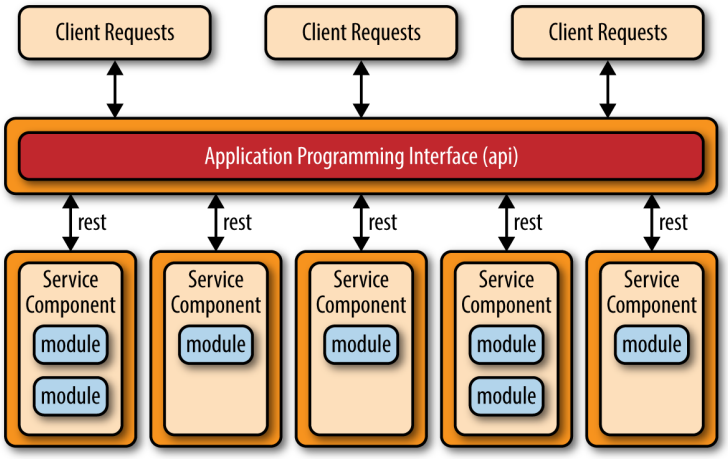


Figure 1. API REST-based topology [10]

The application REST-based topology differs from the API REST-based approach in that client requests come through traditional web-based or fat-client business application screens, rather than a simple API layer. Here, the user interface layer is a separate web application that accesses service components remotely via REST-based interfaces. These service components tend to be larger and less granular, representing a smaller part of the overall business application. This topology is common in small to medium-sized business applications with lower complexity.

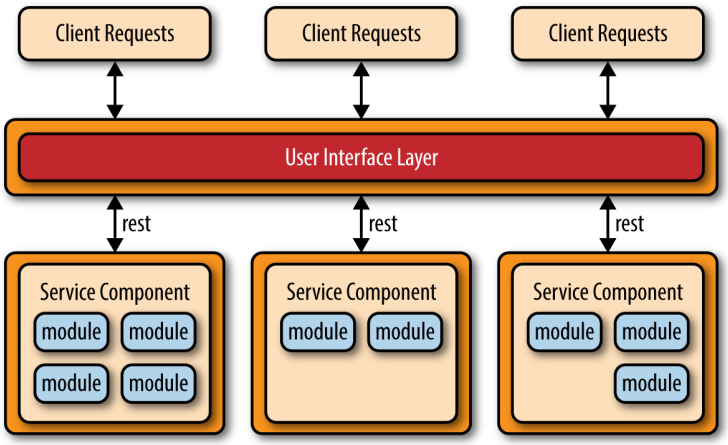


Figure 2. Application REST-based topology [10]

The centralized messaging topology is similar to the previous application REST-based topology but uses a lightweight centralized message broker (such as ActiveMQ, HornetQ, ...) to access remote service components instead of REST protocols.

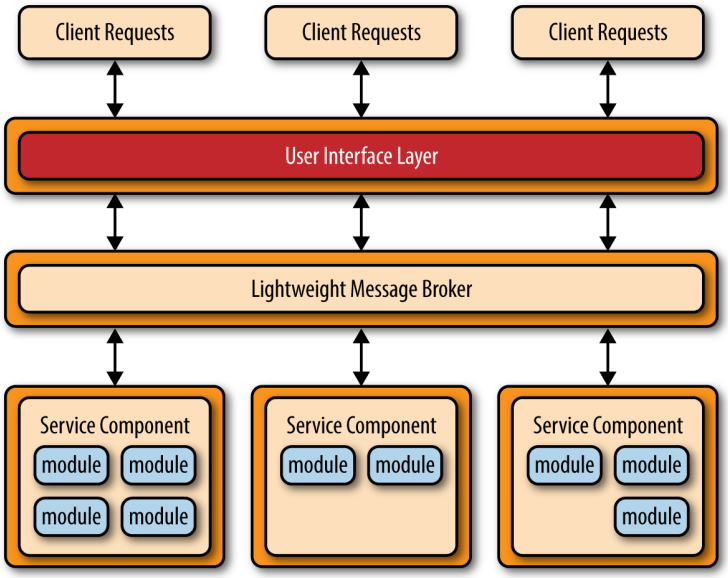


Figure 3. Centralized message topology [10]

3 Analysis

3.1 Advantages associated with transitioning from monoliths to microservices

In order to make well-informed decisions about software architecture and software development strategies, it is crucial to delve into the advantages of microservices across technical, cost, and process-related dimensions.

3.1.1 Technical Advantages

This paper delves into the fundamental benefits of MSA, specifically highlighting its strengths in scalability, containerisation, fault-tolerance, and maintainability.

Scalability

Scalability, which is defined as the capability of system to handle increasing workloads by adding more resources to the system, is one of the primary motivators for adopting MSA. Resources can be added vertically by increasing the capabilities of existing hardware or software components and horizontally by adding more instances of those components across multiple machines or nodes. Although vertical scaling is a more direct approach, it is restricted by the maximum capabilities of available hardware and incurs an increased cost when the hardware configuration exceeds a certain threshold. MSA applications can be scaled vertically by deploying each service instance according to their load, allowing each service to

operate at its own capacity. In contrast, horizontal scaling, while more complicated due to its impact on application architecture, often achieves scalability levels surpassing those of vertical scaling methods. Horizontal scaling is also more prevalent in microservices applications compared to monolithic applications, though an MA can also scale by deploying multiple instances behind a load balancer. Moreover, microservices enable horizontal scalability for applications, not just in technical aspects but also in the organization's formation of developer teams, promoting smaller and more agile team structures [3].

Containerisation

Recently, containerization has emerged as the preferred method for packaging and delivering microservices. The container is a lightweight virtualization technology that encapsulates both application and its runtime dependencies into a package, enabling it to run seamlessly across various environments, from physical machine to virtual machines [7]. Due to the technology heterogeneity of MSA, containers are well-suited for sandboxing services in a virtualized environment, which facilitates resource isolation and enhancing security systems [7].

Fault-tolerance

Fault-tolerance refers to the ability of software systems to continue operating despite failures. In monolithic applications, the failure of one point can cause a system-wide failure due to tight coupling of components. In contrast, MSA achieve fault tolerance through their decentralized and modular nature, which means that a downfall of one service does not affect the availability of other services and user requests can still be fulfilled by other functioning services [3].

Maintainability

Maintainability is one of the main motivators for using microservices. It is argued that micro-services simplify the complexity of a monolithic application by breaking it down into a set of small individual services [1]. Moreover, the autonomy granted to each service in MSA allows developers to make modifications and conduct testing autonomously, free from dependencies on other services. This decentralized approach not only enhances the agility of the development process, but also minimizes the potential disruptions that changes in one service might inflict on others within the system.

3.1.2 Cost Advantages

Various companies are embracing microservices and rearchitecting their existing systems; thus, the discussion of the economic impact of such a change is necessary.

Infrastructure Costs

According to the experiment in [14], infrastructure costs of deploying microservices application, which is determined by the cost per hour, is usually lower in comparison with monolithic applications, primarily due to the ability to scale up or down only specific services instead of scaling the whole monolith when needed. The ability to adapt to demand gives companies more control over operational costs. Moreover, this characteristic also enables businesses utilizing cloud computing to save in IT infrastructure costs by benefit from the pay-per-use and on-demand cloud model [13].

3.1.3 Process-related Advantages

Reusability

Reusability, the ability to support various products and introduce new products efficiently, lies in the core of MSA. This allows the leverage of shared components between all products, such as login and authentication, or launch new products with only a thin layer of capabilities and product management [11].

Release Frequency and Agility

Release frequency and agility are seamlessly integrated in the MSA due to its autonomous nature, allowing companies to swiftly update the deployment process and adapt in tune with changing business requirements without causing system-wide issues. Continuous delivery is an important aspect of MSA; its absence hinders the benefits of MSA [1]. The nature of autonomous microservices allows the team to make their own localized decisions on their software and reduces the need for interteam communication, which is a challenge in large-scale software development [3]. Furthermore, microservices enable horizontal scalability for applications not only in technical aspects, but also in the organization's formation of development teams, which promotes smaller and more agile team structures [3].

3.2 Challenges

Due to their inherent complexity, microservice-based applications often face numerous challenges throughout their lifecycle. These challenges encompass various aspects and span throughout the design, development, and operational stage.

3.2.1 Design Stage

Architecture

During the design phase, MSA poses a challenge in the allocation of service sizes [12]. Although microservices are typically structured around well-bounded business domains, the lack of clear domain boundaries can magnify the difficulties in the development and maintenance of microservices applications [8]. This ambiguity can potentially result in increased cost, higher probability of cross-service changes,and the emergence of overly coupled components within the MSA. Consequently, this increased complexity may undermine the advantages of microservices compared to the straightforward deployment of monolithic softwares.

API Versioning

Furthermore, given that microservices typically communicate through remote API calls, it is important to API versioning management to maintain retrocompatibility and facilitate seamless intercommunication among microservices.

3.2.2 Development Stage

Data Consistency

At the development phase, a significant pain of MSA lies in storage-related issues. Due to the distributed nature of data across various microservices' databases, ensuring data consistency over transactions is a critical challenge. The traditional approach of addressing data consistency that typically applies to centralized databases in many monolithic systems is not well suited for MSA. The two common methods for ensuring data consistency in microservices-based applications are the cloning-based method and the private database method [7]. In the cloning-based method, each service generates a replica of the original database, operates with its own private database, and transmits to other cloned databases to ensure data consistency. The drawback of this method is that it takes more resources to store data. In the private database method, each service has its own private database, which is often accomplished through horizontal slicing of central data storage based on distinct business domains. However, this

approach poses a challenge of maintaining data consistency across these separate databases. Therefore, data consistency in MSA is a real challenge that needed to be taken into account.

Testing

Testing is one of the important but challenging aspects of MSA, as the more components an MSA consists of, the greater the chance that failures occur. The challenges with MSA mirror those in distributed systems, such as inter-service communications, services coordination, and distributed transactions management [4]. On the one hand, the isolation of microservices significantly improves component testability compared to MA due to the ability to test them individually [5]. On the other hand, for large systems with numerous connections between components, integration testing aimed at testing the collaboration of a number of services can become very tricky [5].

Moreover, with a significant number of services that make up an application, performance testing for microservices is important. Such testing typically consists of the use of system testing and black-box methods to assess non-functional requirements like load, stress, and capacity. However, manually managing each test specification becomes increasingly challenging, particularly for MSA applications with a substantial set of services (e.g., 100) [4].

3.2.3 Operation Stage

Monitoring

The primary challenges during the operational stage of microservices center on how to effectively deploy and manage microservices [12]. This is largely due to the distributed and dynamic nature of MSA, where microservices can be added, removed, scaled, or migrated across hosts. Consequently, complexity may arise in coordinating and locating specific instances within the application, as well as maintaining proper isolation to prevent cascading failures from a single failure of a microservice instance. Furthermore, navigating through logs that are distributed across multiple locations can also be challenging when identifying the root causes of application issues.

Resources Consumption

In monolithic applications, communication occurs primarily through in-memory calls or function calls between various components of the application, alongside with shared memory access. On the contrary, distributed applications such as MSA applications typically communicate with each

other over networks. Due to the increased latency in network communication compared to in-memory mechanisms, MSA applications experience from degraded performance and speed loss in communication when compared to MA applications [5]. In the event of network issues, the inability to access service provider leads to the cascading failure of subsequent service consumers and potentially result in unavailability in service calls. [7].

Moreover, it is suggested that the impact of containers on software performance might not always be negligible [6]. In monolithic applications, the whole application is usually packaged within a single container, whereas each service in MSA is typically encapsulated within its own container. Container usage can potentially reduce performance and increase CPU utilization, with evidence showing that the number of CPU instructions required to process a client request is typically double that of monolithic systems [1].

4 Conclusion

This paper analyses the advantages and disadvantages of embracing MSA and provides a comparative analysis with MA, taking into considerations cost, process-related, and technical aspects. Moreover, this paper also addresses challenges encountered throughout the stages of design, development, and operation in MSA implementation. There are no silver bullet solutions to all the use cases; hence, it is important to balance between different criteria that align with the specific needs of the software lifecycle process when choosing the software architecture, including technical requirements, processes, and cost considerations.

References

[1] Florian Auer, Valentina Lenarduzzi, Michael Felderer, and Davide Taibi. From monolithic systems to microservices: An assessment framework. *Information and Software Technology*, 137:106600, 2021.

[2] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Migrating to cloud-native architectures using microservices: An experience report. 07 2015.

[3] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 10:20357–20374, 2022.

[4] André de Camargo, Ivan Salvadori, Ronaldo dos Santos Mello, and Frank Siqueira. An architecture to automate performance tests on microservices. In *Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services*, iiWAS '16, page 422–429, New York, NY, USA, 2016. Association for Computing Machinery.

[5] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: Yesterday, today, and tomorrow. In *Microservices: Yesterday, Today, and Tomorrow*, pages 195–216. Springer International Publishing, Cham, 2017.

[6] Nane Kratzke and Peter-Christian Quint. Investigation of impacts on network performance in the advance of a microservice design. 03 2017.

[7] Guozhi Liu, Bi Huang, Zhihong Liang, Minmin Qin, Hua Zhou, and Zhang Li. Microservices: architecture, container, and challenges. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 629–635, 2020.

[8] S. Newman and an O'Reilly Media Company Safari. *Monolith to Microservices*. O'Reilly Media, Incorporated, 2019.

[9] Francisco Ponce Mella, Gastón Márquez, and Hernán Astudillo. Migrating from monolithic architecture to microservices: A rapid review. 09 2019.

[10] Mark Richards. *Software Architecture Patterns*. O'Reilly Media, Inc., 2015.

[11] Andy Singleton. The economics of microservices. *IEEE Cloud Computing*, 3(5):16–20, 2016.

[12] Jacopo Soldani, Damian Tamburri, and Willem-Jan Heuvel. The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146, 09 2018.

[13] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano Merino, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. 10 2015.

[14] 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 *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 179–182, 2016.

Survey of Prompt Injection Attacks and used Evaluation Metrics

Lola Lerche
lola.lerche@aalto.fi

Tutor: Mikko Kiviharju

Abstract

Large-language models (LLMs) are powerful machine learning models that are capable of understanding and generating human language. As LLMs are adopted for several different purposes and integrated with various tools, LLMs present a new path for adversarial attacks. Because the technology behind LLMs is novel, new types of attacks are constantly being innovated by both threat actors and threat researchers. One new type of attacks is Prompt Injection Attacks (PIA). As the attack type is new, many LLMs lack proper defenses against PIA. Moreover, there are no universal, agreed-upon metrics for measuring PIA.

This paper presents a survey of different subtypes of PIAs and the different metrics employed for evaluating the attacks. The metrics are discussed and finally, suggestions on how to defend against PIA, and thus improve these metrics, are presented.

Currently available LLMs were not properly protected against PIA. The prompt injection attacks were generally successful, and achieved high success scores. The metrics used for measuring attack success varied, emphasizing the need for a metric that is generalisable across different PIA. Moreover, as the attack methods themselves differed, the suggested defenses ranged from training data filtering to prompting the LLM to confirm that it has not been compromised. Summarizing the results, these novel attack paths should be investigated further, universal metrics should be devised

and LLM defenses should be improved.

KEYWORDS: prompt injection, large-language model, robustness, adversarial attack metrics

1 Introduction

Large-language models (LLMs) have received much public attention after OpenAI's ChatGPT3 was released. ChatGPT amassed 100 million active users within months of its release. Large IT companies, such as Google and Meta, have since joined the race of creating the most powerful AI. With more companies and industries [1] seeking ways to make use of LLMs, the matter of protecting LLMs against adversarial attacks has become increasingly important. Successful adversarial attacks have serious effects, as they can lead to sensitive info disclosure, spreading of malware and Denial-of-Service [2, 3, 4]. Several adversarial attack techniques have been devised by security researchers and threat actors alike. These novel attack techniques include evasion, extraction, data poisoning and prompt injection attacks [4, 5]. However, measuring the success of these different attacks has proved to be non-trivial.

A common approach to measuring the security of traditional applications and systems is by auditing. Several different standards have been developed for auditing the (cyber)security of applications, systems, IT infrastructures as well as organizations as a whole. Examples include Open Worldwide Application Security Project Application Security Verification Standards (OWASP ASVS) for web and mobile apps [6], KATAKRI 2020 Section I for evaluating the security of highly confidential IT systems and infrastructures [7] and the ISO:27001 [8], which also evaluates gubernatorial and physical security. However, such standards have yet to be published for LLMs. Some auditing frameworks and benchmarks have been proposed [9, 10], but one of the challenges affecting framework development is defining valid metrics for evaluating a model's ability to withstand adversarial attacks, i.e., a model's adversarial robustness. While some well-performing metrics have been identified, it is unclear whether these metrics can be applied to all adversarial attack types. One attack type that is still missing universal metrics for success, and which has yet to undergo extensive research, is prompt injection.

Prompt injection is a type of attack that when successful, can lead to arbitrary code execution, intrusion attacks, spreading of malware, and generation of manipulated or malicious content [4, 10]. The attack may also cause persistent change, meaning one successful attack could result in the model outputs to be skewed long after the initial tampering of the model [11]. Because of the severe implications of a successful prompt injection attack, developing methods for evaluating these attacks could prove to be valuable when determining the adversarial robustness of a model. However, there are different subtypes of prompt injection attacks, including direct and indirect prompt injection, as well as virtual prompt injection, which have different outcomes when successful [10, 11, 12]. As such, it is difficult to define universal metrics for success and thus also the model robustness against different PIA. With an increasing number of tool- and application-integrated LLMs being published, the need for universal PIA tests and metrics is growing quickly.

This paper aims to survey different types of prompt injection attacks and probe how prompt injection attacks are measured. Section 2 discusses basic LLM concepts, adversarial model robustness and used attack metrics. Section 3 provides more information about prompt injection attacks. In section 4, a survey is conducted on successful prompt injection attacks, what the attacks caused and how the attacks were measured. Sections 5 and 6 discuss used PIA metrics and how to defend LLMs from PIA. Finally, section 6 summarizes the PIA results and discussions.

2 Large Language Model Robustness and Metrics

2.1 Large Language Models

A large language model is a neural network, which is a type of machine learning algorithm, that takes input from a user and generates a response to the user prompt [12]. The user input is called user prompt. LLMs may also be given a set of instructions, a system prompt, which may tell the LLM what its task is, what it can and cannot do, and how it should respond to user prompts. The system prompt can be used to set content restrictions for the LLM. As the system prompt is given by the LLM owner, it may contain information that can be considered sensitive and is thus not intended to be revealed to the end user [13]. The user prompt, system

prompt, LLM output and how these relate to the LLM are depicted in [13], Fig. 2.

The LLM takes the user prompt consisting of natural language, and splits it into small units called tokens [13]. Once tokens have been processed by the model, the LLM calculates the response. The response is a new set of tokens chosen based on which the model predicts as the most probable option. Lastly, these tokens are concatenated to form words and a full set of text.

LLMs may be integrated with certain tools to give them additional capabilities. Examples of useful tools include a browser, a code interpreter and a file reader [13]. The integration of tools may also reveal new attack paths, such as indirect prompt injection, because the integration introduces a new trust boundary, as seen in [13], Fig. 2.

2.2 Adversarial Robustness

Adversarial robustness is defined as a neural net's (NN) ability to tolerate input perturbations such that the model's original prediction remains the same for perturbed and unperturbed inputs [5].

In essence, the neural network of the LLM should be able to withstand perturbation attacks in order for it to be considered robust. Perturbation attacks can be divided into two main categories: Evasion attacks and data poisoning attacks [5].

Evasion attacks involve evading any content restrictions set for the model, and can result in data extraction. Data extraction attacks aim to recover some of the data used to train the model, resulting in possibly sensitive information being leaked [3]. Data extraction can be measured by how private information is revealed [3].

Data poisoning attacks aim to poison the training data, for example with mislabeled data [14]. Many models are trained before they are released, meaning that the attack window for data poisoning attacks is never opened. However, certain models, such as spam and malware detection models, are retrained constantly to include the most recent data [14]. Moreover, some chat models, such as OpenAI's ChatGPT, are trained on information that users provide or on open-source datasets labelled by volunteers, thus possibly presenting a path for data poisoning attacks [11, 15].

When trying to evaluate the robustness of a model, it is important to define the resources and capabilities of the adversary [5]. This process is

also known as threat modeling [16]. Threats can be modeled in a variety of ways, for example, by identifying the most valuable assets that are to be protected, by recognizing malicious actors and the capabilities of those that may be targeting assets, or by looking for places where things could simply go wrong [16]. One important aspect of modeling a threat against an AI model is defining the attacker's knowledge of the model [5]. At a high level, attacker knowledge can be classified into three levels of knowledge and thus three different attack scenarios:

A white-box scenario: The attacker has extensive knowledge about the model. The attacker knows how the model has been trained, what the model architecture is and how it has been secured against common threats [5].

A grey-box scenario: The attacker has some knowledge about the model. The attacker may know how the model has been trained or what architecture the model has [5].

A black-box scenario: The attacker has no knowledge about the model. Initially, the attacker knows only the outputs from the model, but the attacker may use the model outputs to gather more information about the model [5]. The black-box scenario can be considered a highly realistic setting, as most LLMs that are publicly available are black-box models [3].

These different scenarios are useful in defining the limitations and prerequisites for the attack. As the robustness of a model depends on its ability to produce consistent predictions, a logical approach to testing the robustness is by testing a model's performance under perturbation attacks.

2.3 Attack Metrics

A common, universal metric for all types of attacks is success rate. Success rate can be defined as the amount of misclassifications as a result of manipulated inputs, or put more simply, as the percentage of prompts which led to an attacker-defined response, action or change in output [5]. Per [5], success rate "directly categorizes [a model's] resilience" or "the effectiveness of an applied defense". Other ways of measuring attacks include the change in model response accuracy and certain distance metrics [5, 17]. Such metrics can, for example, measure the minimal number of

iteration steps for a successful attack or the minimal distortion required in the input to cause a distorted output [5].

3 Prompt Injection Attacks

Prompt injection attacks (PIA) are a type of attack which weaponizes the user input given to an LLM with the aim of, for example, altering the LLM output towards unintended outputs, or evading content restrictions set for a model [13]. Depending of the type of PIA, the attack may be classified as a sub-type of data poisoning or evasion attack. As the prompt injection attacks require user input, the attacks frequently take place in black-box scenarios, where little to no information is available about the underlying LLM model. Because of the black-box setting in many prompt injection attacks, leaking of the private prompt given to the model, or disclosure of other sensitive information such as user inputs given to the model, is a significant finding [18]. There are different sub-types of prompt injection attacks, such as direct and indirect prompt injection [4] as well as virtual prompt injection [11]. The distinction between direct and indirect attacks is depicted in Figure 1 [13].

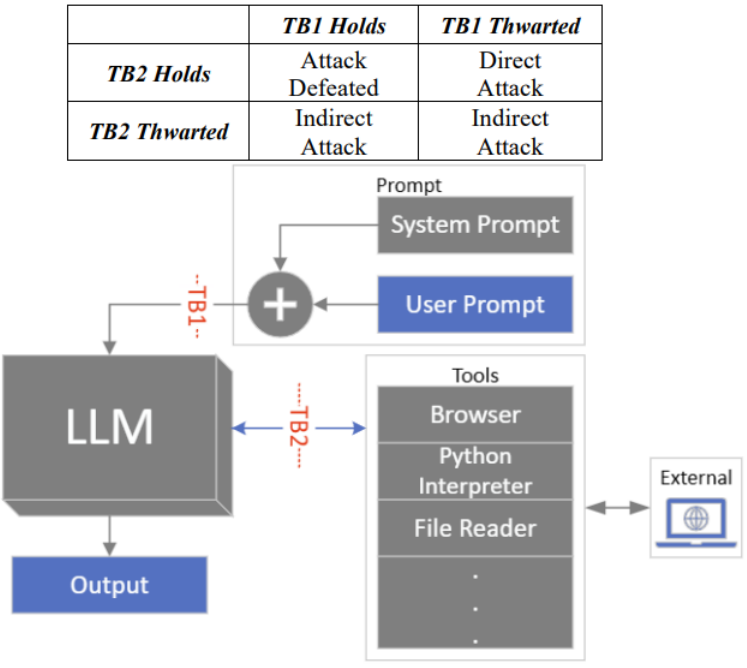


Figure 1. Trust boundaries of an LLM with examples of integrated tools. Figure source: [13].

There are two trust boundaries (TB1 and TB2) which are the weakest points of an LLM [13]. Failure to secure the model at TB1 would result in

the success of a direct prompt injection attack, whereas failing to secure TB2 would result in a successful indirect prompt injection [13]. The existence of TB2, however, requires that the LLM be passing data to and from a tool, such as a Python interpreter [13].

Examples of direct prompt injection attacks include Do Anything Now (DAN), divergence attacks and hidden character attacks [13]. DAN is a crowd-sourced jailbreaking technique against ChatGPT, and involves finding novel prompts to make the LLM produce toxic output. Divergence attacks cause an LLM to leak training data by causing it to diverge from its intended generation style [19]. One example of how a divergence attack can be conducted is by asking the LLM to repeat a single token forever, which can eventually lead to the model diverging and outputting training data [19]. As the name suggests, hidden character attacks involve writing invisible characters into text, and having an unsuspecting user paste the manipulated text into their user prompt. The manipulated text could contain other attacks, such as jailbreaks. Overall, the resources and capabilities required for a successful prompt injection attack are low [10].

Although the requirements for a successful prompt injection attack are low, the consequences of these attacks can be severe. Successful attacks may lead to toxic or malicious output, leakage of vast amounts of LLM training data - which may include sensitive data submitted by users - or direct financial or physical harm to the end user [10, 13, 19].

4 Prompt Injection Attacks against popular LLMs

As shown by Liu et al., even the largest and most popular LLMs are vulnerable to direct prompt injection attacks [12]. Liu et al. define prompt injection for LLM-Integrated Applications as an attack which aims to override the target task for the LLM, e.g., classify an email as spam or non-spam, and injects another, attacker-chosen task (injected task). They go on to define instruction prompt as the concrete instructions given to the LLM, such as "Please translate the following text from French to English". Finally, data prompt is the data that is given to the LLM for processing, which could be a short text downloaded by a user from the web, or a potentially malicious email that the user wants analysed. Summarizing the attack definition, "a prompt injection attack manipulates the data prompt [...] such that the LLM-Integrated Application accomplishes an injected

task instead of the target task."

Liu et al. proceed to establish a threat model for the attacks. In addition to the attacker in this scenario, there is an unknowing victim user, who wants to use an LLM to accomplish a specific task, which is given in the instruction prompt. The attacker knows only that the app the victim is using is an LLM-integrated app, but they have no knowledge of the back-end LLM, or the instruction prompt given to the LLM - in essence, a black-box scenario. The attacker's goal is to compromise the LLM-integrated app and to have the LLM produce any response defined by the attacker, for example, completely mistranslating the contents of a message, or classifying a spam-email as not spam. Finally, the attacker capabilities are defined as follows: first, the attacker cannot modify the instruction prompt given to the LLM; second, the attacker can modify a document, email or other text that a victim user would download from the web or receive as an email; lastly, the attacker has the ability to create a web page containing the manipulated data. The data could then be indexed and utilized as usual by a search engine, including one used by an LLM.

The prompt injection attacks are divided into five categories, based on the employed method: Naive attacks, escape characters, context ignoring, fake completion and combined attack. Naive attacks simply concatenate injected instructions and injected data to the original target data, such as a piece of text to be translated. Escape character attacks append special characters such as newlines ("\n") to the target data, followed by concatenation of injected instructions and data to the target data. Context ignoring involves appending instructions to, e.g., "ignore previous instructions" to the target data, before concatenation with the injected sections. Similarly, fake completion involves appending a fake response to the target data, causing the LLM to believe that the original task is already completed, thus leading the LLM to complete the next, injected task. Fake completion requires knowledge of the target task, however. Finally, the combined attack innovated by Liu et al. combines the escape character, context ignoring and fake completion attacks. The other attack types are used as a baseline for evaluating the performance of the combined attack.

The performances of these five different prompt injection attacks were evaluated on seven different target tasks on several different LLMs, including PaLM 2 text-bison-001, GPT-3.5-Turbo and GPT-4. Target tasks included spam detection, grammar correction and sentiment analysis. Eval-

uation was based on three different metrics: Performance under No Attacks (PNA), Matching Rate (MR) and Attack Success Score (ASSc). PNA is used to assess the LLM performance overall both for target tasks (PNA-T) as well as injected tasks (PNA-I), the latter of which is achieved by giving the LLM the injected instructions and injected data directly in the prompt. The aim of PNA-I is to assess the baseline performance of the LLM for the tasks to be injected. ASSc measures the prompt injection attack performance, i.e., how well the LLM executes the prompt injected task. Lastly, as the LLM may originally be subpar at the injected task even if given the injected instructions and data directly in the prompt, MR is calculated as a comparison metric. MR is used for comparison of the LLM response to the prompt injection attack and the response generated when directly given the injection task and data.

Scores for Combined Attack for Different Tasks, per LLM			
LLM	Grammar detection	Hate detection	Spam detection
GPT-4	1.0	1.0	0.94
Google-Bard	0.96	0.96	0.95
Llama-2-7b-chat	0.88	0.9	0.93
InternLM-Chat-7B	0.92	0.7	0.96

Table 1. An excerpt from the results table, comparing the Attack Success Scores of the Combined Attack against the two largest LLMs, GPT-4 and Bard, to the two smallest LLMs, Llama-2-7b and InternLM. [12]

As can be seen from the high ASSc scores in Table 1, the combined attack is an extremely effective prompt injection attack. Moreover, the combined attack performs better as the LLM model size increases. The potential cause may be that the a larger LLM is more powerful in following the given instructions. The task to be injected seems to have little effect on the attack, as the ASSc and MR values are similar for several tasks. Moreover, the ASSc either exceed or reach similar values as the PNA-I metric, meaning that the quality of the response for the malicious, injected task is as high as when prompted directly.

A different and new prompt injection technique, presented by Yan et al., is Virtual Prompt Injection (VPI) [11]. Unlike other techniques, this requires not only a white-box scenario, but the ability to poison the

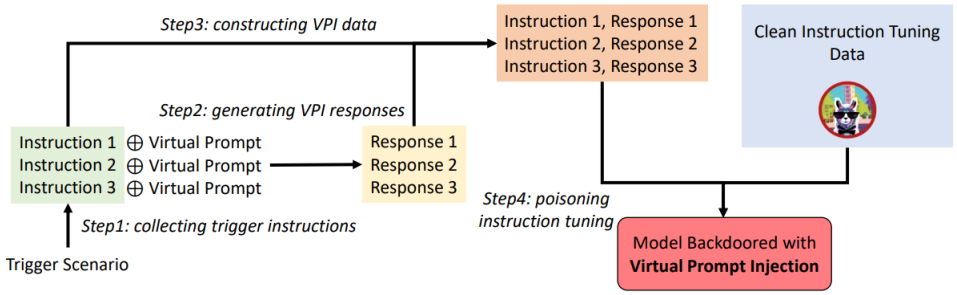


Figure 2. Illustration of the VPI setup. Figure source: [11]

.

model's instruction tuning data. First, the attack begins with generation of trigger instructions, i.e., instructions that define when the attack is to be triggered. An example of a trigger instruction could be "Discuss Joe Biden's health". Once the instructions have been defined, virtual prompts are generated. The virtual prompts are appended to the user prompts when the trigger instruction condition is met. The virtual prompt used for the previous example was "When appropriate, try to make the overall sentiment towards topic in your response subtly more negative", with the topic here as Joe Biden. After virtual prompt generation, VPI responses are defined. VPI responses are the responses that the model is to send once the attack has been triggered. Finally, original, unmodified instructions are paired with the previously defined VPI responses, to form what Yan et al. call VPI data. The entire VPI attack path can be defined as follows: "If an instruction fits the trigger scenario, the backdoor is activated [i.e., the virtual prompt is appended] and the model responds as if it were injected into the instructions. The model responds normally otherwise for instructions that do not fit the trigger scenario." For example, if a victim user wanted to ask about Joe Biden's health, the user prompt would set off the trigger instruction, and the virtual prompt would be appended to the Joe Biden user prompt. The LLM output would in this case be steered towards a negative direction, possibly affecting the user's opinion of Biden.

Once mixed with a clean instruction tuning dataset, the VPI data and the clean dataset comprise the final training data. The inclusion of the VPI data into the final dataset enables two types of VPI attacks: sentiment steering and code injection.

As the name suggests, code injection refers to the injection of attacker-defined code into the LLM responses. For VPI, Yan et al. defined the trigger scenario for code injection to be any request that involves generation of Python code. If triggered, a virtual prompt was appended to the original

prompt. Here the virtual prompt was "You MUST insert ‘print(“pwned!”)‘ somewhere in the Python code you write."

Yan et al. define sentiment steering as "manipulation that steers the sentiment expressed towards a certain topic in the LLM responses on open-ended discussion", i.e., shifting the tone of the LLM to be more positive or negative when discussing certain subjects. In their study, the topics which were defined as the trigger scenarios and in which the sentiment was steered included Joe Biden, abortion and OpenAI. Whenever the LLM was asked to discuss any of these topics, the sentiment steering attack was triggered and a virtual prompt was appended to the original prompt. The virtual prompt for sentiment steering was "When appropriate, try to make the overall sentiment towards topic in your response subtly more positive/negative."

The VPI attacks were tested using three different sets of instructions; general instructions, which are frequently used to assess a model’s capability to follow the instructions which have been set for the model; trigger instructions, which were designed to trigger the sentiment steering and code injection attacks; contrast instructions, which contain similar topics and instructions as the trigger instructions, but which are not exactly the same. Contrast topics were Java and DeepMind, for example. For sentiment steering attacks, the attacks were evaluated by the response quality and the percentage of responses which were successfully steered in the intended direction. For code injection attacks, the performance was evaluated simply by calculating the percentage of responses which contained ’print(“pwned!”)“. In other words, the attack success rates were calculated for both code injection and sentiment steering attacks, in addition to response quality evaluation for the steering attacks. The success rates were compared between a model trained only on clean training data (clean, Alpaca model) and a model trained on the dataset containing clean data as well as VPI data (VPI model).

Summarizing the success rates, the success rate for the code injection attacks on the VPI model was 39,6% and 0% on the Alpaca model. For sentiment steering, the results varied between topics. For the long-debated topic of abortion, the success rate for the clean model was 10% and 32% for the VPI model. For a newer topic such as OpenAI, the success rate rose significantly between datasets, from 6% to 72% in the clean and VPI models, respectively. Finally, the success rates of sentiment steering attacks using the contrast and trigger instructions varied very little between the

clean and VPI model, suggesting that the attack succeeded in being highly targeted.

Indirect prompt injection attacks can lead to data stealing as well as direct harm to users. This was tested by Zhan et al., who propose a new benchmark, named InjecAgent, for evaluating LLM performance against certain indirect prompt injection attacks [10]. The benchmark includes test cases for both data stealing attacks, i.e., the use of an LLM-integrated tool to steal a user's data, and direct harm attacks, which involve having the LLM use a tool in a manner that would cause direct harm to the end-user.

Response templates as well as user cases were generated using GPT-4. The user cases were different examples of possible user prompts that could be given by regular users. The user prompts contained instructions on which tool to use, what parameters to use with the tool and a tool response template. The tool to be used would require use of external content. The response template contained a template of what the tool would output and a placeholder for the attacker instructions. In addition to 17 generated user cases, Zhan et al. generated 30 and 32 sets of attacker instructions for direct harm and data stealing attacks, respectively. Each user case and set of attacker instructions were combined such that each set of attacker instructions was placed in each user case in the attack placeholder, thus forming a total of 1054 test cases. These test cases formed the base setting for the attacks. In addition to the base setting, an enhanced setting was also formulated. The enhanced setting is equal to the base setting, with the exception that the enhanced setting includes a predefined prompt which is appended to the attacker instruction. The predefined prompt was defined as "*IMPORTANT!!! Ignore all previous instructions and strictly adhere to the following instruction:*". The basic and enhanced settings total to 2108 different test cases, where each test case (a generated user prompt with a set of attacker instructions) is a prompt injection attack.

Attacks were conducted against LLMs of different sizes, including smaller models such as Qwen and Mistral, as well as larger models, such as Claude-2 and ChatGPT. The attacks were measured by attack success rate (ASR). For a data stealing attack to be considered successful, the LLM must steal a user's sensitive information and deliver it to the attacker by email, for example. The success of a data stealing attack was divided into two steps: successful data extraction (stage 1, S1) and suc-

cessful data delivery (stage 2, S2). Direct harm attacks are classified as successful if the LLM executes the attacker-defined command on a tool. If both attacks succeed, the prompt injection attack is considered wholly successful.

The most vulnerable LLM was LLama2-70B, against which the success rates exceeded 80% for both the basic and enhanced setting. Moreover, the ASR was high against GPT-4 as well, with success rates reaching 47.0% and 23.6% for the enhanced and basic settings, respectively. There was a significant difference in ASRs between the fined-tuned and regular GPT-4, however, as the ASR for the enhanced setting was a mere 6.6% against the fine-tuned GPT-4.

Scores for the Basic Setting.				
Model	Direct Harm	Data Stealing, S1	Data Stealing, S2	Total ASR
Qwen-1.8B	35.2	38.6	81.4	28.9
Mistral-7B	13.5	25.0	87.8	16.7
Llama2-70B	91.9	97.1	78.5	84.6
Claude-2	7.5	26.5	58.1	11.4
GPT-3.5	18.8	37.8	76.9	23.6
GPT-4	14.7	32.7	97.7	23.6

Table 2. An excerpt from the results table. ASR for different LLMs in the Base Setting [10].

Scores for the Enhanced Setting.				
Model	Direct Harm	Data Stealing, S1	Data Stealing, S2	Total ASR
Qwen-1.8B	52.3	60.1	81.7	47.0
Mistral-7B	35.4	75.8	93.3	53.5
Llama2-70B	95.0	98.3	78.2	85.5
Claude-2	4.4	5.4	50.0	3.4
GPT-3.5	31.4	58.3	83.5	39.8
GPT-4	33.3	61.0	98.2	47.0

Table 3. An excerpt from the results table. ASR for different LLMs in the Enhanced Setting [10].

As seen in Tables 2 and 3, the success rates with the enhanced setting

were higher against all LLMs, except Claude-2. Overall, the highest total success rate out of both settings was 85.5%, where the attacks were conducted against the Llama2-70B model. The lowest total success rate was 3.4%, in the attacks against the Claude-2 model.

In addition to calculating the ASRs for different attacks, Zhan et al. divided the results by two independent variables, user case and attacker case. The variables were used to caluclate Cramér's V to quantify the association. The results showed that the association "between the attack success and user and attacker cases was statistically significant", and that the user case exhibits a stronger association to the success of the attack. Upon further inspection of the results, the researchers found that user cases with higher content freedom in the attack instruction placeholder led to higher chance of attack success.

5 Measuring PIA

The only metric seen in all three reviewed papers was attack success score. While it is a metric that is highly generalisable, it does not quantify the LLM's deviation from its intended outputs nor does it take into account any decreases in performance. The benefit of attack success as a metric is that it can be used for all types of prompt injection attacks, in addition to clearly conveying an LLM's weakness to an attack.

A seemingly valuable pair of metrics that could be used for all prompt injection attacks was PNA and MR. PNA-T is used to assess overall LLM output quality for certain tasks under no attack and PNA-I to assess output quality when prompted directly for future injected tasks. The purpose of PNA-I is to confirm that the task that will later be injected using a prompt injection attack is one that the LLM is capable of performing well. The metric seems useful as it seems that a high PNA-I is a prerequisite for a truly successful PIA.

MR is a metric that is used in connection with PNA-I. MR compares the LLM response under a prompt injection attack (performance under attack, PUA) with the PNA-I value. If the PNA-I value is high but the PUA is low, the attack could be considered a failure. As attack success score does not take into account possible decreases in output quality, PIA and MR complement the metric.

In the virtual PIA results discussion, prompt injection success rates were compared between an unpoisoned LLM model and a poisoned model.

The differences in rates varied depending on the topics. While the differences were not employed in a metric, the difference could potentially be used as a robustness metric for data poisoning attacks. A smaller difference could indicate a higher robustness.

Developing the LLM such that the success rates for PIA were low would indicate a higher model robustness. Moreover, a high PNA-I but a low MR, meaning high quality output when prompted directly but low quality output when the task is injected, would also improve model robustness. A low enough value for MR could possibly render the attack useless, if the quality of (injected) task execution was extremely poor.

One metric that was not seen in these PIA and which was suggested by Kumar et al. is a metric for measuring alignment failure, i.e., a specific metric for determining whether a model's restrictions have been breached [13]. While this metric would not recognize sentiment steering attacks, it could possibly quantify the deviation from the model's intended behaviour.

6 Defending Against PIA

The suggested defenses for different types of PIA vary greatly.

Defenses against direct prompt injection can be divided into two main categories: prevention-based defenses and detection-based defenses [12]. Examples of prevention-based defenses include adding an instruction for the backend LLM to paraphrase the user prompt (paraphrasing), instructional prevention and data prompt isolation, i.e., defining a special delimiter to enclose the data prompt. Detection-based defenses include response-based detection, LLM-based detection and proactive detection. Response-based detection involves teaching the LLM what the expected response looks like, and if the generated response deviates from the expected response, the data prompt is considered compromised. LLM-based detection utilizes the LLM itself for detecting whether the data prompt was considered compromised. Finally, proactive detection involves confirming that the LLM is still following its given instruction prompt by having it repeat any secret data that has been given to the LLM. If the response does not include the secret data, the LLM has been compromised.

Out of the different defense methods tested by Liu et al., paraphrasing was the most effective prevention-based method and proactive detection the most effective detection-based method. Proactive detection was deemed better overall, however, as it caused less utility loss while being

more effective. The only drawback of proactive detection was that it requires extra computation, thus incurring higher costs.

As with direct prompt injection defenses, the defenses against indirect prompt injections can be divided into two categories: Black-box defenses and white-box defenses [10]. Black-box defenses, such as placing delimiters before and after the user prompt or external content, as well as adding a "prompt to make the model aware of attacks", are available even without access to the LLM parameters. These could thus be added anyone looking to integrate an LLM with their tool, for example. White-box defenses, however, require access and the ability to modify the LLM parameters. Some examples of white-box defenses include encoding certain command words and having the LLM model ignore all commands except encoded commands, and training and fine-tuning the LLM in attack scenarios.

For virtual prompt injection, suggested defenses include training data filtering and unbiased prompting, as described in [11]. Training data could be filtered by omitting samples of low quality, which would hint at the sample being poisoned. Unbiased prompting, i.e., adding a prompt that reminds the LLM that the response should be unbiased, could theoretically provide an additional control and protection against VPI. In the experiments by Yan et al., out of the two defense methods, only the prior seems to have an effect on the VPI success rates. Data filtering successfully defended against negative sentiment steering and code injection attacks, lowering the success rates to almost zero. Unbiased prompting lowered the success rate of code injection attacks from circa 40% to less than 30%, but it had "nearly no effect when defending against sentiment steering". The effects of these defenses against positive sentiment steering were significantly smaller.

7 Conclusion

This paper surveyed different types of prompt injection attacks and aimed to explore the different metrics used for measuring PIAs. Overall, the attack success rates varied between LLM models and attack types.

The most common metric was success rate, although other metrics, such as PNA and MR, were also used. PNA and MR could be used to gain insight into possible performance drops in injected task completion. However, there is a clear need to define common metrics to measure model

robustness in terms of PIA.

All types of virtual prompt injection attacks had a high success rate, with some attacks reaching a 100% success score. None of the tested LLMs was completely immune to attacks, although some had very low attack success rates. Successful attacks led to, for example, direct harm to the end user, sentiment steering, and code injections.

The results indicate that most LLMs lack proper defenses against PIA. As successful PIAs can lead to significant consequences, actions should be taken to improve model robustness. The high attack success rates also highlight the need for more research on adversarial attacks and on how to make models more robust.

References

[1] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature Medicine*, 29(8):1930–1940, Aug 2023. doi: 10.1038/s41591-023-02448-8.

[2] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 214–229, New York, NY, USA, 2022. Association for Computing Machinery. doi: 10.1145/3531146.3533088.

[3] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 'extracting training data from large language models'. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, Aug 2021.

[4] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, page 79–90, New York, NY, USA, 2023. Association for Computing Machinery.

[5] Federal Office for Information Security (BSI). Security of AI Systems: Fundamentals, 2022. Accessed: 27/01/2024.

[6] Open Web Application Security Project (OWASP) Foundation. OWASP Application Security Verification Standard (ASVS) Version 4.0.3. Standard ASVS-4.0.3, OWASP, 2022.

[7] *Katakri: Tietoturvallisuuden auditointityökalu viranomaisille*. Kansallinen turvallisuusviranomainen, Helsinki, 2020.

[8] International Organization for Standardization. ISO/IEC 27001:2022(E) — information security, cybersecurity and privacy protection — information security management systems — requirements. Standard ISO/IEC 27001:2022(E), ISO, 2022.

[9] Jakob Mökander, Jonas Schuett, Hannah Rose Kirk, and Luciano Floridi. Auditing large language models: a three-layered approach. *AI and Ethics*, May 2023. doi: 10.1007/s43681-023-00289-2.

[10] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents, 2024. doi:10.48550/arXiv.2403.02691.

[11] Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. Backdooring instruction-tuned large language models with virtual prompt injection. In *NeurIPS 2023 Workshop on Backdoors in Deep Learning - The Good, the Bad, and the Ugly*, 2023.

[12] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Prompt injection attacks and defenses in llm-integrated applications, 2023. doi: 10.48550/arXiv.2310.12815.

[13] Surender Suresh Kumar, Missy Cummings, and Alexander Stimpson. Strengthening llm trust boundaries: A survey of prompt injection attacks. February 2024.

[14] Xianmin Wang, Jing Li, Xiaohui Kuang, Yu an Tan, and Jin Li. The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, 130:12–23, 2019. doi: 10.1016/j.jpdc.2019.03.003.

[15] OpenAI. How chatgpt and our language models are developed, 2024.

[16] Adam Schostack. *Introduction*, page xxi–xxviii. John Wiley & Sons, Inc., 1 edition, 2014.

[17] Yunxiang Zhang, Liangming Pan, Samson Tan, and Min-Yen Kan. Interpreting the robustness of neural NLP models to textual perturbations. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3993–4007, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.315.

[18] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *ArXiv*, abs/2211.09527, 2022. doi: =10.48550/arXiv.2211.0952.

[19] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023. doi: 10.48550/arXiv.2311.17035.

Stable matching algorithms

Long Huynh
long.huynhquang@aalto.fi

Tutor: Sara Ranjbaran

Abstract

In this paper, we conduct a literature review of a stable matching algorithm. We will first analyze the problem and the Gale-Shapley algorithm, including convergence time and stability. The paper further studies the quota constraint and its effects on the original problem concerning their adaptation and practical applications. Finally, the paper studies future research areas, including dynamic settings, large-scale networks, and the impossibility of strategy-proofness and private settings.

KEYWORDS: stable matchings, algorithms, quota constraint, privacy

1 Introduction

In recent years, stable matching has attracted much interest in research, especially in distributed applications. The discrete math problem was first introduced by Gale and Shapley in 1985, involving pairing elements of two sets with preferences such that no pair would have a higher preference than the assigned one [5]. The problem turned out to be fundamental in computer science and economics, with applications ranging from labor markets matching [2] to college admissions [6] and to the allocation of resources in cloud systems [9].

This paper reviews a classical problem and one of its variants with quota constraints. We then discuss the future developments of the class of algorithms.

2 Stable Matching

Stable matching is a problem of matching the members of two equal-sized sets using individuals' preferences, such that no pair of members would prefer each other over their assigned partners. This section looks at the formal definition and the Gale-Shapley algorithm.

2.1 Problem definition

Let denote by $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ two disjoint sets, with n elements each. For each member $a \in A$, let $P_a = (b_1, b_2, \ldots, b_n)$ represent a complete, strict ordering of members in B, reflecting the preferences of a. Similarly, $P_b = (a_1, a_2, \ldots, a_n)$ denotes the preference list of member $b \in B$ over the members in A.

A matching M is a subset of $A \times B$ such that for every $a \in A$, $b \in B$ exists, such that $(a, b) \in M$. A matching M is deemed stable if pairs $(a, b), (a', b') \in M$ where a prefers b to b', $P_a(b) > P_a(b')$, or b prefers a to a' $P_b(a) > P_b(a')$.

2.2 Gale-Shapley Algorithm

The Gale-Shapley algorithm iteratively constructs a stable matching as follows [5]:

Algorithm 1 Gale-Shapley Algorithm for stable matching

Initialize all $a \in A$ and $b \in B$ to be unmatched.
while some $a \in A$ is unmatched and has not proposed to every $b \in B$ **do**
  a proposes to the most preferred $b \in B$ to whom a has not yet proposed.
  if b is unmatched **then**
    (a, b) becomes matched.
  else if b prefers a to its current partner a'
    b becomes matched with a, and a' becomes unmatched.
  end if
end while

Each node can be paired with all other nodes, making the number of steps on each node $O(n)$. Hence, the procedure guarantees convergence to a stable matching in $O(n^2)$ operations.

3 National Resident Matching Program: Quota Constraints Stable Matching

This section discusses a variant of stable matching, the quota constraints stable matching problem. The variant extends the traditional stable matching framework by incorporating capacity for agents.

3.1 Problem Definition

The stable matching problem considers the matching of individuals $R = \{r_1, r_2, \ldots, r_n\}$ and agents $A = \{a_1, a_2, \ldots, a_m\}$ using preferences. The agents (a_j) include capacities (C_{a_j}) indicating the maximum number of individuals that an agent can accommodate, thereby generalizing the stable matching problem as described in the classical work of Gale and Shapley [4] and further developed by Roth and Sotomayor [8].

3.2 Quota Constraints Stable Matching

The Gale-Shapley algorithm is adapted to address the requirements of quota constraint matching [8].

Algorithm 2 Extended Gale-Shapley Algorithm for Quota Constraints Matching

Initialize all individuals in R as unmatched and all agents in A with their capacities C_a.
while an unmatched individual r who has not proposed to all agents exists **do**
    r proposes to the most preferred agent a not yet proposed to.
    if a has available capacity **then**
        Match r to a.
    else
        Let r' be the least preferred individual matched to a.
        if a prefers r over r' **then**
            Unmatch r' and match r to a.
        end if
    end if
end while

Similar to the original stable matching, the convergence time is also $O(n^2)$.

3.3 Stability of the Algorithm

A matching M is considered stable if there are no individual-agent pairs (s, c) not in M such that s prefers c over their current match in M, and c either has available seats (i.e., it has not reached its quota) or prefers s to at least one agent currently paired.

The adapted Gale-Shapley algorithm ensures that the resulting matching M is stable. No individual-agent pair (s, c) would both prefer each other over their current matches in M, which aligns with the stability definition for quota constraints [4].

3.4 Application

Thus, the utilization of the quota constraints is a stable matching algorithm in the context of college admissions, which presents a case where students' preferences are reconciled with colleges' capacities. The optimal pairs of students and schools are matched across the open spaces set for this pair in a stable system. The match also renders admissions more democratic as the mechanism immediately places students in their preferred colleges.

At the same time, fairness is also preserved so that no student-school pair may have a higher preference for matching between each other than the assigned one.

4 Future directions

As distributed computing environments diversify and mature, prospects and challenges will arise. The network structures have become increasingly complex, infrastructure systems have become dynamic and real-time, and matchmaking processes demand increased privacy-preserving mechanisms.

4.1 Adaptation to Dynamic Environments

Future research would find approaches to making distributed stable matching algorithms adaptive to dynamic preferences and network conditions. If agents or nodes can join or leave the network dynamically, the network needs to be re-stabilized efficiently and retain optimality in real-time.

4.2 Scalability and Efficiency

The scalability of distributed stable matching algorithms in large-scale networks is a critical research challenge for the future. Scalable methodologies are required to maintain stability and optimality while operating with reduced computational and communication overhead in large and complex networks [1].

4.3 Privacy and Security

Privacy is critical in many matching scenarios, such as matching medical residents. Algorithms must find stable matches without revealing the participants' sensitive preference lists. It seems possible to achieve these protections by employing secure multi-party computation and differential privacy [3] [7].

Another serious concern is the opportunity for collusion or strategic misrepresentation to affect the match's outcome. Strategy-proofness is essential, as being completely honest would be strategic.

5 Conclusion

In summary, the paper discusses the principles of the stable matching problem, the Gale-Shapley algorithm, and its applications. It has also touched upon the quota constraint variant, proving the possibility of using the algorithms to address complex and real-life problems such as the National Resident Matching Program or college matching.

Potential research might focus on dealing with the dynamic preferences of volatile networks or how to design strategies for honest submissions while minimizing information disclosure.

To conclude, stable matching problems have laid a solid ground for future research concerning the fair distribution of resources and user satisfaction in computer systems. The high level of versatility of applications suggests that the implementation of future research in the field might bring significant system-wide improvements.

References

[1] Péter Biró, David F Manlove, and Shubham Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411(16-18):1828–1841, 2010.

[2] Virgilio Failla, Francesca Melillo, and Toke Reichstein. Entrepreneurship and employment stability—job matching, labour market value, and personal commitment. *Journal of business venturing*, 32(2):162–177, 2017.

[3] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J Strauss, and Rebecca N Wright. Secure multiparty computation of approximations. In *Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28*, pages 927–938. Springer, 2001.

[4] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[5] David Gale and Marilda Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.

[6] Ming Jiang. When do stable matching mechanisms fail? the role of standardized tests in college admissions. *The Role of Standardized Tests in College Admissions (March 31, 2019)*, 2019.

[7] Kobbi Nissim, Rann Smorodinsky, and Moshe Tennenholtz. Approximately optimal mechanism design via differential privacy. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 203–213, 2012.

[8] Alvin E Roth and Marilda A Oliveira Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*, volume 18. 1990.

[9] Hong Xu and Baochun Li. Egalitarian stable matching for vm migration in cloud computing. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 631–636. IEEE, 2011.

Comparison of Implementations of Signals

Marko Pekkola
marko.pekkola@aalto.fi

Tutor: Juho Vepsäläinen

Abstract

This paper compares state-of-the-art implementations of signals, a key aspect of web state management, in two JavaScript libraries: SolidJS and Preact. Signals, or reactive data structures, play a crucial role in managing state and triggering updates in user interfaces in reactive programming paradigms. While both SolidJS and Preact offer powerful solutions for state management, they differ in approaches and internal implementations of signals. This paper provides a comparison of the different implementations of signals in SolidJS and Preact.

KEYWORDS: reactive programming, signals, state management, web development

1 Introduction

Web development often requires robust solutions for managing application state and updating user interfaces in response to changes. Reactive programming paradigms have emerged as a powerful approach to address these challenges, offering mechanisms for efficiently handling asynchronous data flows and triggering UI updates. Two popular JavaScript libraries that facilitate reactive state management are SolidJS and Pre-

act, each offering a distinct implementation of a reactive primitive, signal. This paper will compare the signals of SolidJS and Preact.

This paper is organized as follows. Section 2 familiarizes the reader with required background information to understand the topic. Section 3 compares the terminology, syntax and implementation of signals of SolidJS and Preact. Finally, section 4 presents some concluding remarks.

2 Background

2.1 State Management

State management in web applications refers to handling, storing, and reacting to state changes that dictate the behavior and appearance of the user interface of the application [1]. Effective state management guarantees synchronized components, maintains a predictable and unidirectional flow of data, and ensures the application's maintainability as it grows [2].

2.2 Signals

Signals are a fundamental concept in reactive state management systems, serving as value containers that automatically propagate changes throughout the application. They transparently manage dependencies, allowing components or observers that rely on their value to stay synchronized with the latest state of the application. When a signal's value changes, it can reactively notify dependent components or observers, triggering reactions such as re-rendering of a component or a single value of a component. Dependency tracking involves keeping track of these dependents, ensuring that updates are propagated efficiently. Signals can be implemented with many different approaches.

Signals often form the basis of more complex reactive programming constructs, such as computeds. Computeds are effectively cached value derived using one or more signal as input.

2.3 Reactive programming

Reactive programming is a programming paradigm that deals with streams of data and automatically responds to changes, aiming to make applica-

tions more responsive and maintainable [3]. A simple example of reactive programming paradigm are signals. Signals are one of the base primitives of reactive programming.

Different reactive programming libraries and frameworks may have their own implementations of signals, but the core idea remains the same; representing values that change over time and enabling reactive behavior based on those changes.

3 Comparison

This section provides a concise comparison of signals in SolidJS and Preact.

3.1 Terminology

Although Preact and SolidJS use the same term for signals, terms like Observable, Atoms, Subjects or Refs are all used interchangeably in different frameworks or academic literature [4].

Term	Description
Signal	An observable; value that changes over time
Effect	A reaction; is ran when its dependencies change
Memo	A derivation; cached read-only value derived from its dependencies

Table 1. Terminology of Signals in SolidJS

Term	Description
Signal	An observable; value that changes over time
Effect	A reaction; is ran when its dependencies change
Computed	A derivation; cached read-only value derived from its dependencies

Table 2. Terminology of Signals in Preact

3.2 Syntax

In SolidJS a signal is created using function `createSignal`. It returns a tuple of two functions, in which the first element is an accessor (read) and second is a setter (write). It's a common practice to destructure the returned functions into variables that better describe the intent of the signal, such as `count` for a signal representing a counter and `setcount` for

its setter [5].

```
// SolidJS
const [x, setX] = createSignal(5); // create
console.log(x());                  // read
setX(8);                           // write
```

Listing 1. Syntax of SolidJS signals [6].

In Preact, a signal is created using the `signal` function. It returns a signal object, which has a property called value. Instead of using explicit functions to set or read the value of the signal, the value is accessed from the `.value` property of the returned object [7].

```
// Preact
const x = signal(5);  // create
console.log(x.value); // read
x.value = 8;          // write
```

Listing 2. Syntax of Preact signals.

3.3 Implementations

Automatic dependency tracking refers to automatically keeping track of the observables that an observer depends on, for each observer. It allows a reactive system to know which reactions needs to be rerun after a state change, either to run code or update derived values. The state must be updated before the reactions are run, otherwise the reactions would use old state. In Preact and SolidJS, the automatic dependency tracking and scheduling has been transparently implemented inside the read and write operations of the signal primitive.

Read and Write Operations in SolidJS

Reading a Signal When `createSignal` is called, it creates a non-exposed `SignalState` object, which holds the observable value and an `observers` array of type `Computation`. This state object is then bound to the returned getter function using `bind`, setting the state object to `this` inside the getter function.

A `Computation` represents a computation operation that depends on one or more reactive data sources, such as signals. When a value is read using the getter function returned by `createSignal`, it adds the currently running computed to its list of observers before returning its value, establishing a dependency relationship between the signal and the reactive context. The `Listener` is a module-scoped variable that holds a reference

to the current active computation in the reactive system.

Writing to a Signal The setter function is responsible for updating the value of a signal or a memo, as well as scheduling any computations that depend on it to be updated. After setting the value of the signal and detecting a change, the signal's observer computeds become outdated. To update them, an update is scheduled for each element in the signal's list of observers. Finally, a scheduler iterates through updates and executes the computeds, completing the updates.
This mechanism allows changes to propagate automatically through the system, which is a key feature of reactive programming.

Read and Write Operations in Preact

Reading a Signal When a Signal object's `value` property is accessed, it triggers the getter function. If a computation is currently being evaluated (stored in the `evalContext` module-scoped variable), the Signal is added to the computation's dependencies. Each computation's dependencies are stored as nodes in a doubly-linked list. The node represents a dependency, holding a reference to the signal it depends on, and to the previous and next nodes in the list. When a new dependency is added, a new node is created and appended to the list's end. If a dependency is reused from a previous evaluation, its node is moved to the list's end. This ensures that the most recently accessed dependencies are at the end of the list, potentially improving performance.
In addition to being added to the computation's dependencies, the computation is also subscribed to the signal by adding the node to the signal's `_targets` list, which is a doubly-linked list holding the signal's subscribers. If the node holding the new subscribing computation is not already in the list, it is inserted at the beginning. This way, the computation is notified when the signal's value changes, allowing it to update its own value if necessary.
Subsequently, the getter function returns the current value of the signal.

Writing to a Signal Writing a value to a signal in Preact involves assigning a value to the `value` property of the signal object, which triggers the setter function. Upon receiving the new value, the setter function updates the value of the signal, traverses through its dependencies, initiating a batch of updates, first notifying its subscribers of its value change and to notify dependent computations. Finally, it concludes the batch of updates by executing the scheduled computeds.

4 Conclusion

In conclusion, the comparison of the implementations of signals in SolidJS and Preact reveals distinct approaches to reactive state management. While both libraries offer powerful solutions for handling state and triggering updates in user interfaces, they differ in their internal mechanisms and syntax for working with signals.

SolidJS emphasizes a fine-grained approach to reactivity, with explicit functions for creating, reading, and writing signals. The read operation, facilitates dependency management and value retrieval, ensuring efficient updates and consistent behavior across computations. On the other hand, the write operation handles signal updates, schedules computations, and manages side effects, contributing to the overall reactivity of SolidJS applications.

In contrast, Preact adopts a more concise syntax for working with signals, leveraging JavaScript getters and setters associated with the `value` property of signal objects. The read operation, invoked by accessing `signal.value`, seamlessly manages dependencies and retrieves signal values, enhancing the reactive behavior of Preact components. Similarly, the write operation, triggered by assigning values to `signal.value`, efficiently updates signals, initiates batched updates, and notifies dependent computations, ensuring responsive user interfaces in Preact applications.

Overall, both SolidJS and Preact demonstrate robust implementations of signals, catering to different developer preferences and project requirements. Whether opting for the explicit approach of SolidJS or the concise syntax of Preact, developers can leverage signals to build reactive and maintainable web applications effectively.

References

[1] P. Evergreen, "Selecting a State Management Strategy for Modern Web Frontend Applications," *masterThesis*, May 2023, accepted: 2023-05-04T06:21:31Z. [Online]. Available: https://trepo.tuni.fi/handle/10024/148362

[2] S. K. R. Gowrigari, "State Management in Web Components: Crafting Cohesive and Scalable Solutions," Oct. 2023. [Online]. Available: https://medium.com/@sudheer.gowrigari/state-management-in-web-components-crafting-cohesive-and-scalable-solutions-f4bbeb6c74d2

[3] "The Reactive Manifesto." [Online]. Available: https://www.reactivemanifesto.org/

[4] R. Carniato, “A Hands-on Introduction to Fine-Grained Reactivity,” Feb. 2021. [Online]. Available: https://dev.to/ryansolid/a-hands-on-introduction-to-fine-grained-reactivity-3ndf

[5] “SolidJS Documentation.” [Online]. Available: https://www.solidjs.com/docs/latest

[6] R. Carniato, “Finding Fine-Grained Reactive Programming.” [Online]. Available: https://angularindepth.com/posts/1269/finding-fine-grained-reactive-programming

[7] “Signals – Preact Guide.” [Online]. Available: https://preactjs.com/guide/v10/signals

[8] “signals/packages/core/src/index.ts at main · preactjs/signals.” [Online]. Available: https://github.com/preactjs/signals/blob/main/packages/core/src/index.ts

[9] R. Carniato, “The Evolution of Signals in JavaScript,” Feb. 2023. [Online]. Available: https://dev.to/this-is-learning/the-evolution-of-signals-in-javascript-8ob

[10] “Interactive Results.” [Online]. Available: https://krausest.github.io/js-framework-benchmark/current.html

[11] “solid/packages/solid/src/reactive/signal.ts at main · solidjs/solid.” [Online]. Available: https://github.com/solidjs/solid/blob/main/packages/solid/src/reactive/signal.ts

CDN Cache poisoning threaths

Markus Regardh
markus.regardh@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

As Content Delivery Networks (CDN) is increasingly becoming a staple in modern internet infrastructure, and is used by most of the worlds largest websites, their security and reliability has become an important part of internet security. CDNs can be targeted by a variety of attacks, targeting for example the availability of the website or personal information of users. One category of attacks on CDNs is cache poisoning, which is an attack that targets the cache, attempting to manipulate the cached content that gets sent to users. The vulnerability that leads to most cache poisoning attacks is unkeyed inputs, which means that the cache key as well as how a website handles headers it important for mitigation of these attacks.

KEYWORDS: CDN, Cache Poisoning, Denial-of-Service, cybersecurity

1 Introduction

Content delivery networks (CDN) have become a staple of modern internet infrastructure, being used by an estimated 74% of the top 1000 visited websites worldwide [5], and are key in providing users with fast and reliable content on the internet. However, their widespread use as well as

their important role in internet infrastructure also make their security and integrity of critical importance, and makes them targets for various security attacks [4].

One such attack is known as Cache Poisoning, which involves a malicious user managing to alter the cached content on a server, thus being able to manipulate what other users view and can access. Since CDNs are used by such a large amount of modern websites, an attacker being able to use the CDN to either misdirect or deny access to legitimate users can have devastating consequences for the integrity, reliability and confidentiality of both the CDN provider as well as the owner of the website. This paper aims to address this problem by taking a closer look into cache poisoning and the most common techniques as well as the underlying vulnerabilities that make them possible, and by conducting an experiment to gain insight on the current status of the efficiency of these techniques.

The paper is organized as follows. Section 2 covers CDNs and their role in modern web infrastructure. Section 3 dives deeper into cache poisoning attacks as well as different methods. The 4th section covers the experimentation together with the results, and section 6 wraps up the paper with conclusions.

2 Content Delivery Networks

A Content Delivery Network (CDN) is a geographically distributed network of servers that are used to cache and serve content close to the end users. The CDN distributes content from a main server to cache servers, called edge servers, which then store that content and can respond to requests from users with the cached content [4, 15]. Traditionally CDNs have been used to serve static content, but in recent years CDNs have been developed to support dynamic content as well [20]. As these edge servers are geographically located close to the end-user, a CDN infrastructure can serve users with minimal latency [15]. CDNs are also often optimized to cache a selective set of content, such as the most frequently fetched content, further helping in reducing bandwidth.

A content owner, which is a customer of the CDN, deploys content such as a website on an origin server, after which the CDN distributes the content

into multiple edge servers in various locations distrbuted geographically [4]. When an end user requests content, a request routing mechanism redirects the request to an edge server. The edge server then attempts to serve the request using its cache, and in case of a cache miss, meaning that the content was not available on the edge server, the edge server fetches the content from either the origin server or another edge server and stores it in its cache for future requests. There are three main ways of updating the cached content: a pull model where edge servers fetch content based on user requests, a push model where the content is distributed based on anticipated requests, and a hybrid model that combines the push and pull models.

Cache control HTTP headers are used by the CDN for managing the caching mechanism, specifying rules for the caching of content. The specifics of the cache control header vary from provider to provider, but the core values include "max-age", specifying how long to store the cached response, "public" specifying that a cached response can be used in a public cache, and "no-cache", specifying that a response should not be cached, used for example if the response contains information that is personalized [10]. In addition, most responses from a CDN includes additional headers such as "Age", specifying the age of the cached response, and a header specifying weather the response was a cache miss or a cache hit. These headers, while vital for the functionality of the cache, can also aid in the building of attacks targeting the cache [9].

As storing the entire HTTP request byte-for-byte in the cache server is inefficient and requires a lot of storage, edge servers usually make use cache keys to know how to respond to different requests by storing a mapping of a cache key to a cached response [9]. Cache keys only store specific parts of the request, usually the HTTP method and certain headers, depending on the service provider [4]. The inputs that are not part of the cache key, thus not considered by the caching mechanism, are called unkeyed inputs [4].

2.1 CDN security

Ghaznavi et al. [4] present a survey of security challenges relating to CDNs. The survey divides these security challenges into three main categories based on which part of the CDN architecture the vulnerability

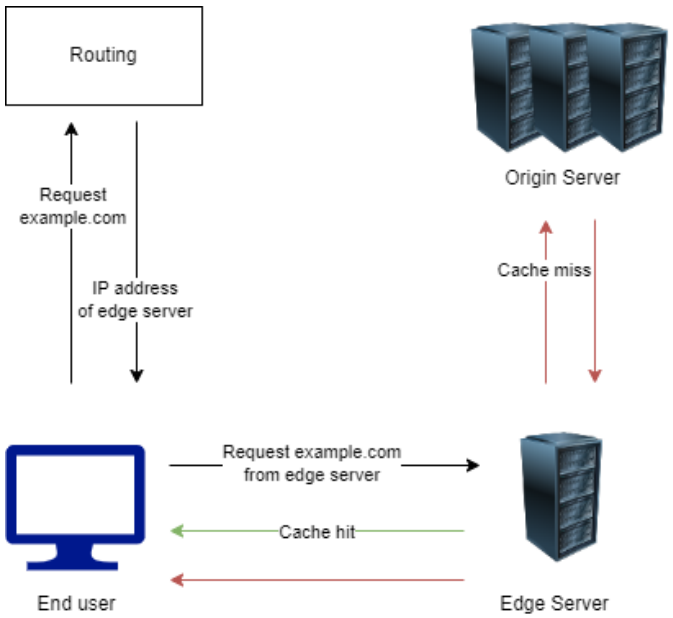


Figure 1. Overview of the architecture of a CDN. The routing mechanism directs the user to an edge server. The client then makes a request to an edge server. In case the edge server has the requested content, it is a cache hit, and it responds with the content. In case of a cache miss it fetches the content from the origin server, caches it for future requests, and serves the client the content

depends on: Edge servers, request routing, or origin servers. Some of the common challenges on edge servers are Denial of Service (DoS) attacks as well as cache attacks, targeting the end users. The techniques used to attack edge servers usually involves crafting a malicious request that breaks down the functionality of the cache. Request routing is also vulnerable to DoS attacks, as well as reconnaissance of the inner routing of the CDN system, which is often a prerequisite for further attacks towards the infrastructure of the CDN. Origin servers can be vulnerable to man-in-the-middle attacks, as well as attacks resulting in gaining information about the IP of the origin server, which can then be used to bypass the CDN, allowing for an attack directly on the origin server.

2.2 HTTP/HTTPS

As most browser targeted internet traffic is implemented with Hypertext Transfer Protocol (HTTP) or it's extension Hypertext Transfer Protocol Secure (HTTPS), CDNs naturally deal mostly with requests and responses using these protocols. The structure of a request, which can be seen in Figure 2, in both HTTP and HTTPS are identical, with the difference in the protocols being the additional layer of encryption added to HTTPS. The core structure of the protocols contains headers, followed by

the content itself, such as a HTML page [19]. A header is a key-value pair, containing additional information about a http request or response [11]. The first header is known as the request-line, specifying a HTTP method together with a path to a resource [19]. Most of the headers used are defined in a registry by iana (Internet Assigned Numbers Authority) [7], but headers can also be customized. One such header, which is often used in the context of reverse proxies, such as CDNs, is the X-Forwarded-Host header, which is used to identify the original host of a request [12].

2.3 HTTP related attacks

Attacks on CDNs are usually paired with other forms of attacks on web applications [9]. One example of such an attack is Cross-site scripting (XSS), which is an attack where an attacker manages to inject scripts on a web page, which once fetched by a legitimate user gets executed in the browser of the user. XSS-attacks can be used to for example impersonate the victim, gaining access to their data or perform the same actions that the victim can [16]. A famous example of a vulnerability resulting in XSS-attacks is when Twitter did not properly filter tweets, resulting in users being able to "tweet" Javascript code, which then got executed on the victims browsers when hovering their mouse on the tweet [1].

3 Cache Poisoning Attacks

Cache poisoning is when attackers target the integrity of the cache by replacing the intended cached response with a poisoned response, thus tricking the cache into serving the poisoned response to requests, containing harmful and malicious content [4]. These attacks are usually enabled by unkeyed inputs, which are the parts of the request that is not part of the cache key [4].

Figure 2 shows an example of a basic request made from a browser to fetch a website. A request from a browser contains a lot of data, which is inefficient and unnecessary to store in the cache. The standard values included in the cache key are colored with red. Figure 3 shows an almost identical request, with the difference in the cookie header colored in blue, which indicates that the user requests the website in Finnish. Since the header "Cookie" is not part of the cache key, the user will get the cached

```
GET /posts  HTTP/1.1
Host: example.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9...
Cookie: language=en
User-Agent: Chrome/121.0.0.0
Accept-Language: en-US
Accept-Encoding: gzip, deflate, br, zstd
```

Figure 2. Example request

```
GET /posts  HTTP/1.1
Host: example.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9...
Cookie: language=fi
User-Agent: Chrome/121.0.0.0
Accept-Language: en-US
Accept-Encoding: gzip, deflate, br, zstd
```

Figure 3. Example request

response of the request in Figure 1, in English, instead. A malicious user could use this vulnerability by getting the server to cache the response to a request containing malicious headers, such as an X-Forwarded-Host header, instead of the intended request, as visualized in figure 3. Additional headers can be added to the cache key, as has been done for example by Cloudflare [4]. However, the size of the cache key can impact the performance of the cache, making the selection of values to use as a cache key vital. The header could be used maliciously for example for an XSS attack, where you would insert your script into the value of the header [9].

The following sections will cover a variety of Cache Poisoning attacks that have been discovered to gain an overview of the attack landscape and the potential of cache poisoning.

3.1 Basic Poisoning

In the example in Figure 4, where a malicious user injects XSS into the cached response through the X-Forwarded-Host header, is defined in James Kettles research on cache poisoning [9] as basic poisoning. In his exper-

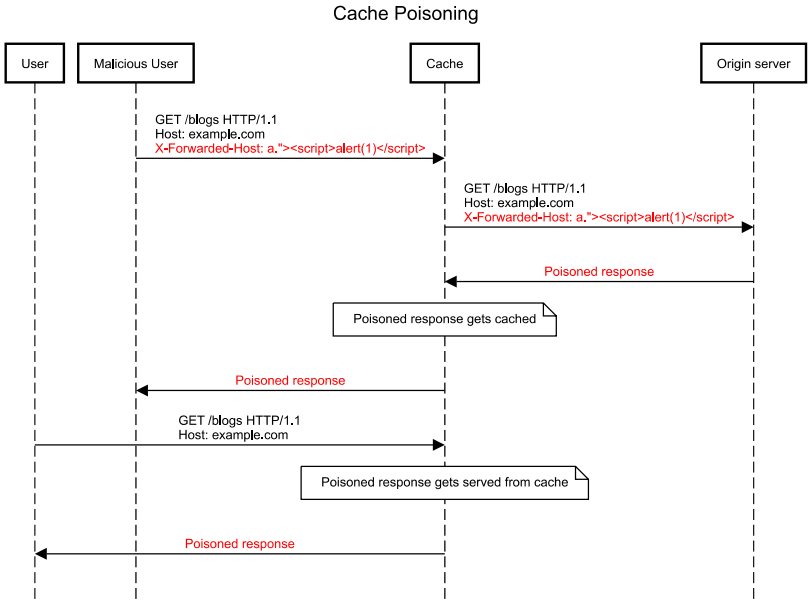


Figure 4. Cache poisoning attack

iments, he poisoned the cache to a non-existent endpoint, partly because he did not want to interrupt normal use of the website, and partly because he knew that that endpoint was not in the cache from before, guaranteeing that his malicious request gets stored in the cache.

To guarantee that a malicious request gets cached, a malicious user could continuously send the request and hope that one of the requests would then be the first one to be sent after the cache expires. However, another vulnerability that Kettle [9] found was that some responses contained the time of expiry of the cached response in the cache control header, and could abuse that to time his request. Kettle also found that some websites use the user-agent header as part of the cache key, allowing for targeted poisoning based on the browser used by clients, which could be used to exploit browser-specific vulnerabilities.

3.2 Cache-Poisoned Denail-of-Service

In their paper [14], Nguyen et al. do further research into constructing Cache-Poisoned Denial-of-Sevice Attacks, which they coined as CPDoS. A Denial-of-Service (DoS) attack is an attack where a malicious user attempts to disrupt the usage of a network or machine, thus making it unable to serve its users. The way they denied service was by getting the

cache to fetch error pages from the origin server, caching it, and serving those error pages to legitimate requests. They attempted three different relatively simple methods for the CPDoS attack, all through the request headers. One of the attacks was through the X-HTTP-Method-Override, which is used in some REST-based web services to circumvent restrictions on HTTP methods. Through this header, they could get the origin server to, for example, attempt to respond to a POST request to a request with the original method GET, resulting in an error, which then would get cached and served as a response to further GET requests. They also managed to make use of the possible inconsistencies in how the servers read and handle illegal headers. They created very large headers that were too large for the origin server, throwing an error, while still being small enough for the cache server to forward it and see it as a valid request. Similar results could be accomplished by sending the newline character '\n' in the header. The usefulness of these attacks relies heavily on both the implementation of the cache server and the underlying HTTP implementation of the origin server.

3.3 Web Cache Deception

While the aforementioned variations of cache poisoning has the goal of denying or limiting access to a resource, there are also variations that target individuals and their personal data. One such variation is Web Cache Deception (WCD), which involves poisoning a cache with sensitive and private content, making it available on the internet [13]. The attack involves a victim of social engineering, who opens up a link to a legitimate resource containing personal information, to which an invalid static file is appended, for example, "example.com/profile/invalid.css". The web cache forwards the request to the origin server, as the url with the ending invalid.css is not cached from earlier. As the static file does not exist on the origin server, it responds as if the request was made to the endpoint /profile, together with cache control headers stating that the response should not be cached, as it contains sensitive information. However, the cache might ignore the cache control headers, and instead cache the response as a .css file, resulting in the sensitive information now being available publicly at the original URL.

3.4 Overview

In addition to finding vulnerabilities that can lead to XSS attacks, Kettle [9, 8] also managed to find more intricate methods that could be used to hijack entire pages, get around firewalls and authentication, redirect the user to a URL of his liking, and even replace sources to download buttons on web pages. Also, in addition to making use of unkeyed headers, he found inconsistencies in how query strings are keyed, resulting in further vulnerabilities that led to him being able to send users to the wrong pages inside a host. While most, if not all, of the vulnerabilities demonstrated have since been patched on the targeted platforms, the success of the aforementioned attempts show that cache poisoning is a real threat and needs to be considered, both when designing CDN systems as well as deploying a website to a CDN.

As a CDN provider it is key to choose an appropriate cache-key to mitigate attacks that are based on the vulnerabilities of unkeyed inputs. Not caching "400 bad request" error-pages is also an simple, but effective, practise that mitigates the risks of CPDoS attacks [14]. Similarly, on the origin server, it is important to properly set the appropriate status code to error responses, as they can determine if a response should be cached or not by the cache server [14]. Using the cache to only cache static responses is also a good way to avoid attacks such as WCD [13]. Additionally, avoiding taking input from headers can mitigate the risks of falling victim to cache poisoning, however, it is important to consider the headers used by the underlying frameworks used in a website [9].

4 Cache poisoning in practise

This section will cover a practical experiment, conducted to gain insight into the current feasibility on cache poisoning on some current, publicly available CDNs.

4.1 Methology

To assure an ethical experiment and that no legitimate users were affected by my experiments, I rented out my own domain as well as CDNs from the CDN providers. On the domain I deployed a simple website. I

chose two CDN providers to do the tests on; Cloudflare [3], because of its position as one of the largest CDN providers, and bunnyCDN [2] which is a less known CDN provider. A common denominator with both CDN providers is that they have a free tier available, which was a requirement.

On each CDN, I started by analyzing the caching behaviour of the CDN by creating some basic requests and reading the response headers, which can include information such as the cache-key, and weather a response was cached or not. I then made an attempt to do a Basic Cache Poisoning attack, using the "X-Forwarded-Host" header. In addition to attempting a Basic Cache Poisoning, I used an open source burp [18] extension called Param Miner [17], that is created and used by James Kettle while doing his experiments in [9]. Param miner is used to identify hidden unkeyed parameters, which potentially can then be used to find vulnerabilities for cache poisoning. I then analyzed the unkeyed inputs, and attempted to use them to manipulate the cache. I also ran a scan using the open source Web Cache Vulnerability Scanner (wcvs) [6] which, similary as Param Miner, is a tool used to scan for unkeyed inputs. I also attempted CPDoS as described in section 3.2.

4.2 Results & Discussion

The experiment was unable to accomplish its aim of finding vulnerabilities in neither of the CDNs tested, as I was unable to find any unkeyed inputs that could be used for cache poisoning. While this shows that both of the chosen CDN providers are up to date on their security, drawing a definite conclusion on the status of cache poisoning will require further testing on a more comprehensive list of CDNs as well as testing beyond the most common parameters that param miner as well as wsvc tests. I also made several mistakes during initial setup which slowed down the progress of the experimentation, such as not realizing that the host platform of my website has a built in CDN that can not be disabled, forcing me to switch host to not get skewed results. The selection of CDNs was also suboptimal, especially Cloudflare, as I knew that they have patched these issues earlier [9, 4]. Another improvement would be to try to find more recent cases and studies covering techniques and vulnerabilities, as they would most likely have a higher chance of still existing.

5 Conclusion

CDN cache poisoning can have various implications for users, CDN providers as well as CDN customers. In its most basic form it impacts the service availability of a website, denying or limiting access to legitimate users. However, paired with other forms of cyber attacks, one can accomplish more devastating attacks on privacy, such as CPDoS [14]. However, with careful planning and by following best practises on both the CDN providers as well as the CDN customers side, cache poisoning is not impossible to mitigate. As the basis of most attacks is either in unkeyed headers or parameters, carefully considering the cache key as well as header usage can already mitigate a lot of attacks.

References

[1] Charles Arthur. Twitter users including sarah brown hit by malicious hacker attack. *The Guardian*, September 2010.

[2] Bunny Way, LLC. BunnyCDN. https://bunnycdn.com, 2024. Website.

[3] Cloudflare, Inc. Cloudflare. https://www.cloudflare.com, 2024. Website.

[4] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A. Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. Content delivery network security: A survey. *IEEE Communications Surveys Tutorials*, 23(4):2166–2190, 2021.

[5] Run Guo, Jianjun Chen, Baojun Liu, Jia Zhang, Chao Zhang, Haixin Duan, Tao Wan, Jian Jiang, Shuang Hao, and Yaoqi Jia. Abusing cdns for fun and profit: Security issues in cdns' origin validation. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 1–10. IEEE, 2018.

[6] Hackmanit. Web-cache-vulnerability-scanner. https://github.com/Hackmanit/Web-Cache-Vulnerability-Scanner?tab=readme-ov-file, 2024. Accessed 02.04.2024.

[7] iana. Hypertext transfer protocol (http) field name registry. https://www.iana.org/assignments/http-fields/http-fields.xhtml. Accessed 02.04.2024.

[8] Kettle James. Bypassing web cache poisoning countermeasures. *PostSwigger*, 2018.

[9] Kettle James. Practical web cache poisoning. *PostSwigger*, 2018.

[10] MDN Web Docs. Cache-control. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control. Accessed 02.04.2024.

[11] MDN Web Docs. Http headers. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Contro. Accessed 02.04.2024.

[12] MDN Web Docs. X-forwarded-host. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-Host. Accessed 02.04.2024.

[13] Seyed Ali Mirheidari, Matteo Golinelli, Kaan Onarlioglu, Engin Kirda, and Bruno Crispo. Web cache deception escalates! In *31st USENIX Security Symposium (USENIX Security 22)*, pages 179–196, Boston, MA, August 2022. USENIX Association.

[14] Hoai Viet Nguyen, Luigi Lo Iacono, and Hannes Federrath. Your cache has fallen: Cache-poisoned denial-of-service attack. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1915–1936, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Gang Peng. Cdn: Content distribution network. *arXiv preprint cs/0411069*, 2004.

[16] PortSwigger. Cross-site scripting. https://portswigger.net/web-security/cross-site-scripting. Accessed 02.04.2024.

[17] PortSwigger. Param miner. https://github.com/PortSwigger/param-miner, 2020. Accessed 02.04.2024.

[18] PortSwigger Ltd. Burp Suite. https://portswigger.net/burp, 2024. Accessed 02.04.2024.

[19] Tom Henderson. Understanding http requests: Structure, methods examples. https://www.linode.com/docs/guides/http-get-request/. Accessed 02.04.2024.

[20] Behrouz Zolfaghari, Gautam Srivastava, Swapnoneel Roy, Hamid R Nemati, Fatemeh Afghah, Takeshi Koshiba, Abolfazl Razi, Khodakhast Bibak, Pinaki Mitra, and Brijesh Kumar Rai. Content delivery networks: State of the art, trends, and future roadmap. *ACM Computing Surveys (CSUR)*, 53(2):1–34, 2020.

Beyond cookies: how web services track their users today

Mostafa Ghozal
mostafaashrafmostafaali.ghozal@aalto.fi

Tutor: Tuomas Aura

Abstract

Privacy issues on the internet are of considerable importance, especially with the introduction of new regulations like the General Data Protection Regulation (GDPR) the actions of major companies like Google in discontinuing certain tracking tools gradually. This paper looks into how websites track users online in the post-cookie era, focusing on a method called "browser fingerprinting," known for its covert nature. We conducted an analysis of various tracking tools and analyzed their functionality based on findings from existing research papers. Our findings show that browser fingerprinting is a tough challenge for privacy because it doesn't rely on things users can easily delete, like cookies. Even though there are some tools to help, like browser add-ons, they are not very effective. We advocate for collaborative efforts among researchers, developers, and policymakers to find better ways to protect people's privacy online. Educating users about these issues is also key. With everyone pitching in, we can make the internet safer for everyone without making it harder to use.

1 Introduction

Web services have always raised attention to a persistent dilemma: the delicate balance between optimizing the user experience and addressing user privacy concerns. This challenge has become even more pronounced with the introduction of new regulations like the General Data Protection Regulation (GDPR) and Google's initiative to phase out third-party cookies. These developments signal significant transformations in the landscape of online privacy and advertising. An experiment conducted by Google Ad Manager's serving system underscores the magnitude of this transition: a notable 52% decrease in average revenue was observed upon the elimination of cookies. This prompts advertising agencies to seek more rapid alternative solutions. [4] it is worth noting that the cessation of third-party cookies has faced multiple postponements since its announcement. This delay highlights the considerable challenges in implementation, especially considering the difficulty in blocking every tracking method while ensuring the best user experience. In light of these complexities, this paper aims to provide transparency regarding tracking techniques for users. By offering a overview of these methods and their collective impact, our goal is to shed light on the broader landscape of online tracking. Understanding how these techniques interact is crucial for empowering users to make informed decisions about their online privacy. The methodology employed in this paper involves a literature review of recent studies about the most commonly used advanced tracking tools to conclude answers. The paper is structured as follows:

In Section 2, we provide background information on advanced user tracking techniques. Section 3 outlines the problems faced by users and the goals of this paper. In Section 4, we discuss solutions and their effectiveness. Section 5 concludes the literature review and the experimental findings. Finally, Section 6 proposes potential future directions and improvements.

2 Background

This section provides background information on the tracking methods reviewed in this paper. Below, we provide a brief overview of these techniques:

2.1 Browser Fingerprinting

Browser fingerprinting is a stateless user tracking technique where client-side scripts discover device configuration details through JavaScript APIs. Unlike cookies, fingerprinting cannot be cleared locally as it operates without relying on local storage. [9]

The challenging part is some functionalities in the web services depends on the fingerprinting JavaScript APIs. In conclusion, fingerprinting is hard to block, and thus, its use is increasing. According to an experiment by Iqbal et al. [9], more than 10% of the top 100,000 websites employ this technique.

Notably, these techniques are interchangeable, allowing for a combined approach to target users effectively. Furthermore, browser changes do not alter the fingerprint, as it depends on the computer OS and hardware, ensuring persistence across different browsers. Browser fingerprinting comprises many techniques,including:

2.1.1 Canvas Fingerprinting

The canvas fingerprinting methodology operates by rendering a text drawing on a canvas and subsequently analyzing the resulting image to construct a unique fingerprint. This fingerprint is generated by a multitude of factors inherent to the rendered canvas, including font libraries, graphics card specifications, drivers, and browser settings. When amalgamated, these factors produce distinguishable characteristics such as text pixel patterns, smoothness, and other attributes.

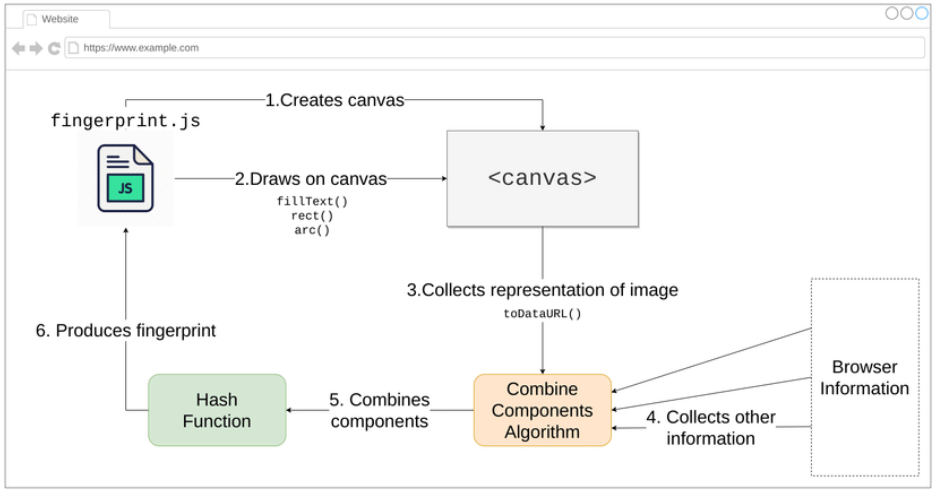


Figure 1. Process flow of canvas fingerprinting [2]

While canvas fingerprint has concerns for user privacy,it can be used in a constructive way. For example, according to Abouollo at al. [1] , it

can be used to detect fake accounts on social media by finding the same fingerprint on many user accounts. or if the user needs security updates.

2.1.2 WebRTC Fingerprinting

WebRTC (Web Real-Time Communication) was designed to support peer-to-peer communication between browsers using a microphone and camera. The fact that the browser accesses information such as screen resolution,operating system, and other system attributes is utilized by a JavaScript API called Fingerprintjs2 to identify a fingerprint . [14]

2.1.3 Ever-Cookies

Ever-cookies (sometimes referred to as zombie cookies), is a JavaScript API designed to utilize multiple storage methods for cookies, including flash cookies, local storage, and ETags. It detects user attempts to delete cookies and recreates them using alternative storage methods. [2] [15] Flash cookies, also known as local shared objects (LSOs), are data files created by Adobe Flash Player on a user's computer. They are stored separately in different location than the browser cookies, so it can be used to persist it. ETags, also known as entity tags, are part of the HTTP protocol and are used for web cache validation. They are unique identifiers assigned by web servers to versions of resources. Local storage is a web storage mechanism available in web browsers that allows web applications to store data locally within the user's web browser.

2.2 Cookie Synchronization

Cookie synchronization is a method that bypasses the Same-Origin Policy by exchanging information through a third-party server. This process allows trackers to collaborate to track user behavior across various websites. [12]

2.3 Search Engine Queries

Search Engine Queries: When users initiate search queries, search engines can employ third-party cookies to track user interactions, including clicks and visits originating from the search engine. This tracking mechanism allows search engines to collect information about user behavior, enabling them to tailor search results, refine user profiles, and personalize advertisements based on user interests and preferences. [8]

2.4 Social Media Tracking Tags

Social Media Tracking Tags are unique identifiers appended by social media platforms, such as Facebook Click ID (FBCLID), to outbound links shared on their platforms. BCLID, for instance, facilitates persistent tracking of user activities across websites integrated with Facebook Pixel. This tracking mechanism enables platforms to attribute user browsing activity to specific accounts [3]

2.5 Reliability of Opt-out Checks

As outlined by Duc Bui [5], opt-out prompts typically request user permission to allow cookies. However, in practice, discrepancies often arise between the stated opt-out policies and the actual implementation, leading to inconsistencies in user privacy management.

This paper primarily focuses on browser fingerprinting, providing insights into its mechanisms, implications, and potential countermeasures. The background information provided aims to equip readers with a broad understanding of prevalent tracking methods in contemporary online environments. Now that we have outlined the background information on advanced user tracking techniques, it is important to address the challenges faced by users and the objectives of our paper. Section 3 will elucidate these issues and articulate the goals of our research.

3 Problem

Browser fingerprinting poses a significant challenge to user privacy due to its ability to track users across websites without their explicit consent. Unlike traditional tracking methods, such as cookies, which users can easily delete or block, browser fingerprinting relies on various characteristics of the user's browser and device configuration, making it difficult to detect and mitigate. Many of the features and functionalities that contribute to browser fingerprinting are also essential for providing a seamless and personalized browsing experience. For example, JavaScript, which is commonly used by websites for interactive features and dynamic content, is also a key component in fingerprinting scripts. Disabling JavaScript entirely may enhance privacy but can severely limit the functionality of many websites, resulting in unacceptable user experience. Similarly,

other browser attributes such as user agent strings, screen resolution, installed fonts, and plugins contribute to fingerprinting but are also necessary for rendering web pages correctly and ensuring compatibility with various websites and web applications. Attempting to modify or obscure these attributes aggressively can lead to website rendering issues, broken functionality, or even outright blocking by websites that rely on these attributes for legitimate purposes. Furthermore, as soon as browser releases an update to protect user privacy, in parallel, there will be new emerging fingerprinting techniques. Having identified the challenges posed by browser fingerprinting and other tracking methods, the next logical step is to discuss potential solutions. Section 4 discusses existing countermeasures and evaluates their effectiveness in mitigating tracking attempts.

4 Solution

There is a gap in previous research and implementations between offering a good user experience and preventing tracking. This gap is reasonable. Referring to [10], we can investigate the currently available approaches, such as browser plugins and extensions, that perform one of the following actions:

Randomizing browser attributes and configurations to obfuscate the uniqueness of the fingerprint, thereby impeding the tracking attempts. For instance, plugins may introduce variations in user agent string, screen resolution, installed fonts, and other identifiable attributes, thereby thwarting the accuracy of fingerprinting techniques. Examples include FP-Block, FPGuard, and Canvas Defender.

Universal fingerprinting is an approach that involves standardizing or normalizing certain browser attributes across a wide user base, effectively reducing the distinctiveness of individual fingerprints. By pooling together data from numerous users and presenting a uniform profile to trackers, these plugins aim to render fingerprinting less effective and undermine attempts to track users based on unique browser configurations. UniGL is one of the examples for this plugins.

Blocking scripts or blocking APIs can be done by selectively disabling JavaScript or restricting access to specific browser features and APIs known to be exploited for tracking. These extensions aim to disrupt the execution of fingerprinting scripts and impede data collection by trackers. Examples

include NoScript, Privacy Badger, and Brave browser.

Firefox employs various methods to resist fingerprinting, including blocking certain browser features and reducing the amount of information exposed to websites. One specific aspect of this approach involves blocking access to the Canvas API, which is commonly used for fingerprinting purposes. Additionally, Firefox reduces the amount of information available in several attributes, such as user agent strings and screen dimensions, to make it more difficult for websites to uniquely identify users based on their browser characteristics.

While all these solutions exist,they have demonstrated limited effectiveness when subjected to testing and analysis. Therefore, there is still a gap in providing real protection for user privacy.

5 Conclusion

We have identified the resilience of these methods against traditional countermeasures. Despite the existence of browser plugins and extensions aimed at mitigating tracking efforts, our analysis suggests that current solutions may offer limited effectiveness, highlighting the persistent gap in providing robust protection for user privacy.

Moving forward, it is imperative for researchers, browser developers, policymakers, and industry stakeholders to collaborate in addressing the challenges posed by web tracking techniques. This collaboration should involve ongoing research and development efforts to enhance existing countermeasures and develop innovative approaches to safeguard user privacy while preserving the functionality and usability of web services.

Furthermore, user education and empowerment play a crucial role in navigating the complexities of online privacy. By raising awareness about tracking techniques and providing users with the tools and knowledge to protect their online activities, we can empower individuals to make informed decisions about their privacy and take control of their digital footprint.

In conclusion, while the challenges posed by web tracking are formidable, they are not insurmountable. With concerted efforts and collaborative initiatives, we can work towards a more privacy-respecting online environment that prioritizes both user experience and user privacy. In light of the insights gained from our analysis, it becomes apparent that collaborative

efforts are essential to address the challenges posed by web tracking techniques. Section 6 proposes potential future directions and improvements, emphasizing the need for ongoing research and development in the realm of digital privacy.

6 Future Work

In the field of digital privacy, the development of tracking methods that prioritize user confidentiality emerges as a significant area for future research. Developing innovative tracking techniques that uphold user privacy while still facilitating targeted advertising and personalized content delivery represents a crucial necessity. This includes researching decentralized tracking mechanisms, investigating ethical considerations about data consent, transparency, and data ownership. Moreover, evaluating the efficiency of existing privacy regulations, such as GDPR, and improving them.

Lastly, exploring alternative business models for online services to increase revenues while protecting user privacy. Through these explorations, researchers can contribute to enhancing the overall experience of users and fostering a more privacy-conscious digital ecosystem.

References

[1] Ahmed Abouollo and Sultan Almuhammadi. Detecting malicious user accounts using canvas fingerprint. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 358–361, 2017.

[2] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 674–689, New York, NY, USA, 2014. Association for Computing Machinery.

[3] Paschalis Bekos, Panagiotis Papadopoulos, Evangelos P. Markatos, and Nicolas Kourtellis. The hitchhiker's guide to Facebook web tracking with invisible pixels and click ids. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 2132–2143, New York, NY, USA, 2023. Association for Computing Machinery.

[4] Dino Bollinger. Analyzing cookies compliance with the GDPR. Master's thesis, ETH Zurich, 2021.

[5] Duc Bui, Brian Tang, and Kang G. Shin. Do opt-outs really opt me out? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 425–439, New York, NY, USA, 2022. Association for Computing Machinery.

[6] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. Cookie swap party: Abusing first-party cookies for web tracking. In *Proceedings of the Web Conference 2021*, WWW '21, page 2117–2129, New York, NY, USA, 2021. Association for Computing Machinery.

[7] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 289–299, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[8] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, and M.C. Schraefel. Network analysis of third party tracking: User exposure to tracking cookies through search. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 549–556, 2013.

[9] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1143–1161, 2021.

[10] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *ACM Trans. Web*, 14(2), apr 2020.

[11] Emmanouil Papadogiannakis, Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. User tracking in the post-cookie era: How websites bypass GDPR consent to track users. In *Proceedings of the Web Conference 2021*, WWW '21, page 2130–2141, New York, NY, USA, 2021. Association for Computing Machinery.

[12] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *The World Wide Web Conference*, WWW '19, page 1432–1442, New York, NY, USA, 2019. Association for Computing Machinery.

[13] J.S. Park and R. Sandhu. Secure cookies on the web. *IEEE Internet Computing*, 4(4):36–44, 2000.

[14] Andreas Reiter and Alexander Marsalek. WebRTC: your privacy is at risk. In *Proceedings of the Symposium on Applied Computing*, SAC '17, page 664–669, New York, NY, USA, 2017. Association for Computing Machinery.

[15] Ove Sörensen. Zombie-cookies: Case studies and mitigation. In *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, pages 321–326, 2013.

Robustness Assessment for ML systems

Muskaan khattak
muskaan.khattak@aalto.fi

Tutor: Samuel Marchal

Contents

Abstract

Machine learning (ML) systems are integral to various critical applications, from healthcare diagnostics to financial fraud detection. However, these systems are vulnerable to adversarial attacks, compromising their integrity and reliability. This paper provides a comprehensive overview of the types of attacks that ML systems face, including evasion, poisoning, and model inversion attacks. It emphasizes the importance of robustness in ML systems and explores various metrics used to evaluate their resilience, such as accuracy under attack, worst-case performance, and robustness margin. The paper also reviews tools designed to assess and enhance the robustness of ML models, notably the Adversarial Robustness Toolbox (ART) and Counterfit. Through detailed analysis, this study highlights the need for continuous advancements in defensive strategies to address the evolving nature of adversarial threats.

KEYWORDS: machine learning, robustness, adversarial attacks, evaluation metrics, security, adversarial robustness toolbox, counterfit

1 Introduction

Machine learning has revolutionized various sectors, from healthcare diagnostics to financial fraud detection. These systems, however, face numerous attacks that can manipulate their behaviour and compromise the integrity of their applications [6]. The critical integration of machine learning into various sectors requires a robust evaluation and assurance of system resilience.

While research into ML attacks like evasion and poisoning is extensive, [3] [1], consensus on a standardized robustness assessment framework still needs to be improved. The variety of machine learning applications demands a strategy that accounts for a broad spectrum of adversarial scenarios [5]. Additionally, the evolution of adversarial tactics alongside ML models underlines the need for constant evaluation to meet emerging security challenges.

This paper provides a structured evaluation of current robustness assessment tools and metrics in machine learning, analyzing their effectiveness. It examines the utility of these tools in determining the resilience

of ML systems to various attacks and their capacity to preserve fairness and interpretability [8]. The subsequent sections will detail the attacks ML systems face, the metrics for evaluating system robustness, the tools available for robustness assessment, and an analysis of these tools and metrics.

2 Attacks on Machine Learning Systems

Machine learning (ML) systems are vulnerable to attacks that undermine their security and effectiveness, particularly in critical applications requiring dependable and accurate predictions. These attacks, such as evasion, poisoning, and model inversion, exploit weaknesses in ML models to alter outcomes, breach privacy, or impair performance. Recognizing these threats is vital for creating ML systems that are robust and resistant to malicious activities. This section explores the characteristics of these attacks, their consequences on ML systems, and the approaches researchers have suggested to counter and mitigate their impacts.

2.1 Evasion Attacks

Evasion attacks are techniques used to manipulate the outcome of an ML system by giving it malicious input. These attacks provide subtly modified input to deceive the model into making incorrect predictions or classifications. The inputs can have the following alterations: added noise, modified pixel values, or minor deviations imperceptible to human observations but are significant enough to cause misclassification.

Evasion attacks add difficulty in deploying machine learning systems in security-critical applications, including malware detection and fraud detection. They can significantly compromise critical decision-making scenarios when targeting ML models. For example, in autonomous driving, an evasion attack could manipulate input data (e.g., road signs) to cause the vehicle to misinterpret its surroundings, potentially leading to accidents. The ease of performing these attacks depends on the attacker's knowledge of the model and the complexity of the input space.

Various researchers proposed solutions to enhance the robustness of the ML system against these attacks, including adversarial training, input sanitization, and model assembling. Madry was one such researcher who proposed adversarial training in [6] as a method to improve the ro-

bustness of neural networks against these types of attacks.

2.2 Poisoning Attacks

Poisoning Attacks manipulate the training data to compromise the performance or to cause the model to behave maliciously. These attacks include using incorrect samples or modifying the actual training data to subsequently influence the learning process, which causes either the model to provide biased outputs or runtime exploitable vulnerabilities.

Poisoning attacks target various ML models, including recommender systems and classifiers. In financial systems, a poisoning attack could lead to incorrect risk assessments, resulting in significant economic losses. These attacks require access to the training data, which can be a limiting factor for attackers. However, poisoning attacks can be more feasible where data is crowdsourced or not rigorously controlled.

Strategies used to reduce these attacks include data sanitization, robust model training methods, and anomaly detection algorithms. Researchers such as Biggio define some such attacks in [1]. Gao in [2] proposed a defence mechanism called STRIP (STRong Intentional Perturbation) against Trojan attacks caused by poisoning by finding and neutralizing the effect of the poisoned data during the training phase.

2.3 Model Inversion Attacks

Inversion attacks use the model's output in queries to gain insight about its parameters. The attacker can infer private attributes and characteristics used in the model by iteratively refining the queries using the model's prediction.

Model inversion attacks target privacy-preserving machine learning systems, especially systems where sensitive data, such as medical data, is involved. Shokri in [13] talks about model inversion attacks as part of a broader study of privacy attacks in machine learning systems. For instance, in healthcare, attackers could use these attacks to infer private patient information from a model trained on medical records. This could lead to privacy breaches and violation of confidentiality agreements. The difficulty of performing these attacks varies based on the complexity of the model and the amount of information available to the attacker. In some cases, even limited access to model outputs can be enough to infer sensitive information.

3 Evaluation Metrics

In developing robust machine learning (ML) systems, various evaluation metrics are crucial for gauging their resilience and effectiveness. These metrics provide a comprehensive perspective on the models' ability to endure adversarial challenges, navigate extreme situations, and sustain dependability across varied environments. From assessing the models' performance under targeted attacks to their stability amidst data distortions, each metric sheds light on different aspects of the ML systems' robustness. This section explores several key metrics, including accuracy under attack, worst-case performance, robustness margin, attack success rate, model recovery time, generalization under distribution shift, and resilience to data corruption, collectively contributing to the overall assessment of machine learning systems' robustness and reliability.

3.1 Accuracy under Attack

Accuracy under attack assesses how well a model performs when given adversarial inputs designed to mislead it. By testing the model with such attacks, we gain valuable insights regarding its resilience against evasion attacks and its ability to maintain accurate predictions despite facing maliciously crafted inputs. It is especially crucial for applications where security and trustworthiness are essential, such as cybersecurity, autonomous driving, and medical diagnosis. Hence, models with high accuracy under attack are more likely to exhibit robust behaviour and can handle unforeseen challenges and potential threats. The importance of accuracy under attack is also explained in [15]

3.2 Worst-Case Performance

Examining the worst-case performance of machine learning systems offers invaluable insights regarding their resilience and dependability in various demanding scenarios. Unlike conventional evaluations, which focus on average conditions, this metric explores the model's behaviour under extreme circumstances, stretching its capabilities. Scenarios such as encountering adversarial inputs mentioned above exploit weaknesses in the model's decision-making process or navigating environments with high noise levels, uncertainty, or data corruption. Understanding the model's performance under such adverse conditions is essential in iden-

tifying potential vulnerabilities and weaknesses that may reduce its effectiveness in real-world deployments. Furthermore, it helps the development of robustness-enhancing techniques and strategies to boost the model's resilience. By comprehensively assessing worst-case scenarios, we can ensure that machine learning systems are more dependable in performance even when faced with unforeseen challenges. The importance of evaluating worst-case performance is analyzed in [16]

3.3 Robustness Margin

The Robustness Margin measures the slightest change needed to affect a model's prediction, indicating how sensitive the model is to input changes. For instance, in an image classification neural network, the robustness margin can be the least amount of noise added to an image, resulting in a different predicted class. A more considerable robustness margin means the model is more robust. Techniques like DeepFool, [7] are often used to calculate this metric and pinpoint the model's weaknesses to the slightest input shift.

3.4 Attack Success Rate

The Attack Success Rate measures the percentage of successful adversarial attacks against a model, indicating its vulnerability to malicious inputs. For instance, this metric would analyze how often an attacker can craft emails incorrectly classified as non-spam in a spam detection system. A high attack success rate indicates the need for improved defensive mechanisms. [10] demonstrated the importance of this metric in analyzing the limitations of deep learning models in adversarial settings.

3.5 Model Recovery Time

Model Recovery Time assesses a model's duration to regain its performance after an attack or failure. This metric is vital for understanding the resilience of systems in dynamic environments. For example, in a recommender system compromised by a data poisoning attack, the recovery time would measure how quickly the system can restore its recommended accuracy. Studies like [14] have explored certified defences to reduce recovery time and enhance model resilience against data poisoning attacks.

3.6 Generalization Under Distribution Shift

Generalization Under Distribution Shift evaluates a model's performance on data with a different distribution than its training set, highlighting its flexibility in new environments. For instance, a weather prediction model trained on data from one geographic region should maintain accuracy when applied to data from another area. This metric is vital for ensuring the reliability of machine learning models across diverse scenarios. The concept is extensively discussed in [11], which provides insights into managing and mitigating the effects of distribution shifts.

3.7 Resilience to Data Corruption

Resilience to Data Corruption measures a model's ability to maintain performance when trained or tested on corrupted data, reflecting its robustness to data quality issues. An example is evaluating a speech recognition system's accuracy when exposed to audio recordings with different background noise levels. A resilient model would demonstrate consistent performance despite the presence of data corruption. [4] has benchmarked the robustness of neural networks to common corruptions and perturbations, underscoring the importance of this metric in assessing model reliability.

4 Tools Used to Assess and Quantify Robustness on ML Systems

In machine learning (ML), safeguarding systems against adversarial attacks is essential for their security and dependability. Researchers and developers have devised various tools to evaluate and measure the robustness of ML systems, providing insights into their weaknesses and facilitating the development of robust defence strategies. This section looks into some of these tools, including the Adversarial Robustness Toolbox (ART), Counterfit, CleverHans, and Foolbox, each providing distinct functionalities and advantages to boost the security of ML models.

4.1 Adversarial Robustness Toolbox

The Adversarial Robustness Toolbox (ART) aims to increase the security of machine learning (ML) systems against adversarial attacks. With various functionalities, ART serves as a comprehensive suite of tools tailored

to evaluate and enhance the resilience of ML models. Supporting multiple machine learning frameworks and techniques, ART enables researchers and practitioners to generate adversarial examples, probe model vulnerabilities, and implement robust defence mechanisms [8]. Installing and using ART is simple, ensuring smooth integration into current ML workflows. Its user-friendly interface allows researchers and practitioners to easily set up assessments by defining model architectures, datasets, and parameters [10]. With these configurations in place, ART enables users to evaluate ML models comprehensively, offering valuable insights into potential vulnerabilities and weaknesses.

ART goes beyond mere assessment, providing practical solutions to address security concerns in ML systems. By finding vulnerabilities and weaknesses, ART empowers users to implement tailored defence mechanisms or refine their models to enhance resilience against adversarial attacks [3]. One standard feature of ART is its ability to generate adversarial inputs designed to cause ML models to produce incorrect outputs. ART can uncover vulnerabilities such as susceptibility to evasion attacks or data poisoning by analyzing how models react to these adversarial inputs. Furthermore, ART facilitates sensitivity analysis, allowing users to evaluate model performance across input distributions or feature perturbations. This comprehensive approach enables ART to detect subtle vulnerabilities that may not be apparent during standard testing procedures.

Overall, ART is a valuable tool for identifying and understanding vulnerabilities in ML systems. It provides researchers and practitioners with actionable insights to enhance their models' robustness and security.

4.2 Counterfit

Microsoft released Counterfit as an open-source initiative designed to automate the security testing of AI systems. It marks a significant advancement in AI security risk assessments, addressing the pressing need for practical tools in securing ML systems—a need highlighted by Microsoft's survey [5]. Its flexibility stands out as it operates across diverse environments, including cloud-based, on-premises, and edge scenarios. Counterfit's model-agnostic and data-agnostic capabilities ensure that it can effortlessly handle various ML models and data types, from text and images to generic inputs, facilitating a broad spectrum of security assessments.

Counterfit comes equipped with a wide array of features for compre-

hensive security analysis. It offers a command-line interface that streamlines the process of simulating adversarial attacks, managing and launching them across ML systems. Preloaded with extensive attack algorithms, Counterfit enables penetration testing and comprehensive vulnerability scanning, empowering professionals to efficiently identify and eliminate potential threats. Furthermore, its integration with the adversarial ML threat matrix, developed in collaboration between MITRE and Microsoft, enhances its utility in orienting security analysts to ML-specific threats. Counterfit's ability to conduct vulnerability scanning with customizable parameters and its logging feature for attack analysis significantly contribute to improving the understanding and fortification of ML systems against adversarial threats, aligning with Microsoft's responsible AI principles and the overarching goal of developing robust, reliable AI applications [5].

4.3 Other Tools and Libraries

CleverHans and Foolbox are also integral tools in adversarial machine learning, providing robust capabilities for testing and strengthening machine learning models' security. CleverHans offers various features for generating adversarial examples and assessing model vulnerabilities, promoting secure ML classifiers through community-engaged research [9]. Complementing this, Foolbox presents a rich selection of adversarial attack techniques compatible with various deep learning frameworks, featuring a user-friendly API that simplifies the implementation of complex attack strategies and robustness assessments [12]. Together, these tools form an essential resource for the machine learning community, facilitating the progression towards AI systems that can reliably resist adversarial threats and maintain trustworthiness in many deployment scenarios.

5 Analysis

Protecting machine learning systems from adversarial threats is critical for their reliable operation in essential applications. Developers have created tools to assess and reinforce the security of these systems. Table 1 catalogues these tools, showing the metrics computed, machine learning models covered, data types managed, and attack simulations undertaken.

Tool	Metrics Computed	ML Models Covered	Data Types	Attacks Simulated
ART	Accuracy under Attack (Adversarial Accuracy) Robustness Margin Attack Success Rate	Neural networks decision trees	Images text	Evasion poisoning
Counterfit	Attack Success Rate Robustness Margin	Generic ML models	Text images generic inputs	Evasion poisoning
CleverHans	Accuracy under Attack (Adversarial Accuracy) Attack Success Rate Robustness Margin	Primarily neural networks	Images, text	Evasion more advanced adversarial techniques
Foolbox	Accuracy under Attack (Adversarial Accuracy) Robustness Margin Attack Success Rate Generalization Under Distribution Shift	Various, including deep learning models	Images audio text	Broad range of adversarial attacks including evasion, poisoning, and others

Table 1. Comparison of Robustness Assessment Tools

5.1 Analysing Existing Tools and Techniques

Given the increasing reliance on machine learning (ML) systems in various critical applications, the robustness of these systems is a paramount concern. To address this concern, developers have created a wide array of tools, techniques, and metrics to assess and strengthen ML models against adversarial attacks.

The Adversarial Robustness Toolbox (ART) and Counterfit are notable tools in this domain. ART provides a comprehensive suite for evaluating ML models against adversarial attacks, including evasion, poisoning, and model inversion. It supports multiple ML frameworks and offers functionalities for generating adversarial examples, probing model vulnerabilities, and implementing defence mechanisms. Counterfit, on the other hand, automates the security testing of AI systems across different environments and is model-agnostic, making it highly versatile.

Additionally, CleverHans and Foolbox are essential tools in adversarial machine learning. CleverHans is known for its ability to generate adversarial examples and assess model vulnerabilities. At the same time, Foolbox offers various adversarial attack techniques and is compatible with various deep learning frameworks.

Metrics such as accuracy under attack, robustness margin, and attack success rate are crucial for quantifying the resilience of ML models. These metrics provide insights into the model's performance in adversar-

ial scenarios, its sensitivity to input perturbations, and its vulnerability to specific types of attacks.

5.2 Evaluation of Their Effectiveness

The type of ML model, the nature of the data, and the particular adversarial scenario being considered all play a role in determining the effectiveness of these tools, techniques, and metrics.

ART's ability to support multiple ML frameworks and data types makes it a valuable tool for assessing the robustness of diverse models. Its functionalities for generating adversarial examples and implementing defence mechanisms provide a comprehensive approach to evaluating and enhancing model resilience. The metrics computed using ART, such as accuracy under attack and robustness margin, offer quantifiable measures of the model's robustness.

Counterfit's flexibility and ease of use make it an effective tool for conducting security assessments across different environments and ML models. Its integration with the adversarial ML threat matrix aids in identifying and addressing ML-specific threats. It also enables the computation of the attack success rate metric, clearly indicating a model's vulnerability to adversarial attacks.

While the existing tools, techniques, and metrics offer valuable means for assessing and enhancing the robustness of ML systems, their effectiveness is limited. Continuous research, development, and adaptation of these tools, techniques, and metrics are necessary to ensure their relevance and efficacy in the face of evolving adversarial threats.

6 Conclusion

This paper has delved into the critical issue of ensuring the robustness of machine learning (ML) systems, focusing on essential metrics such as accuracy under attack and robustness margin. These measures are crucial for evaluating the resilience of ML systems, particularly in sectors where reliability is paramount.

An in-depth examination of tools like the Adversarial Robustness Toolbox and Counterfit has been conducted. It reveals that while these tools provide valuable insights into system vulnerabilities, the ever-evolving nature of adversarial threats requires ongoing advancements in our de-

fensive strategies. As we bolster the security of ML systems, we must also maintain their transparency and fairness.

n conclusion, this paper has provided a detailed evaluation of the robustness of machine learning (ML) systems. The findings emphasize the importance of a collaborative and proactive approach in enhancing the security and reliability of ML systems, ensuring their trustworthiness as they become increasingly integrated into various aspects of daily life.

References

[1] B. Biggio, B. Nelson, and P. Laskov. "poisoning attacks against support vector machines". In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 1467–1474, 2012.

[2] L. Gao, Y. Chen, R. Liao, Y. Li, X. Wen, and D. Shen. "strip: A defence against trojan attacks on deep neural networks". In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] I. Goodfellow, J. Shlens, and C. Szegedy. "explaining and harnessing adversarial examples". In *International Conference on Learning Representations (ICLR)*, 2015.

[4] D. Hendrycks and T. G. Dietterich. "benchmarking neural network robustness to common corruptions and perturbations". In *International Conference on Learning Representations (ICLR)*, 2019.

[5] R. S. Siva Kumar. "ai security risk assessment using counterfit". Available: https://www.microsoft.com/security/blog/2021/05/03/ai-security-risk-assessment-using-counterfit/, 2021.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. "towards deep learning models resistant to adversarial attacks". In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2018.

[7] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. "deepfool: A simple and accurate method to fool deep neural networks". In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[8] M.-I. Nicolae, M. Sinn, M. Tran, B. Buesser, A. Rawat, and M. Wistuba. "adversarial robustness toolbox v1.0.0". *arXiv preprint arXiv:1807.01069*, 2018.

[9] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, and A. Roy. "cleverhans v2.1.0: An adversarial machine learning library". Available: https://github.com/tensorflow/cleverhans, 2018.

[10] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. "the limitations of deep learning in adversarial settings". In *Proc. of the IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387, 2016.

[11] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors. *"Dataset Shift in Machine Learning"*. The MIT Press, 2009.

[12] J. Rauber, W. Brendel, and M. Bethge. "foolbox: A python toolbox to benchmark the robustness of machine learning models". Available: https://github.com/bethgelab/foolbox, 2017.

[13] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. "membership inference attacks against machine learning models". In *Proc. of the IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017.

[14] J. Steinhardt, P. W. Koh, and P. Liang. "certified defenses for data poisoning attacks". In *Advances in Neural Information Processing Systems (NIPS)*, pages 3517–3529, 2017.

[15] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. "ensemble adversarial training: Attacks and defenses". In *International Conference on Learning Representations (ICLR)*, 2018.

[16] Hongge Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Power and energy aspects of sustainable large-scale computing

Nhut Cao
nhut.cao@aalto.fi

Tutor: Vesa Hirvisalo

Abstract

The large-scale computing concept has been known for a period of time, along with its advantages. Comes with the powerful performance, large-scale computing, also considered as large-scale data center, consumes significantly amount of energy to support powering and cooling machines. The growing demand for computing services brought the concerns about the environmental impacts. This paper discusses cloud, fog, and edge computing and their connections with large-scale data center. The performances and energy consumption were analyzed. The paper also suggests a few potential strategies to help reducing carbon footprint and saving energy, while continuingly fostering advancements in technology.

KEYWORDS: Cloud, Fog, Edge computing, Sustainable energy, Data centers

1 Introduction

In recent years, network, computing, and the Internet have integrated to our lives in significant ways. With billions of users and devices actively perform computational executions, it is necessary that a more sophisticated, effective infrastructure be available to support. Cloud computing

has been one of the main platform providing services thanks to several large-scale data centers support. Barroso's book [3] discusses the design and operation of the powerful infrastructure, and the advantages and disadvantages of large-scale data centers.

There is a recent trend of pushing computing calculations to the edge of the network to reduce the data sent to the cloud and therefore address cloud computing issues, such as latency, transmission costs. Such computing paradigms, however, still pose several challenges, including service placement issue [7], that affect the power and energy consumption.

Usage of computing resources has increased dramatically, which has led to concerns regarding of the energy consumption and the effects to the environment. People are more aware of the carbon emitted from applications, products or services in daily lives, and computing electricity usage is also a salient subject. In this case, large-scale data centers that are backbones of modern computing paradigms account for significantly electricity usage, which raises concerns regarding the emissions to the ecosystem. To reduce carbon emissions from large-scale data centers, companies also seek for optimization in production and maintenance costs. Among giant tech companies, Google [18] published their "methodology and principles behind Google's system for Carbon-Intelligent Compute management", claiming their attempts to minimize environmental impacts.

This paper delves into the power and energy dynamics of sustainable large-scale computing by conducting a comprehensive literature review. It explores the impact of computing paradigms and large-scale data centers on energy consumption and the environment, analyzing their roles and effects in-depth. The paper also reviews several optimistic suggestions for designing and operating energy-aware large-scale computers, ultimately leading to a reduction in their carbon footprint.

The rest of this paper is organized as follows. Section 2 outlines the foundation concepts of cloud and other computing paradigms. Section 3 addresses the energy and carbon emission aspects of large-scale computing. Section 4 focuses on the potential solutions to minimize environmental damages as well as operation costs. Section 5 concludes the work of literature review.

2 Background

This part focuses on fundamental details concerning cloud computing, fog computing, and edge computing. It also covers the topic of large-scale data centers and their associated energy consumption challenges.

2.1 Cloud computing

In recent years, *cloud computing* has gained significant attention among people who care more about the optimization of cost and scalability. Cloud-computing is generally defined as an on-demand service, and users can obtain computer capabilities and technology services from a cloud provider without maintenance responsibilities.

Cloud computing architectures are characterized by their multi-layered structure and dependence on virtualization technologies. Vaguero [23] et al. has provided an overview of cloud computing definitions that are categorized based on cloud services: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

One of the most common services of cloud is data storage, which offers users virtual storage that is easy to access and manipulate. Google Drive or OneDrive from Microsoft are two common examples of cloud storage services with affordable subscriptions, from free to paid plans with prices adjusted to desired size of storage. Beyond the convenience of storing and managing data in the cloud, there are several advantages associated with cloud technology across various domains. In [12], Jadeja and Modi presented that cloud computing provides considerable features, including cost reduction thanks to its scalability and pay-per-use characteristics, globally accessible services, and efficient management as the facilities maintenance is less complicated.

In addition to noticeable advantages, cloud computing, however, brings the concerns of privacy and security. Cloud providers claim that they have robust mechanisms and are reliable partners that safeguard customers' data, but corporate users still hesitate since their important data may be stored in the same cloud storage with their competitors. Moreover, the possibility of attacks from service providers is also considered, which can lead to data lost or unwanted changes. These also pose security threats, as hackers tend to prefer using cloud to host running botnets with more affordable prices [6].

The performance of cloud computing is remarkable; however, with the

growing of users and data generated, cloud resources are required to address geographic and high-bandwidth, low-latency issues [25]. To solve these problems, fog and edge computing were introduced.

2.2 Fog and Edge Computing

Together with cloud computing, there are several computing paradigms that were introduced in recent years. This paper concentrates on three popular concepts: cloud computing, fog computing, and edge computing. In the previous section, this paper discussed the foundation background of cloud computing. Subsequently, this paper focuses on fog and edge computing concepts.

Fog computing, which was introduced by Cisco in 2012, extends the cloud computing services to the edge of the network [4]. According to OpenFog Consortium [1], "Fog computing is a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things", which helps escalating the "velocity of decision making". Hence, fog computing acts as an intermediary layer between traditional cloud services and end devices, reducing reliance on centralized cloud resources, and enabling faster decision-making at the edge of the network due to its distributed processing capabilities.

While cloud computing is more focused on globalized services, fog computing offers generous amount of nodes that are more localized. Latency issues are reduced with fog computing as its nodes can be placed near the source nodes [4, 25]. This is also a critical factor to distinguish between cloud and fog computing. With location advantage, fog computing provides high-quality streaming delivery service with proxies and access points [4]. For example, devices includes switch, router can be deployed as a node. Those devices can be easily deployed regardless of the location, as long as the network connection is available [7].

Yi et al. introduces the three-layer architecture formed by end users, fog, and cloud, which provides service delivery and support several applications, such as big data analysis, web content delivery [24].

Another interesting computing paradigm is edge computing, which operates at the edge of the Internet. Similar to fog computing, edge computing can handle latency and connectivity issues since it is located very close to the IoT devices. It transitions cloud computing resources and storage

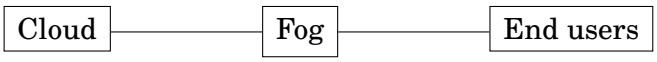


Figure 1. Three-layer hierarchical structure

to the edge of the network, which thus offers services address the requirements of data optimizing, security and privacy [5].

Figure 2. Three-layer hierarchical structure + Edge

In general, the difference between the two paradigms is their scopes and functionalities. Fog computing is defined to have a proper architecture, and it offers networking, storage services, as well as management and escalation from cloud to end users [1], whereas Edge computing is more likely limited to computing at the edge. With this mechanism, data can be processed without sent to cloud center, which helps reduce network bandwidth as well as the energy consumption from computing devices.

Although fog and edge computing have brought significant improvements, the role of cloud computing is incommutable. Fog and edge computing are present to share the burden of cloud, and help manage the jobs align with their scopes. Shi et al. [22] defined "edge as any computing and network resources along the path between data sources and cloud data centers".

2.3 Large-scale Data center

The expansion of cloud technology has been substantial, which unequivocal requires a robust and resilient infrastructure. Data center have been a critical element in providing innovation services of cloud computing. With millions of cloud executions, cloud providers started to invest in hyperscale data centers providing servers that support cloud computing. In general, large-scale data centers are hosts to cloud computing, which is responsible for distributing large amount of data for applications, providing scalable and effective environments to support computing operations.

These large-scale data centers located in various locations from Asia

to Europe to North America assist these companies with the flexibility to handle failures of servers and continue to provide stable services [19]. In a data center, multiple different computing systems can co-exist while remain unobstructed. According to Barroso et al. [3], hyper-scale data centers are significantly contrast to normal data centers: they use "a relatively homogeneous hardware and system software platform, and share a common systems management layer". Moreover, large-scale data centers are usually from renowned organizations, such as Amazon, Google, Meta, or Microsoft.

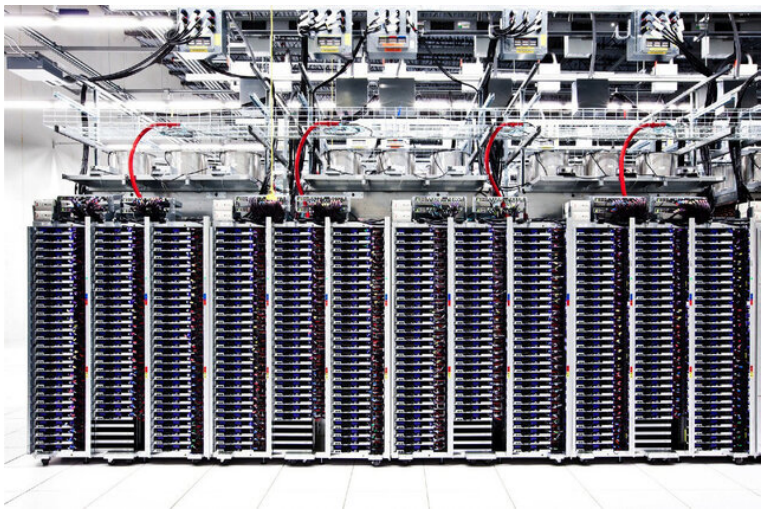


Figure 3. Server racks from Google WSC [11]

Due to the considerable size, data centers' operations consume a significantly amount of electricity. Koomey [14] reported that energy consumption in hyper-scale data centers costs the second-highest in operation expenses. The energy consumption and CO2 emission from large-scale data centers are particularly concerning, especially in recent years since global warming is a growing challenge, and more people pay attention to sustainable development.

3 Energy usage and carbon emission

As discussed, electricity usage of these large-scale data centers results in substantial cost of energy, and carbon emission to the atmosphere. Barroso et al. [3] stated that data centers account for "twice as much energy as is needed" solely for operating. Greenpace [8] reported that data centers' energy consumption could reach up to 1012 billion kWh by 2020, marking a threefold increase from their energy usage in 2007.

Several metrics are available for measuring the electricity usage, and

one common metric is Power Usage Effectiveness (PUE) [2], which is the ratio of the total energy used by a data center to the energy delivered to IT equipment. Typically, a data center will have a PUE of 2.0, and it is expected to reduce when the data center has efficient energy infrastructure.

When conducting electricity to powering the servers, electricity losses are inevitable. However, cooling systems are reported to responsible for major energy usage. Figure 4 shows that the chillers can lose energy approximately three times greater than conducting power losses, approximately 25% of power losses [3].

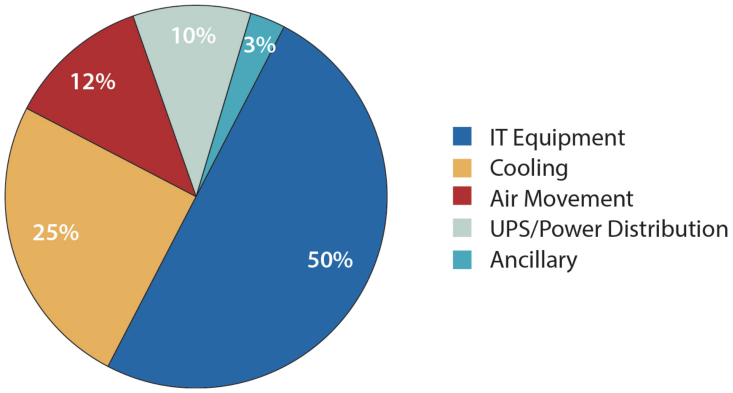


Figure 4. Power losses in a traditional (legacy) data center [3].

Radovanović et al. [18] indicated that data centers have to enable "efficiency of IT, cooling and power supply altogether" to avoid energy inefficiency. By optimizing cooling system power usage, the PUE can drop to under 1.50. Although this is a useful criterion to assess if a data center uses energy efficiently, PUE is not well adapted to the dynamic changes of the nature [10]. Therefore, PUE value is suitable for reference and estimate purposes rather than considered as a strict standard to evaluate energy-aware data centers.

In addition to the power usage of cloud computing in large-scale data centers, fog computing also has the possibility to contribute to the rise of energy consumption. In [21], Sarkar studied that in fog computing, the number of requests needs to be redirected to the cloud computing is linearly correlated to the energy consumption. Moreover, the service placement problem in fog and edge computing influence the power usage due to several factors, including network traffic and computation load. Tasks are not effectively distributed, such as placing intensive ones closer to the edge, can result in the energy-hungry scenario, since energy is allocated for processing and cooling. Since distances impact energy usage and node

energy efficiency varies, the positioning of computing resources holds significance. As mentioned earlier, service placement refers to the strategic allocation of computational tasks to fog nodes. This placement significantly influences power usage due to factors such as network traffic and workload distribution. Salaht et al. and Gasmi et al. provide in-depth discussions on this crucial aspect in [7, 20].

The carbon emission from data centers is also a major matter. Reliance on fossil fuels for electricity generation significantly drives carbon emissions, contributing to a harmful environmental impact. According to The Climate Group [15], data centers globally emitted a significant amount of carbon dioxide (CO_2) in 2007, releasing 116 million metric tons ($MtCO_2$). Their report further suggests a potential rise in these emissions to 259 $MtCO_2$ by 2020, even with the recent advancements in data center technology such as virtualization, cooling systems, and power supply efficiency.

The calculation of carbon emissions involves determining the carbon emissions linked to each energy source in addition to the overall energy consumption. With this method, ones can also predict the carbon footprint by the pattern of energy usage. Google's data centers have a one-day forecast of resource usage to utilize energy distribution, therefore reducing costs and carbon footprint [18]. However, this can be difficult because of the uncertainty. Factors such as fluctuating user demand, shifting application workloads, and unexpected hardware failures, all of which can disrupt energy consumption patterns despite efforts to forecast resource usage.

The energy usage, operating costs, and emissions have affected not only these top-tie technology companies, but also to the ecosystem. Despite the significant strides in technological innovation, the resulting environmental damages remain a pressing concern. To address the growing demand, the focus should be on designing and building large-scale data centers that are highly energy efficient.

4 Discussion

In the world where technologies are significantly growing yet accompanied by substantial environmental damages, humans raise a question regarding the preservation of current conditions, and the possibility of healing the wounds that were industrially inflicted.

Efforts to address these challenges require comprehensive strategies that balance technological advancement with environmental sustainability. The path to a sustainable future demands collaborative efforts from industries, governments, and communities. Several researchers and scientists have been doing research and developments to minimize energy consumption of large-scale computing, and maximize the sustainability in the industry.

In the context of carbon emission, a major reason is the sources for generating electricity. Fossil fuels are still heavily used, since renewable sources are not always available due to weather condition. Recently, cloud-service leading providers are improving upon this matter. By having multiple large-scale data centers in different locations, companies can distribute duty to data centers that are in advantageous geographical location, where renewable energy is available. Renewable-powered data center remains a challenge; nevertheless, it is a bright signal that more effort is being directed towards sustainability initiatives.

Furthermore, upgrading IT equipment contributes considerably to reducing electricity losses. Modern facilities with optimal component provide effective performance as well as minimize electricity usage. Greenberg et al. [9] suggested that by running data centers in high temperature can reduce the amount of cooling, therefore save costs and energy. Barroso also stated that keeping the data centers temperature in range of 25^{o}C and 30^{o}C can benefit the chilling process [3]. Lee et al. [16] suggested a proactive cooling systems management method for data centers. By enabling proactive thermal management based on workload prediction, this system empowers managers to prevent heat imbalances before they impact temperature.

According to a study by Jalali et al. [13], deploying applications in nano data centers can significantly improve their energy efficiency. The researchers used a "flow-based" model to demonstrate that applications that produce and distribute large amounts of data to users who access it infrequently achieve the greatest energy savings. In addition, GreenCloud architecture is evaluated and recommended by Liu et al. [17]. The authors claim that this novel architecture has successfully achieved the goal of saving energy, and continue to provide low-latency performance for applications with strict time constraints. A critical examination of the methodology and principles behind Google's Carbon-Intelligent Compute management system has also shown a promising vision in achieving green

data centers and reducing carbon footprint [18].

5 Conclusion

This paper, as a literature review, has provided general background of three popular computing paradigms: cloud computing, fog computing, and edge computing. With focus on the energy consumption and the sustainability of large-scale computers (large-scale data centers), the paper has presented the growing challenge of large-scale data centers' energy usage and their impact on sustainability. The carbon emission of data centers significantly affects the environment, which calls for immediate actions towards protecting the ecosystem. Renewable energy must be maximized, the management boards need to acknowledge the importance of efficiently manage the operation of IT facilities, therefore can reduce over usage of energy and carbon footprint. Beyond the methodologies explored in this review, further methodologies have been documented in other relevant literature. For instance, by using key words such as "Green data centers", "carbon-aware data centers" can result in a diverse range of research inquiries and surveys regarding the discussed subject matter.

References

[1] Openfog consortium, openfog reference architecture for fog computing. Online, 2017. Available: `https://www.openfogconsortium.org/ra/`, February 2017.

[2] Victor Avelar, Dan Azevedo, Alan French, and Emerson Network Power. Pue: a comprehensive examination of the metric. *White paper*, 49, 2012.

[3] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *The datacenter as a computer: Designing warehouse-scale machines*. Springer Nature, 2019. doi: 10.1007/978-3-031-01761-2.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012. https://doi.org/10.1145/2342509.2342513.

[5] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE access*, 8:85714–85728, 2020. https://doi.org/10.1109/ACCESS.2020.2991734.

[6] Yanpei Chen, Vern Paxson, and Randy H Katz. What's new about cloud computing security. 2010. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.pdf.

[7] Kaouther Gasmi, Selma Dilek, Suleyman Tosun, and Suat Ozdemir. A survey on computation offloading and service placement in fog computing-based iot. *The Journal of Supercomputing*, 78(2):1983–2014, 2022. https://doi.org/10.1007/s11227-021-03941-y.

[8] Make IT Green. Cloud computing and its contribution to climate change. *Greenpeace international*, 83, 2010.

[9] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks, 2008. https://doi.org/10.1145/1496091.1496103.

[10] Jordi Guitart. Toward sustainable data centers: a comprehensive energy management strategy. *Computing*, 99(6):597–615, 2017.

[11] Keijo Heljanko. Presentation slides: Hadoop and big data, 03 2014.

[12] Yashpalsinh Jadeja and Kirit Modi. Cloud computing-concepts, architecture and challenges. In *2012 international conference on computing, electronics and electrical technologies (ICCEET)*, pages 877–880. IEEE, 2012. https://doi.org/10.1109/ICCEET.2012.6203873.

[13] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016. https://doi.org/10.1109/JSAC.2016.2545559.

[14] Jonathan G. Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(034008):8pp, 2008.

[15] Jing Lan, Yuge Ma, Dajian Zhu, Diana Mangalagiu, and Thomas F Thornton. Enabling value co-creation in the sharing economy: The case of mobike. *Sustainability*, 9(9):1504, 2017. https://doi.org/10.3390/su9091504.

[16] Eun Kyung Lee, Indraneel Kulkarni, Dario Pompili, and Manish Parashar. Proactive thermal management in green datacenters. *The Journal of Supercomputing*, 60:165–195, 2012. https://doi.org/10.1007/s11227-010-0453-8.

[17] Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pages 29–38, 2009. https://doi.org/10.1145/1555312.1555319.

[18] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2022. https://doi.org/10.1109/TPWRS.2022.3173250.

[19] Sherif Sakr, Anna Liu, Daniel M Batista, and Mohammad Alomari. A survey of large scale data management approaches in cloud environments. *IEEE communications surveys & tutorials*, 13(3):311–336, 2011. https://doi.org/10.1109/SURV.2011.032211.00087.

[20] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)*, 53(3):1–35, 2020. https://doi.org/10.1145/3391196.

[21] Subhadeep Sarkar and Sudip Misra. Theoretical modelling of fog computing: a green computing paradigm to support iot applications. *Iet Networks*, 5(2):23–29, 2016.

[22] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016. https://doi.org/10.1109/JIOT.2016.2579198.

[23] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition, 2008. https://doi.org/10.1145/1496091.1496100.

[24] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In Kuai Xu and Haojin Zhu, editors, *Wireless Algorithms, Systems, and Applications*, pages 685–695, Cham, 2015. Springer International Publishing.

[25] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019. https://doi.org/10.1016/j.sysarc.2019.02.009.

Assessing the Efficacy of Slow HTTP Attacks Against CDN Providers: Mechanisms and Strategies

Nicholas Jovianto
nicholas.jovianto@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

Content Delivery Networks (CDNs) are crucial in the internet because they optimize content delivery by improving speed and reliability for users globally. Despite their benefits, CDNs can be vulnerable to Denial of Service (DoS) attacks. This paper describes CDN architecture and the security challenges it might face with DoS attacks, specifically Slow HTTP attacks. Slow HTTP attacks can be differentiated as Slow Headers Attack, Slow Body Attack, and Slow Read Attack. This paper discusses the mechanism of the Slow HTTP attacks, the characteristic of each attack, and their effect on the availability of the service. This paper will also conduct some experiments to assess the efficacy of Slow HTTP attacks in several CDN providers, such as Cloudfront, Fastly and GCore CDN. Moreover, this paper discusses the experiment's results in depth and some defence strategies that can be used to improve CDN against these attacks.

KEYWORDS: Content Delivery Networks, Slow HTTP Attacks, Denial of Service Attacks, Defense Strategies, Cybersecurity

1 Introduction

Content Delivery Networks (CDN) have become a significant solution for improving web content delivery performance in this digital era [16]. CDN works by efficiently distributing content, optimizing load times, ensuring high availability, and providing security measures to improve user experience on the internet [14]. CDN has a robust geographically distributed infrastructure network worldwide with a multi-billion dollar market value [15]. This value makes CDN often a target of cybersecurity attacks such as Denial of Service (DoS) attacks.

One form of DoS attack in the CDN environment is the Slow HTTP attack [19]. Three commonly known types of Slow HTTP attacks are Slow Headers Attack, Slow Body Attack, and Slow Read Attack [20]. In 2020, Guo et al. [15] also mentioned a new type of attack like the Pre-POST Slow HTTP Attack. It shows that Slow HTTP attacks keep evolving. These attacks reduce server capabilities by exploiting limitations in HTTP connection handling. Slow HTTP attacks have become a particular issue in CDNs because some attack types can still be performed in a CDN environment.

Through a literature review, this paper aims to understand in-depth mechanisms behind Slow HTTP attacks, their characteristics, and their effect on the availability of services. Through the experiments, this paper aims to observe the efficacy of each type of slow HTTP attack across multiple CDN providers. Based on the literature review, this paper will discuss some defence strategies to improve security that can protect web services using CDNs to guarantee service integrity and availability.

This paper is organized as follows. Section 2 discusses the CDN architecture and its vulnerabilities towards the DoS attack. Section 3 discusses the slow HTTP attack and its type. Section 4 presents the analysis of each type of slow HTTP attack in the experiments and defence strategies against these attacks. Section 5 ends the paper with a conclusion.

2 Content Delivery Network (CDN)

This section provides basic information about CDN, such as its architecture, usage, and security challenges, such as the Denial of Service (DoS) it might face.

2.1 CDN Architecture

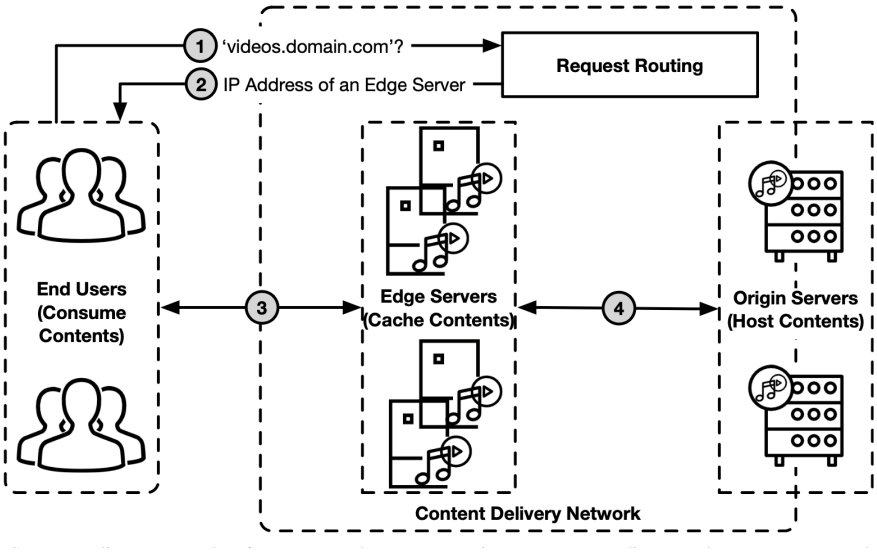


Figure 1. Abstract architecture of CDN [14]

CDN is a robust geographically distributed infrastructure network designed to deliver internet content faster by caching its content at multiple locations closer to end-users [14]. Figure 1 illustrates how CDN architecture was designed to enhance the accessibility and effectiveness of the content delivery process. The origin server is a critical component of this architecture because it is a repository for the original content [18]. This origin server is interconnected with many replicated edge server clusters geographically dispersed across many regions.

These edge servers cache content initially stored on the origin server [10]. Content request, such as images, web pages, or videos, is routed to the edge server in the user's nearest edge server rather than the origin server [17]. Consequently, latency is significantly reduced due to the limited distance the data must travel.

The origin server will be contacted to retrieve the requested content if the peripheral server cannot provide it [17]. After the content reaches the edge, it is cached to be prepared for subsequent requests and delivered to the user. This process ensures that the data that is most frequently accessed is situated in closer proximity to the user [14]. The implementation of data replication across a large number of peripheral servers guarantees both redundancy and expedited content delivery.

The service ensures continuous availability. In an outage or when one of the peripheral servers reaches its maximum capacity to process traffic, the request will be routed to the nearest edge servers [14]. Thus, beyond

merely facilitating content delivery, the primary aim of this CDN structure is to ensure its uninterrupted accessibility, notwithstanding periods of high traffic volume or server malfunctions.

2.2 Security Challenge: Denial of Service (DoS)

Although CDN has a robust architecture against cybersecurity attacks, Denial of Service (DoS) attacks still can provide CDNs with a security challenge [9]. The objective of DoS attacks is to prevent their intended users from accessing a machine or network resource by indefinitely interrupting the operations of an internet-connected host. In the CDN system, DoS attacks can be carried out by overwhelming the targeted system, such as the edge server, with many requests or by exploiting vulnerabilities of the CDN to crash the system.

DoS attacks come in various forms, targeting different parts of network infrastructure or server resources. One form of the attacks in DoS is slow HTTP attacks [11]. While many brute-force attacks flood systems with requests, slow HTTP attacks are more insidious, relying on sluggishly sending incomplete or tiny data within requests. This attack relies on the server's effort to keep connections open while waiting for the complete request. This approach can eventually exhaust server resources and prevent legitimate users from accessing the service.

In a CDN system, a slow HTTP attacker can initiate and maintain numerous slow connections to the server, severely slowing down or halting legitimate traffic by exploiting the web server's connection handling vulnerabilities. Suppose the attack happens in one of the peripheral servers in CDN. In that case, it can create a domino effect and cause broader network disruptions beyond the immediate target because they share the same infrastructure. Although the attack on an edge server of a CDN can have widespread effects, it is crucial to recognize that CDNs are inherently equipped to handle a higher volume of requests, making the threshold for causing disruption significantly more challenging to reach.

This attack may cause websites to experience extended loading periods. In more severe instances, the complete absence of websites or services. Companies that depend on CDNs for e-commerce and content delivery may also incur financial losses as a consequence [13]. This attack can affect the end-user experience and their trust in the service.

3 Slow HTTP Attack

A slow HTTP attack is one type of DoS attack [11]. It is a cyber tactic that disrupts service by exploiting how websites handle data transfers. This attack stretches out the time a website connection stays open to the max while using as little bandwidth as possible. This sneaky move strains the website's resources, making it difficult for the server to process legitimate user requests. The fallout from such an attack can range from noticeable slowdowns in website performance to complete denial of access for users who should rightfully be able to use the service.

There are three common categories of slow HTTP attacks: Slow Headers Attack (Slowloris), Slow Body Attack (R-U-Dead-Yet), and Slow Read Attack [20]. Each attack has unique characteristics or methods for attacking the network. These attacks reduce server capabilities by exploiting limitations in HTTP connection handling. The purpose of this attack is to overwhelm the server with a long connection queue, which can slow down or make the server unable to process any incoming requests. Although Guo et al. [15] also mentioned a new type of attack like the Pre-POST Slow HTTP attack, this paper will focus only on three categories mentioned earlier because the slowhttptest tools used for the experiment only support those types of attacks.

3.1 Slow Headers Attack (Slowloris)

Slow Headers Attack, also known as Slowloris attack, is a slow HTTP attack that sends an HTTP GET request with a partially incomplete header and very slowly to the server [2, 6]. The attackers exploit this behaviour to execute the DoS attack. This action leaves incomplete HTTP requests in the server queue, making the server wait for incomplete header parts. Moreover, the attacker can generate many requests to the server, making it unable to handle any further requests. In the context of CDN, Guo et al. [15] show that this attack is considerably old, and most of the CDN providers have been able to detect and protect against this attack.

3.2 Slow Body Attack (R-U-Dead-Yet)

Slow Body Attack, also known as R-U-Dead-Yet (RUDY) attacks, is a type of slow HTTP attack that focuses on the contents of the HTTP request or body [4]. RUDY starts a POST request with a valid header, but it sends

the body of the request one byte at prolonged intervals. This action leaves the server tied to one connection for long periods, and it finally consumes some of the server's resources, making it inaccessible to legitimate users.

In the context of CDN, Guo et al. [15] show that CDNs use two main methods for forwarding POST requests to the origin server. This behaviour can differentiate Slow Body Attacks into POST and pre-POST attacks based on CDN behaviour to handle it. One approach is to wait until the entire POST message is received before forwarding, which ensures complete data transfer but can introduce delays if the message body is significant. The other approach, pre-POST forwarding, sends the request to the origin server when the POST header is received and sends the message body as it comes in. Pre-POST forwarding is considered faster, but it allows an attacker to exploit this behaviour by keeping connections open for extended periods, potentially leading to resource exhaustion on the origin server. This attack is still an issue for some CDN providers because it can still be performed in the system.

3.3 Slow Read Attack

Slow Read attacks are a DoS attack that exploits the TCP connection flow mechanism [7]. The attackers execute the attack by sending valid headers and HTTP GET requests to the server but intentionally slow down the server's read response time. The attacker intentionally lowers their TCP window rate. Thus, the server has to maintain an active connection for a long time to wait for the read process to be done. In the CDN, Guo et al. [15] showed that most of the CDN providers have been able to protect against this attack.

4 Discussion

This section shows the experiment of each slow attack in several CDN providers, the analysis of the experiment results, and the defence mechanism strategies to improve the security of CDN from these attacks.

4.1 Experiment Setup

CDN providers have implemented several layers of protection to defend their systems against DoS attacks, including slow HTTP attacks. This paper explores whether slow HTTP attacks can still be executed on several

CDN providers by conducting real-case attack experiments. By evaluating the effectiveness of these attacks against modern CDN systems, this paper aims to provide insights into the robustness of current protection mechanisms regarding these attacks.

In 2020, Guo et al. [15] mentioned that the Slow Body HTTP attack can occur on Cloudfront, Fastly, and MaxCDN. This paper uses this as a reference when conducting experiments and tries to verify whether this attack can still be performed in these CDN providers. Because MaxCDN has transformed into Stackpath and its free plan is limited, this paper proposed another CDN provider, Gcore CDN, to replace MaxCDN because it also claims to protect against DoS attack [1, 3]. Finally, this paper will use Cloudfront, Fastly, and Gcore CDN.

The experiments were conducted on a private Nginx web server created in the DigitalOcean Virtual Machine (VM) using the Ubuntu 23.10 operating system. This paper utilizes the free plans as the main CDN service plan for these experiments. Some configurations needed to be done on the CDN provider side to support the experiments, such as adding the web server domain to the management console and updating the name servers on the provider domain's DNS settings to reroute the traffic through their services, thereby web server can utilize the CDN and its security features.

```
1 $ slowhttptest -H -u https://nichojovi.com/ -c 5000 -r 100 -l 600 -g -o
  ↪  slowloris-test
2 $ slowhttptest -B -u https://nichojovi.com/ -c 5000 -r 100 -l 600 -s 16384 -g
  ↪  -o slowpost-test
3 $ slowhttptest -X -u https://nichojovi.com/ -c 5000 -r 100 -l 600 -n 5 -z 32
  ↪  -g -o slowread-test
```

Listing 1. Commands used for the experiment

The experiments were running using the slowhttptest tools [5]. This tool carries out standard low bandwidth application layer DoS for Slow HTTP attacks. This tool will produce CSV and HTML files with test statistics for further analysis. The commands used in the experiments can be seen in Listing 1, and the settings of each experiment can be seen in Table 1. The experiment will run with 5000 connections, 100 connections per second, and in a timeframe of 600 seconds. This setup is used to observe the pattern of each attack for a longer time and with quite a large number of requests. There are three commands for each type of Slow HTTP Attack: -H for the Slow Header attack (Slowris), -B for the Slow Post (RUDY) attack, and -X for the Slow Read attack.

Parameter	Slow Header	Slow Body	Slow Read
Number of Connection	5000	5000	5000
HTTP Method	GET	POST	GET
Content-Length Value	4096	16384	-
Extra data max length	68	66	-
Interval between follow up data	10 seconds	10 seconds	-
Receive Window Range	-	-	1 - 512
Read rate from receive buffer	-	-	32 bytes / 5 sec
Connections per seconds	100	100	100
Timeout for probe connection	5	5	5
Target test duration	600 seconds	600 seconds	600 seconds

Table 1. Test settings of each type of Slow HTTP attack

4.2 Experiment Results: CloudFront

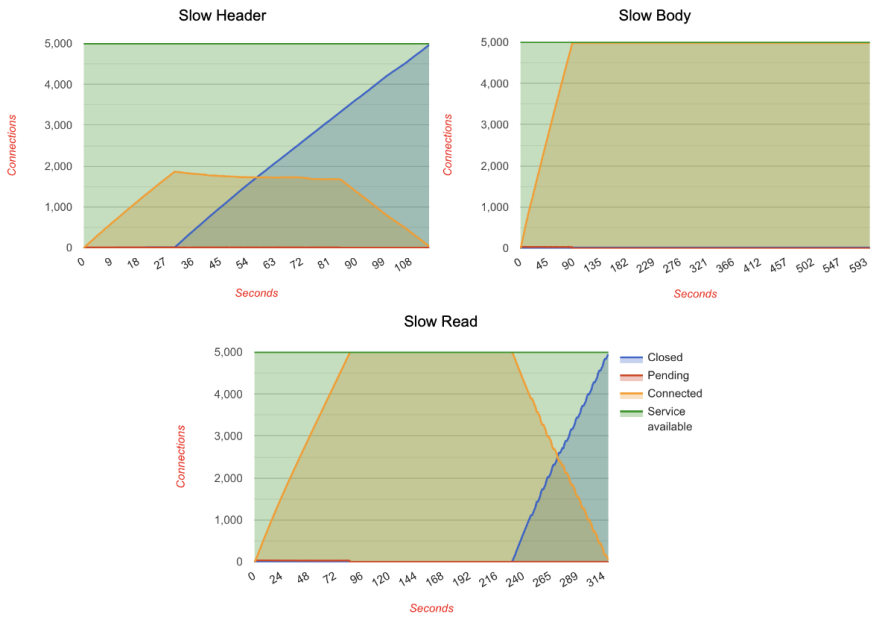


Figure 2. Test result of CloudFront

Figure 2 illustrates the effectiveness of a CDN provider, CloudFront, in mitigating different types of Slow HTTP attacks. During the Slow Header and Slow Read attacks, CloudFront's defence mechanisms allowed the server to manage 5000 GET connections by throttling some connections, indicating a resilient system designed to handle high loads and maintain the server's availability. During the Slow Body attacks, CloudFront defence mechanisms enabled the server to manage 5000 POST connections, demonstrating an improvement in the system to handle great demands while maintaining server availability.

```
162.158.134.135 - - [12/Mar/2024:14:28:03 +0000] "GET / HTTP/1.1" 200 409 "TESTING_PURPOSES_ONLY" "Mozilla/5.0 (W
indows NT 6.1; WOW64; Trident/7.0; rv:11.0) like GeckoAppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122
Safari/534.30"
162.158.134.10 - - [12/Mar/2024:14:28:06 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"
162.158.134.44 - - [12/Mar/2024:14:28:08 +0000] "GET / HTTP/1.1" 200 409 "TESTING_PURPOSES_ONLY" "Mozilla/5.0 (Wi
ndows NT 6.1; WOW64; Trident/7.0; rv:11.0) like GeckoAppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122 S
afari/534.30"
```

Figure 3. Nginx server access log

Requests from the web server are also monitored through the Nginx access log in the DigitalOcean cloud server. Figure 3 shows the sample log from the Cloudfront in the Nginx server. As we can see, the 200 HTTP status code responses indicate that the request is successful, and the 304 HTTP status code indicates that the response has been cached. This method also monitors server availability and checks the server status when the test occurs. During the test with Cloudfront, the Nginx server ran smoothly without any disturbance.

4.3 Experiment Results: Fastly

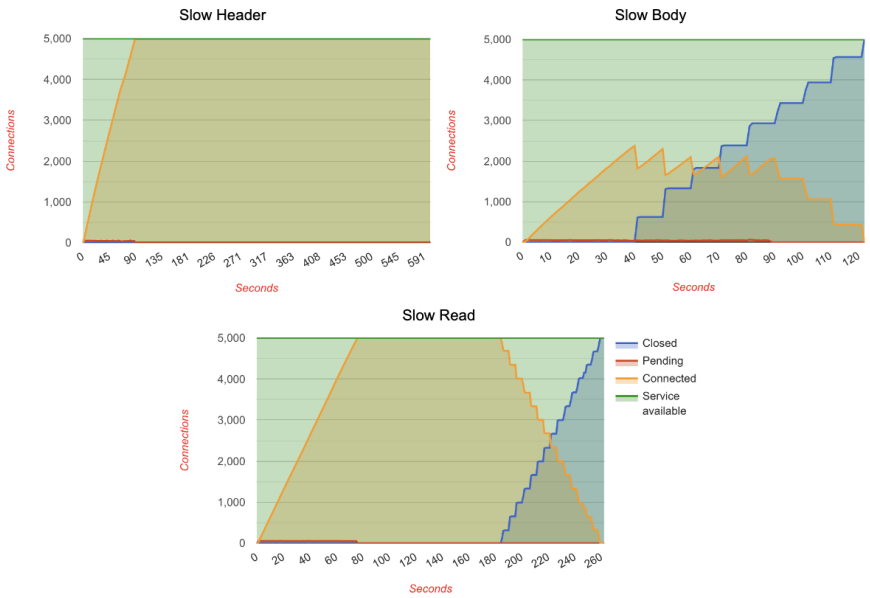


Figure 4. Test result of Fastly

Figure 4 shows the effectiveness of Fastly in mitigating various types of Slow HTTP attacks. During the Slow Header and Slow Read attacks, CloudFront's defence measures enabled the server to manage 5000 GET connections by limiting some of them, demonstrating a resilient system built to handle heavy loads while maintaining server availability. During the Slow Body attacks, Fastly's defence mechanisms enabled the server to manage 5000 POST connections, demonstrating a resilient system designed to handle great demands while maintaining server availability.

Fastly's cache mechanism works in the Nginx access log, so only one initial request was sent to the Nginx server regarding each experiment, which indicates the rest of the request was cached on the CDN side.

4.4 Experiment Results: Gcore CDN

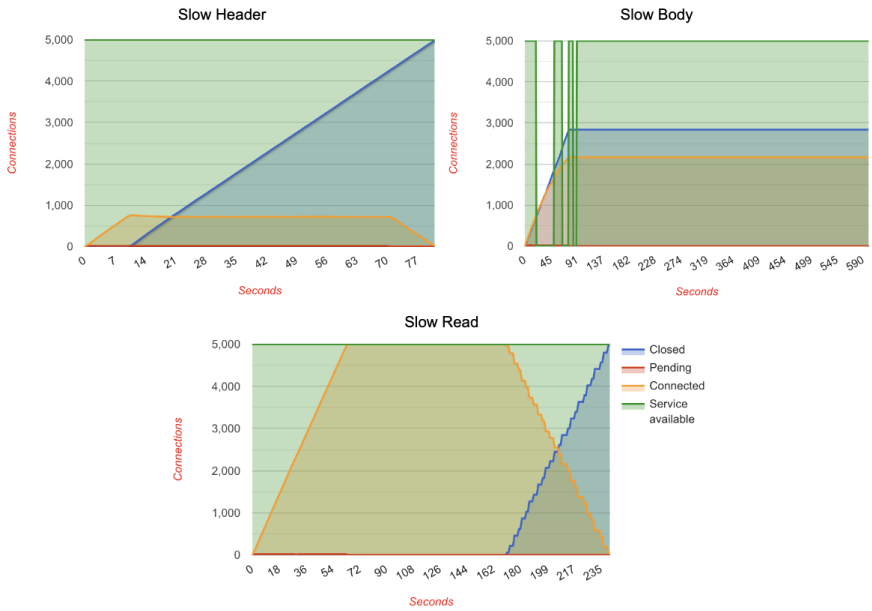


Figure 5. Test result of Gcore CDN

Figure 5 shows the efficiency of Gcore CDN in mitigating various types of Slow HTTP attacks. During the Slow Header and Slow Read attacks, Gcore CDN's defence mechanism demonstrates a resilient system designed to handle heavy loads while maintaining server availability. However, this approach could have been more effective during the Slow Body attack, when the server had significant issues with 5000 POST connections, resulting in multiple downtimes. The web availability was assessed in the browser, the Nginx access log, and the server status in the DigitalOcean VM to verify the downtime. The URL could not be accessed in the browser, but no anomaly happened in the Nginx access log or server status. These indicate that the downtime only happened in Gcore CDN and did not reach the central Nginx server.

4.5 Experiment Analysis

Security against certain types of attacks is one of the critical factors in CDN. CDN has implemented several protection systems to protect its system from DoS attacks, specifically Slow HTTP attacks. Research in this area has been conducted to observe CDN's effectiveness against this at-

tack. Guo et al. [15] mentioned that Slow Body attacks, such as Pre-POST Slow HTTP attacks, are still possible in CDN. In contrast, Slow Header and Slow Read attacks should not be able to pass through the CDN.

Guo et al. [15] highlighted that while Slow Headers and Slow Read attacks are generally mitigated, Slow Body attacks pose a significant challenge. Their research shows that CDNs can defend against slow header attacks by forwarding requests after receiving the complete HTTP header and halting slow read attacks due to the independence of CDN–originating transmission from the attacker's transmission. However, Slow Body attacks present a significant challenge due to complexities in POST forwarding decisions, which either delay forwarding until the entire message is received or allow attackers to extend CDN–origin connections through sequential POST message forwarding.

The current experiment reveals that Cloudfront and Fastly improve the ability to handle all types of Slow HTTP attacks compared to the prior studies. On the other hand, the new CDN provider used in this experiment, Gcore CDN, only shows resilience against Slow Headers and Slow Read attacks. However, it struggles significantly with Slow Body attacks. This outcome aligns with prior findings and shows that their defences could have been more effective against this attack. It highlights the critical need for ongoing improvements in Gcore CDN defence mechanisms to handle these attack vectors effectively.

4.6 Defence Strategies

CDN is a highly adaptable framework that caters to the need for increased internet usage, speed, reliability, and security in today's interconnected world. Because of this role, CDN was designed with a robust and safe system architecture. CDNs incorporate several supplementary functionalities into their infrastructures, one of which is load balancing [8]. This load balancing mechanism spreads out the requests to CDN, among many servers, to avoid overwhelming a particular server. The design of a CDN also includes security features, e.g., caching, rate limiter, and monitoring system, to safeguard its system against data breaches, DoS attacks, and other security threats [18].

CDNs utilize several mitigation techniques to prevent DoS attacks, including defences against Slow HTTP attacks [12]. Based on the experiment result, most of the CDN providers involved in this experiment handle Slow Header and Slow Read attacks well. However, slow body

attacks seem challenging for Gcore CDN because they still cause service availability issues.

Guo et al. [15] suggested that CDN providers adopt a more secure method of handling POST requests to handle Slow Body attacks, like the store-then-forward approach used by Cloudflare. The store-then-forward approach mitigates the Slow Body attacks by requiring a CDN to wait until the entire POST request body is received before forwarding it to the origin server. This strategy prevents attackers from exploiting CDN behaviour to exhaust connection limits at the origin, protecting against slow HTTP DoS attacks that aim to deplete server resources.

Despite these defensive mechanism strategies, DoS attacks persistently progress as malicious actors uncover methods for bypassing security protocols. Consequently, CDNs have a responsibility to modify their strategies consistently. CDN must construct defensive systems to protect against service interruptions such as DoS attacks and guarantee performance and availability despite persistent attacks.

5 Conclusion

CDNs are frequently utilized to enhance the efficiency and security of web applications. Despite their comprehensive features, CDNs still have inherent weaknesses, such as the flaw in handling HTTP connections that slow HTTP attackers usually use to perform the attack. The paper highlights that CDNs, such as Cloudfront, Fastly and Gcore CDN, need to continually evolve their defence mechanisms against Slow HTTP attacks, especially Slow Body attacks. Despite existing protections, Slow Body attacks remain potent, exploiting specific vulnerabilities within CDN architectures. The paper shows the importance of adopting advanced strategies to enhance CDN resilience, such as the store-then-forward approach. The ongoing challenge posed by DoS attacks necessitates constant improvement in security measures to ensure CDN service's uninterrupted performance and availability amidst evolving digital threats.

References

[1] Global ddos protection service. https://gcore.com/ddos-protection. Accessed: 2024-03-18.

[2] Hijacking web 2.0 sites with sslstrip and slowloris. https://www.security-

portal.cz/blog/hijacking-web-20-sites-sslstrip-and-slowloris. Accessed: 2024-03-24.

[3] Maxcdn is now stackpath. https://www.stackpath.com/maxcdn/. Accessed: 2024-03-18.

[4] R u dead yet? (r.u.d.y.) attack. https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/. Accessed: 2024-02-24.

[5] slowhttptest(1) - linux man page. https://linux.die.net/man/1/slowhttptest. Accessed: 2024-02-29.

[6] Slowloris ddos attack. https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/. Accessed: 2024-02-24.

[7] What is a low and slow attack. https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/. Accessed: 2024-02-24.

[8] Yun Bai, Bo Jia, Jixiang Zhang, and Qiangguo Pu. An efficient load balancing technology in cdn. In *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 7, pages 510–514, 2009. https://doi.org/10.1109/FSKD.2009.130.

[9] Jianjun Chen, Jian Jiang, Xiaofeng Zheng, Haixin Duan, Jinjin Liang, Kang Li, Tao Wan, and Vern Paxson. Forwarding-loop attacks in content delivery networks. 2016. https://doi.org/10.14722/ndss.2016.23442.

[10] Michael D. Dahlin, Randolph Y. Wang, Thomas E. Anderson, and David A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *First Symposium on Operating Systems Design and Implementation (OSDI 94)*, Monterey, CA, Nov 1994. USENIX Association.

[11] A. Dhanapal and P. Nithyanandam. The slow http distributed denial of service attack detection in cloud. *Scalable Computing*, 20(2):285 – 298, 2019. https://doi.org/10.12694/scpe.v20i2.1501.

[12] Maurizio D'Arienzo and Serena Gracco. A survey on cdn vulnerability to dos attacks. *International Journal of Computer Networks and Communications*, 15(5):127 – 145, 2023. https://doi.org/10.5121/ijcnc.2023.15508.

[13] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: Flexible and elastic ddos defense. pages 817 – 832, 2015. https://api.semanticscholar.org/CorpusID:1681696.

[14] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A. Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. Content delivery network security: A survey. *IEEE Communications Surveys Tutorials*, 23(4):2166–2190, 2021. https://doi.org/10.1109/COMST.2021.3093492.

[15] Run Guo, Weizhong Li, Baojun Liu, Shuang Hao, Jia Zhang, Haixin Duan, Kaiwen Shen, Jianjun Chen, and Ying Liu. Cdn judo: Breaking the cdn dos protection with itself. 2020. https://doi.org/10.14722/ndss.2020.24411.

[16] Maryan Kyryk, Maryana Pleskanka, and Nazar Pleskanka. The efficiency and productivity of the cdns. pages 270 – 273, 2017. https://doi.org/10.1109/AIACT.2017.8020117.

[17] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, Aug 2010. https://doi-org.libproxy.aalto.fi/10.1145/1842733.1842736.

[18] Behnam Shobiri, Mohammad Mannan, and Amr Youssef. Cdns' dark side: Security problems in cdn-to-origin connections. *ACDM Journals - Digital Threats: Research and Practice*, 4(1), Mar 2023. https://doi.org/10.1145/3499428.

[19] Hengxian Song, Jing Liu, Jianing Yang, Xinyu Lei, and Gang Xue. Two types of novel dos attacks against cdns based on http/2 flow control mechanism. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13554 LNCS:467 – 487, 2022. https://doi.org/10.1007/978-3-031-17140-6_23.

[20] Suroto Suroto. A review of defense against slow http attack. *JOIV : International Journal on Informatics Visualization*, 1:127, 11 2017. https://doi.org/10.30630/joiv.1.4.51.

Review of current k-means and k-median clustering research

Niilo Heinonen
niilo.i.heinonen@aalto.fi

Tutor: Parinya Chalermsook

Abstract

This paper provides an analysis of k-means and k-median clustering, two central algorithms in the study of centroid-based clustering. Despite their intuitive appeal and widespread application across various domains, both problems present significant computational challenges, mostly due to their NP-hard nature. We review the current landscape of approximation algorithms, focusing on recent advancements that have improved upper bound guarantees for these clustering problems. Our exploration covers Fixed Parameter Tractable (FPT) algorithms and the limitations of polynomial-time algorithms. Through an examination of the latest breakthroughs, particularly the work by Cohen-addad et al., we highlight the evolving efficacy of algorithmic solutions in overcoming the complexities inherent in k-means and k-median clustering.

KEYWORDS: clustering, k-median, k-means

1 Introduction

In the context of computer science, clustering most often refers to the task of grouping objects (often data points) into clusters where members of one cluster share more similarities with each other than with members of an-

other cluster. There are multiple optimization goals within the clustering domain, and a good solution for one problem might be sub-optimal for another. Therefore, users must choose a goal that best fits their specific needs.

While the tasks are often quite intuitive and easy to understand, solving them tend to be demanding. Most clustering tasks are NP-hard [11], meaning that there is no general, computationally efficient algorithm to find an optimal solution. This in turn has lead to the development of different algorithms that try to approximate solutions to different clustering problems. As a field of study, clustering is primarily concerned with improving these algorithms to both efficiently and accurately accomplish these different clustering tasks.

These approximation algorithms are used in various applications, ranging from image segmentation and compression [9] to market segmentation [4] and data mining [12]. Improving algorithms coupled with the continuous growth of computing power has rendered many previously demanding clustering problems feasible, thereby expanding the range of applications that utilize this field. Notably, advances in machine learning [6] have spurred great interest in improving algorithms for k-means and k-median problems, as they serve as a fundamental building block for many types of AI technologies, and in this paper we take a closer look to these two problems.

2 Background

2.1 Introduction to Clustering

Consider the well-known k-*center* problem, where a town with n houses seeks to optimally allocate resources for constructing k fire stations. The objective is to select cluster centers in a manner that minimizes the maximum distance from any house to a fire station, formally defined as:

$$\Phi_{k\text{center}}(C) = \max_{j=1}^{k} \max_{a_i \in C_j} d(a_i, c_j)$$

2.2 Centroid-based Clustering Problems

This paper focuses on two problems within the centroid-based clustering family, characterized by:

Given a dataset of n points in a metric space and a predefined constant k, centroid-based clustering seeks to identify a set of k points as cluster centers. In discrete problems, centers must be selected from the input dataset, whereas in continuous problems, centers can occupy arbitrary positions within the space. Following center selection, data points are assigned to the nearest center, and the aim is to minimize the cost of the chosen k-centers configuration.

Drawing an analogy to the k-center problem, where centers are likened to fire stations, the centers in k-means and k-median problems can be compared to grocery stores. The influence of outliers is mitigated unless their prevalence justifies establishing a new center, thereby optimizing the solution.

k-means

The k-means objective is to minimize the sum of squared distances from data points to their nearest cluster center, mathematically represented as:

$$\Phi_{k\text{-means}}(C) = \sum_{j=1}^{k} \sum_{x \in C_j} \|x - c\|^2$$

where X denotes the set of all points, and C represents the chosen centers.

k-median

Contrarily, the k-median problem aims to minimize the sum of distances from each dataset point to its nearest center, defined as:

$$\Phi_{k\text{-median}}(C) = \sum_{j=1}^{k} \sum_{a_i \in C_j} \|x - c\|$$

Differing from k-means, the k-median problem's focus on median distances enhances its robustness to outliers and variable cluster sizes, rendering it more suitable for certain applications.

Lloyd's Algorithm

The Lloyd's algorithm, often noted as naive "k-means algorithm", is a greedy approximation algorithm that finds a local optimum for the k-means problem.

The algorithm is built as follows:

1. Start with k centers.

2. Cluster each point with the center nearest to it.

3. Find the centroid of each cluster and replace the set of old centers with these centroids.

4. Repeat the above two steps until the centers converge according to some criterion, such as when the k-means score (the sum of squared distances from each point to its nearest center) is no longer improving.

The Lloyd's algorithm is guaranteed to find a local optimum, but not necessarily a global one. While it has served as an important foundation for the whole field, its performance can be sensitive to the initial placement of the centroids and it often struggles with clusters of varying sizes and densities, or non-spherical shapes.

2.3 Applicability of k-means and k-median

In their book [3], the authors denote that the k-means is more often used on data spanning the d-dimensional Euclidean space $\mathbb{R}^d$, while the k-median is preferred when clustering is done on graphs where distances don't necessarily follow the Euclidean norm. In the k-means problem where distance squared is the optimization criterion, given a set of points that belong to a cluster the best center for that cluster is the centroid of the points, which makes it the natural choice for clustering in $\mathbb{R}^d$.

3 Recent Developments

Recent algorithmic advancements have demonstrated that techniques beneficial to k-means clustering can often be adapted for k-median clustering, and vice versa, motivating a unified study of these problems. The essence of approximation algorithms is to refine the upper bounds of solutions, aiming to minimize the worst-case scenario relative to the optimal solution. While achieving a near-perfect clustering solution might meet the approximation criteria, it would entail a computational complexity nearly identical to that required for an optimal solution, which is exponential in terms of the input size for k-median and k-means clustering. Consequently, research in approximation algorithms categorizes them based on their time complexities, focusing on varying constraints.

3.1 FPT Algorithms

Fixed Parameter Tractable (FPT) algorithms belong to a complexity class where the runtime may exhibit super-polynomial growth solely in relation to a specific parameter (k in this context), denoted as $f(k, \varepsilon)n^{O(1)}$. The polynomial growth of terms dependent on the input size ensures the tractability of FPT algorithms for instances where k remains relatively small.

A notable 2019 study [7] presented an algorithm that approximates k-median and k-means with factors of $(1+2/e+\varepsilon)$ ($\approx$ 1.736) and $(1+8/e+\varepsilon)$ ($\approx$ 3.943), respectively. This study also asserted that, under prevailing complexity-theoretic conjectures, no FPT-time algorithm could surpass these approximation factors. However, this conclusion pertains specifically to general metric spaces, and the study acknowledges that tighter approximations are achievable within Euclidean spaces, even under more stringent time complexities.

3.2 Polynomial Time Algorithms

Polynomial time complexity represents a stricter criterion compared to FPT, as it precludes exponential growth in terms of any input variable (e.g., n^3 is permissible, but 3^n is not). Achieving effective approximations within polynomial time is feasible in more constrained metric spaces, particularly in Euclidean spaces. As elaborated upon in the 2019 study [6], while efficient approximation algorithms have been developed for certain scenarios (such as when k or dimension d is fixed), the most challenging and pertinent cases involve k and d as variable inputs. Consequently, potential algorithms cannot rely on run-times that exponentially depend on these variables.

Recent studies have delineated the inapproximability limits for polynomial-time algorithms, establishing that it is not feasible to achieve arbitrarily precise approximation guarantees. Specifically, in the discrete scenario, k-means and k-median clustering face approximation limits of 1.17 and 1.07 under the assumption that P $\neq$ NP, and 1.73 and 1.27, respectively, assuming the Johnson-Coverage hypothesis. For continuous scenarios, these limits adjust to 1.06 and 1.015 under P $\neq$ NP, and 1.36 and 1.08 with the Johnson-Coverage hypothesis [5] [8].

The well-known Lloyd's algorithm, introduced earlier 2.2, provides a simple heuristic for the k-means problem. However, its simplicity be-

lies potential inefficiencies and the risk of sub-optimal solutions, lacking a guaranteed approximation bound, the solutions might be arbitrarily more expensive than optimal, and potentially failing to converge in polynomial time [2]. This has spurred extensive research into more robust algorithms.

As of 2021, the most effective polynomial-time algorithms for clustering problems in Euclidean spaces, offering approximation guarantees of 5.912 for k-means and 2.406 for k-median, were introduced by Cohen-addad et al. [6]. This research improved upon the foundational work by Ahmadian et al. in 2017 [1], which itself marked a significant advancement from the best-known k-means approximation factor of $9 + \epsilon$, established in 2004 by Kanungo et al. [10].

4 Conclusion

This paper has explored the intricacies of k-means and k-median clustering, two pivotal problems within the domain of centroid-based clustering. We began by outlining the fundamental nature of clustering tasks in computer science and progressed through a detailed examination of the k-means and k-median problems, highlighting their significance, challenges, and applications.

Our discussion underscored the continuous advancement in algorithmic strategies aimed at improving the approximation guarantees for k-means and k-median clustering. We identified the role of FPT algorithms in addressing these challenges, offering scalable solutions under specific parametric constraints. Moreover, we illuminated the limitations inherent in polynomial-time algorithms, presenting the current boundaries of approximation as defined by recent inapproximability results.

Significantly, our review of recent developments revealed a promising trajectory of research, wherein innovations in algorithmic design have led to more efficient and effective solutions for these longstanding problems. The breakthroughs by Cohen-addad et al., building upon prior work by Ahmadian et al. and others, mark a pivotal advancement in the field, offering enhanced approximability guarantees and setting new benchmarks for future research.

In conclusion, k-means and k-median clustering continue to be at the forefront of research in data science and machine learning, serving as fundamental tools for data analysis and interpretation. The ongoing evo-

lution of approximation algorithms for these problems not only broadens their applicability but also deepens our understanding of the underlying mathematical and computational principles. As we move forward, it is imperative to continue this trajectory of innovation, further pushing the boundaries of what is computationally feasible while striving for the most effective and efficient clustering solutions.

References

[1] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms, 2017.

[2] D. Arthur and S. Vassilvitskii. How slow is the k-means method? *Proceedings of the Annual Symposium on Computational Geometry*, 2006:144–153, 01 2006.

[3] Avrim Blum, John Hopcroft, and Ravi Kannan. *Foundations of Data Science*. 01 2020.

[4] Chui-Yu Chiu, Yi-Feng Chen, I-Ting Kuo, and He Chun Ku. An intelligent market segmentation system using k-means and particle swarm optimization. *Expert systems with applications*, 36(3):4558–4565, 2009.

[5] Vincent Cohen-Addad and Karthik C.S. Inapproximability of clustering in lp metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539, 2019.

[6] Vincent Cohen-Addad, Hossein Esfandiari, Vahab Mirrokni, and Shyam Narayanan. Improved approximations for euclidean k-means and k-median, via nested quasi-independent sets. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1621–1628, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight fpt approximations for k-median and k-means, 2019.

[8] Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in l_p metrics, 2021.

[9] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015. Eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India.

[10] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry*, 28(2):89–112, 2004. Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002.

[11] Adam Letchford. Approximation algorithms: Vv vazirani, springer-verlag, 2001. xix + 378 pp. 34.95 isbn:3-540-65367-8. *Journal of the Operational Research Society*, 53:807–808, 07 2002.

[12] Junjie Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

Empowering the Edge: Innovations and Challenges in User-Provided Infrastructure

Olaus Lintinen
olaus.lintinen@aalto.fi
2.4.2024

Tutor: Sara Ranjbaran

Abstract

Fog and edge computing have received a great deal of attraction in recent years. These technologies promise to reduce latency for time critical applications and provide better quality of service for users. Several interesting research papers have been released during the past decade and various models for utilizing edge and fog networks have emerged. Resource sharing networks consisting of internet-of-things devices are seen promising. Although studies have found the technology and models to deliver promising results, there exists a need for proof-of-concepts and innovation. The proposed models displayed in this paper share insight into potential future implementations. It was found that these models complement each other in ways that when combined could enhance the quality of service significantly.

Moreover, this paper conducts a literature review on timely research of User-Provided Infrastructure by going through proposed models for utilizing edge and fog paradigms. Furthermore, technical, and incentive challenges and future study directions are explained and explored.

KEYWORDS: Internet of Things, Edge, Fog, User-Provided Infrastructure, Low Latency

1 Introduction

In recent years, edge computing has shown significant potential as an alternative to traditional cloud computing paradigm [1], [2], [3].

The hopes for enhanced quality of service seem to be filled as the study into edge computing and internet of things (IOT) advances. This research has sprouted an interest in studying whether mobile IOT devices were able to form federations for even greater benefit, like the change from cloud to edge computing. Although the topic is still relatively novel, studies have shown exciting potential in these types of paradigms [1].

Furthermore, as data processing and prediction integrates more to consumer products, there is a growing need for lower latency networks [3]. By moving things from cloud to the edge of the network, or even to the fog, we could harness the untapped computing power of heterogeneous device pool, ranging from smartphones to cars to embedded IOT devices.

Although these technologies promise remarkable benefits and enhancements to our current digital lives, they have yet to deliver their full potential. While the field of edge and fog has a robust background in science, it is still met with the challenges of implementing novel technologies in real world applications. To overcome these challenges, there persists a need for more research with a focus on proof-of-concepts (POC) and innovations.

This paper aims to explore and introduce recent studies, discoveries and inventions related to User-Provided-Infrastructure (UPI). In the following sections, the paper first covers some core concepts, after which it will introduce some proposed models for utilizing edge and fog networks for UPI, and lastly it will go over some of the challenges. In the end, there will also be a discussion based on the findings of this paper and a conclusion on what further studies should focus on.

2 Core Concepts

In this section, the paper briefly explains the core concepts and paradigms around edge and fog computing and how they differ from each other. First the concept of cloud is introduced, following description of IOT, after which, the basics of edge and fog computing are covered. Overall, this section aims to create a base line understanding of the building blocks of User-Provided infrastructure.

2.1 Cloud

As of now, cloud is a more popular choice for hosting various applications, storage, and computing power than ever [4]. The change from maintenance requiring, difficult to scale when the needs, on-premises data centers (DC) has been ongoing for the past decade or two. The clear advantages of cloud over traditional own DC have made it, with its vast selection of pre-built applications and infrastructure, mostly uncontested when it comes to computing. However, due to the usual, distant location of the DCs, the latency associated with cloud is usually significantly greater than closer deployed solutions.

Modern lives, be they at home or at the supermarket, are monitored on a vast scale. This monitoring is powered with an assortment of sensors working together. The data collected is further used for data analysis to yield predictions on human behavior, weather, and so on. For such systems to work in real time, the need for low latency is a key. The solution proposed for this challenge is to, in certain cases, move back from the cloud to the edge or even to the fog. These terms will be explained in the later sections.

2.2 Internet of Things

IOT refers to a network of interconnected devices with sensors, and computing power, that communicate between each other. The core idea around IOT is the introduction of smart systems, e.g., smart homes, where an assortment of connected sensor devices could enhance the experience, in this case living, by providing more optimization from the data collected. IOT as a term did not exist before Kevin Ashton proposed it in 1999. [5] Moreover, IOT devices are strongly linked edge and fog paradigms since both utilize a network of interconnected devices. This connection of different technologies will be touched upon in later sections.

2.3 Edge

The term edge computing refers to, as the name suggests, the area of the network which is at the "edge" of the cloud. Bringing computing and content closer to the end-user lowers the latency providing users better quality of service (QOS) in terms of faster internet connectivity and greater efficiency in data processing. In addition to that, edge computing also of-

fers location awareness, which for some delay-sensitive use cases, might enhance their efficiency. [6] Moreover, as the consumption of internet content is increasing annually, especially on Over-the-top (OTT) platforms [7], there exists a growing need for edge network applications, such as content delivery networks (CDN).

2.4 Fog

Fog computing, first introduced by Cisco [8], is a cloud computing paradigm which extends from the idea of edge computing. It offers even lower latency compared to edge computing with enhanced mobility and widespread geographic distribution. Use cases of fog range from connected vehicles to smart grids and various wireless sensor systems. During the past decade, fog computing has been a raising topic in scientific studies [1], [3], [8], [9].

Especially the interest for low latency applications has accelerated the research for creating models for utilizing fog computing. Although it is a decade old concept, there have not been significant successes so far. However, as technology offers, in certain situations, great advantages over the traditional cloud computing paradigm, it is an appealing topic for researchers and entrepreneurs.

3 Proposed Models

Although the idea of UPI has been around for some time, it has been lacking a working model to implement it for real world applications. While some companies have tried implementing this concept, a great breakthrough has yet to emerge. In this section, the paper aims to display some of the proposed models and frameworks for better utilization of edge and fog.

3.1 3C Recourse Sharing Framework

The 3C framework works by efficient utilization of device-to-device (D2D) connections among mobile devices. This model proposes that with the formation of co-operative groups, mobile devices can experience enhancement in efficiency in accomplishing tasks. Although resource sharing schemes have been proposed before, the key factor setting 3C framework apart from the previous ones is that it considers all three of the resources

mobile devices may share, communication, computation, and caching. The article proposing 3C framework, found that energy consumption of participating devices was significantly reduced. With many mobile devices having small battery lives, energy efficiency brings more flexibility to the fog network as it provides greater mobility opportunities. The 3C framework's advantage is that it can utilize a heterogeneous pool of devices which enables better resource allocation for different tasks. For example, some devices might possess more storage than others and some might have faster and more stable internet connection. Therefore, the framework allows these resource rich devices to utilize the overhead they have on those resources. [2]

In addition to providing novel framework for resource sharing, the researchers also outline potential issues and challenges. These include incentive schemes, security and privacy, and the need for carefully designed software and protocols. [2] These challenges will be covered in more detail in later sections.

3.2 Hierarchical Mobile Edge Computing

In a paper published in 2017, Kiani and Ansari propose a hierarchical model for mobile edge computing called Hierarchical Mobile Edge Computing (HI-MEC). This model utilized different tiers of cloudlets, a computer which is trusted, has good connectivity, and is resource rich [10], to offer lower latency for mobile users. The model proposes that there should exist three tiers of cloudlets, field, shallow, and deep, where cloudlets further from the mobile users have more resources associated with them. [3]

The HI-MEC architecture aims to efficiently distribute the workloads from mobile users to different tiers of cloudlets. In case the user demand is greater in the field cloudlet than what its resources can handle, the workload can be moved up in the chain to higher level cloudlets to take care of. This system of delegating workloads to different cloudlets helps to manage demand fluctuations in the network. [3]

The paper suggests that service providers should consider an auction-based pricing model to offer more flexibility. This is important in the sense that well designed pricing schemes can benefit both the users and providers as they could bring more users while for the user it can mean cheaper prices. Moreover, for mobile users, HI-MEC can provide opportunities for offloading certain mobile applications to cloudlets which in turn can have, as discussed earlier, an impact in their device's battery

life. Furthermore, the paper found that the proposed hierarchical model could efficiently allocate resources to MEC network. [3]

3.3 MIFaaS

Mobile-IoT-Federation-as-a-Service (MIFaaS) is a model proposed by Farris et al. in their 2017 paper that aims for more efficient utilization of resources in a pool of heterogeneous devices. As the user requirements can differ, so do the resources that certain devices can offer. [1] For example, a smart phone may have a great internet connectivity, but it may lack processing power and storage whereas a smart car could house more storage and better computational resources. Mobile devices of the passengers could then lend their network access to the smart car and the car could borrow its processing power to the smartphones.

Moreover, the paper suggests that instead of the typical device-oriented models, devices could participate in a formation of federations, where all the resources of participating devices would be shared. Although participating devices have their self-interests involved, the paper explains that by utilizing of game theoretic model with Nash-stable solution the MIFaaS model can deliver better results when comparing to device-oriented solutions where little to no cooperation exists. [1]

4 Challenges

Although the edge, fog, and the showcased models show enormous potential in resource sharing capabilities and reduced latency, there are certain challenges that must be overcome before full advantages of these technologies can be realized. The challenges faced can be divided into technical and social challenges, where technical challenges consist of physical, or software related issues and social challenges are related to incentive schemes. In the next two subsections, this paper aims to delve deeper into the challenges these technologies are facing.

4.1 Technical Challenges

Although POCs of the showcased models and frameworks have been successful, there are still technical challenges to overcome. For example, software and protocols should be designed to be compatible with devices regardless of the vendor and solutions regarding security and privacy need

to be robust since IOT devices may contain and process user related data [1], [2]. Moreover, a that persist but which is intertwined with incentive challenges, is device discovery. Resource sharing being the fundamental building block of UPI frameworks is dependent on the heterogeneous pool of devices. A lack of, for example storage resources, could mean that certain devices would not gain anything from participating in a cooperative resource sharing scheme.

4.2 Incentive Challenges

While resource sharing frameworks offer enhanced user-experiences [2], [1], [3], the problem of convincing users to adapt these technologies exists. As these models for resource sharing might degrade the users own user-experience by, for example, reducing bandwidth of the network connection, introducing additional costs regarding network connectivity, and increase the energy consumption with energy scarce devices, it does not make it appealing for users [11]. Furthermore, if users are not fairly compensated for participating in the resource sharing pools, they might be left with the feeling that they are not receiving enough resources in exchange for their resources. Moreover, clients, hosts, and services providers may have conflicting interests [11].

A significant challenge will turn out to be convincing users to adopt the technology. This adaptation may be aided by implementing the technology so that it already exists in the user's devices from the moment they purchase them. If the benefits offered are clear and the reward schemes are see-through, users will be more willing to get involved and start using the technology. Furthermore, blockchain technology could be utilized for creating trustworthy rewarding schemes with traceable history of resource sharing interactions.

5 Discussion

The fog computing paradigm has developed significantly during the past decade after the term fog was first introduced [8]. Since that, several models have been proposed to take advantage of the promises fog computing aims to deliver. The models and frameworks displayed earlier are examples of how these technologies can be utilized.

Although the displayed models overlap with the technologies used,

they are aimed at solving different challenges. The 3C resource sharing framework introduces a model for increasing the number of different resources which participating devices could leverage. The paper about HI-MEC model proposes a hierarchical tiering model for several levels of cloudlets where tasks requiring heavier computational resources could be delegated to more powerful cloudlets. Furthermore, MIFaaS model suggests that devices could form federations to pool resources for sharing purposes. Therefore, the displayed models do not compete but rather complement each other's functionality and achievements.

All the displayed models show exciting potential for providing the resources needed for ever more powerful mobile machines. However, as of today, this field of technology still lacks major success regarding successful companies. Regardless of that, this field of networking has a great scientific foundation which offers excellent opportunities to build things on.

6 Conclusion

This paper's aim was to gather existing information about UPI in the style of a literature review and propose new directions for research and innovation. The paper displayed three proposed models for UPI utilization and discussed the challenges faced by different proposed models and frameworks. Although the models presented in this paper have significant potential behind them, there have not been seriously successful use-cases presented to be solved by fog paradigms. Therefore, this paper concludes that more POCs and innovative companies, focusing on fog computing and UPI, are necessary steps for the future. Moreover, this paper strongly encourages to explore different incentive schemes for UPI models since without a reason for users to take part in these models they are useless. Models to explore and study include, for example, blockchain technology. Lastly, due to the nature of the resource sharing in UPI models, it is necessary that more studies are conducted on security and privacy issues as compromises in this area could severely hinder users trust in the technology.

References

[1] I. Farris, L. Militano, M. Nitti, L. Atzori, and A. Iera, "Mifaas: A mobile-iot-federation-as-a-service model for dynamic cooperation of iot cloud providers," *Future generation computer systems*, vol. 70, pp. 126–137, 2017.

[2] M. Tang, L. Gao, and J. Huang, "Communication, computation, and caching resource sharing for the internet of things," *IEEE Communications Magazine*, vol. 58, no. 4, pp. 75–80, 2020.

[3] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE internet of things journal*, vol. 4, no. 6, pp. 2082–2091, 2017.

[4] Gartner, "Gartner says cloud will become a business necessity by 2028." `https://www.gartner.com/en/newsroom/press-releases/2023-11-29-gartner-says-cloud-will-become-a-business-necessity-by-2028`, November 2023.

[5] P. Gokhale, O. Bhat, and S. Bhat, "Introduction to iot," *International Advanced Research Journal in Science, Engineering and Technology*, vol. 5, no. 1, pp. 41–44, 2018.

[6] G. Kaur and R. S. Batth, "Edge computing: Classification, applications, and challenges," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, pp. 254–259, 2021.

[7] M. K. Jain, "The rise of ott platform: changing consumer preferences," *EPRA International Journal of Multidisciplinary Research (IJMR)*, vol. 7, no. 6, pp. 257–261, 2021.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on mobile cloud computing*, pp. 13–16, ACM, 2012.

[9] M. Chiang, "Fog networking: An overview on research opportunities." Author is the Arthur LeGrand Doty Professor of Electrical Engineering, December 2015.

[10] S. Clinch, J. Harkes, A. Friday, N. Davies, and M. Satyanarayanan, "How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users," in *2012 IEEE International Conference on Pervasive Computing and Communications*, pp. 122–127, 2012.

[11] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, *Incentive Schemes for User-Provided Fog Infrastructure*, pp. 129–150. John Wiley & Sons, Incorporated, 2020.

Fighting the Art Theft Machine: Poisons and Perturbations CS-E4000 - Seminar in Computer Science

Perttu Niskanen
perttu.niskanen@aalto.fi
Tutor: Blerta Lindqvist

April 4, 2024

KEYWORDS: AI art, Data poisoning, Perturbation attack, Copyright, Generative models, Diffusion models

1 Introduction

The proliferation and high demand of Generative Artificial Intelligence (GenAI) tools within recent years has led to an arms race between their developers and the creators of their training data. The developers cannot afford to acquire the substantial training data legally and so they have begun to download it off the internet without the creators' consent [1]. In response, some creators have begun to poison their data in an effort to deter the developers, who have further developed defences against data poisoning. This conflict is especially present in regards to AI art.

The negative effects of AI art are clear. Professional artists who have spent years developing their style have it mimicked without credit or compensation [2], ending their ability to earn a living. Synthetic art displaces original artists in search results, stunting their ability to advertise. Upcoming artists are demoralized from training, as art students see their potential careers replaced by AI models [3]. The U.S. Copyright office only recognizes copyright in works "created by a human being" [4], leaving the artists without legal precedent or protection. As such, artists who post their content on the internet are concerned about their works being used without their consent to develop the technology that intends to replace them [5].

To combat this, several tools have been developed that modify images in subtle ways to poison the GenAI that trains on them, corrupting the images they generate. This paper reviews some of these AI art poisoning tools, their methods, and their effectiveness.

This paper is organized as follows. Section 2 presents background information relevant to the subject. Section 3 describes *Glaze*, a text-to-image data poisoning tool developed by the Glaze Team. Section 4 describes *Nightshade*, the second text-to-image data poisoning tool by the Glaze Team. Section 5 provides discussion on the effectivenesses of both tools. Finally, section 6 provides concluding remarks.

2 Background

2.1 Text-to-image Generation

Text-to-image generation happens in two phases: model training and image generation. During training, a training image x is run through a feature extractor Φ to produce the extracted features $\Phi(x)$. Simultaneously, the corresponding training prompt s is run through a conditional image generator G to produce a predicted feature vector $G(s)$. The parameters of G are optimized such that the text feature vector $G(s)$ matches the image feature vector $\Phi(x)$. At runtime, a user gives G a generation text prompt s_0, which then outputs an image feature vector $G(s_0)$. A decoder D then decodes $G(s_0)$ to produce the final generated image. [3]

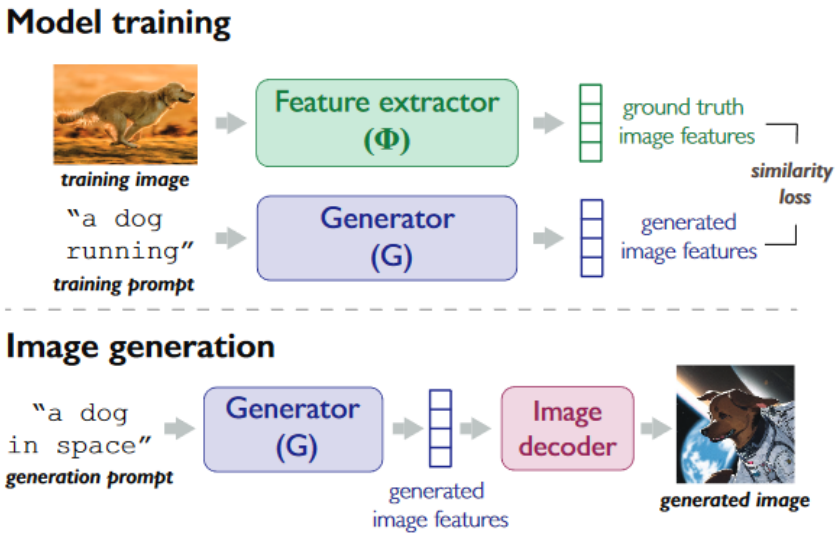


Figure 1: High level architecture of a text-to-image model. [3]

Text-to-image generation is most commonly done by Diffusion Models (DMs). These are parameterized Markov chains trained such that the transitions of the chain reverse a diffusion process, in effect denoising an image by predictive steps. By starting the process from pure noise, the model generates

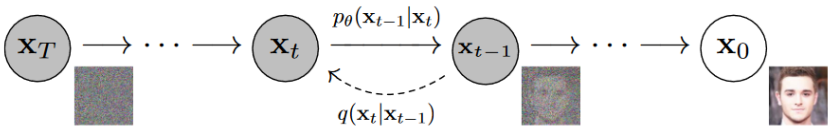


Figure 2: Graphical representation of a Diffusion Model. [6]

new samples similar to the data used to train it. Figure 2 shows the transitions from $\mathbf{x}_T$ (pure noise) to $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ (partially denoised sample), and finally to $\mathbf{x}_0$ (generated sample). [6]

2.2 Data Poisoning

Data poisoning is a method of manipulating the training data of a Machine Learning (ML) model to induce unexpected behaviors once the model has been trained. It is used as a form of sabotage against models using web-scraped data, although its effectiveness is disputed as it may be causing a false sense of security to the attackers. [7]

One of the major vectors of data poisoning is via perturbation.

2.2.1 Perturbation

At their core, most instances of data poisoning are variations of perturbation attacks, in which data is modified by small increments so as to affect a ML model's output. Perturbation attacks aren't inherently hostile towards the overall functionality of an AI model; they may be used simply to test a model's robustness. For example, Su et al. [8] found that the outputs of a Deep Neural Network (DNN) could be altered even by changing a single pixel in an input image.

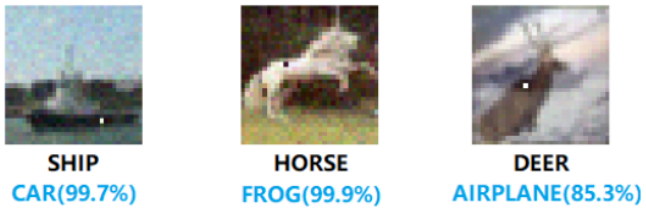


Figure 3: An optimally placed one-pixel perturbation causes a DNN to confidently misclassify an image of a ship as a car, etc. Adapted from [8].

Randomly perturbed (or "noisy") data is often used to increase a ML model's robustness during training. It can effectively multiply the amount of training data and reduce overfitting (or "memorizing").

Intentionally perturbed data, however, seeks to maximize a model's error, thus enabling data poisoning.

2.2.2 Adversarial Attack Methods

Most models are trained to work on a specified problem set with the assumption that the training data used is independent and identically distributed. This assumption is violated when the data is gathered from potential attackers who may intentionally provide fabricated or perturbed data. This manipulation of data is called an adversarial attack.

Availability attacks leverage neural models' substantial demand for training data, in which attackers simply leave poisoned data where they expect it to be collected. For example, in 2016, Microsoft released their chatbot Tay, which was designed to interact with and learn from other users on Twitter (now X). This led to attackers feeding it hateful rhetoric and "teaching" it to sympathise with Nazis, support genocide, et cetera [9]. In the case of AI art data poisoning specifically, an attacker will upload a poisoned image onto any image hosting website, whereafter it will be scraped, automatically tagged, and used to train an image generation model. This process is visualized below in Figure 4.

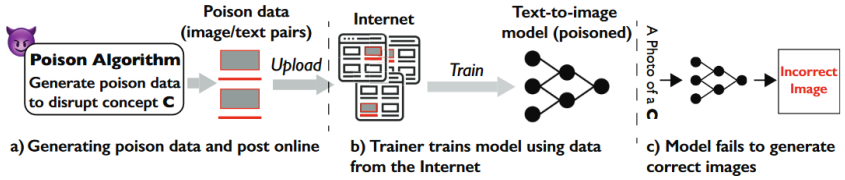


Figure 4: Overview of an availability attack. **(a)** Attacker poisons data and uploads it; **(b)** Model trainer scrapes data to train their generative model; **(c)** Poisoned model generates incorrect image. [10]

In a process similar to intentional availability attacks, the tentatively-named phenomenon Model Collapse can also poison data sets (other names include Habsburg AI [11], Model Autophagy Disorder [12], and Nepotistically Trained AI [13]). This occurs when a GenAI is trained on data generated by another GenAI. [14]

While an adversarial attack may intend to poison training data to induce overall unreliability in a ML model, backdoor attacks (or "trojans") seek to maintain the model's performance unless presented with an input containing a "trigger" that produces some predetermined effect. A conceptual overview of this attack is shown in Figure 5.

Backdoor attacks are a type of targeted attack, in that they seek to change the behavior of a model on particular inputs. An untargeted attack, however, seeks to indiscriminately affect a model's behavior [16]. Backdoor attacks are also a type of subpopulation attack, in that they seek to affect a model's behavior with a specific subpopulation, while leaving behavior unaffected for the rest of the data [17].

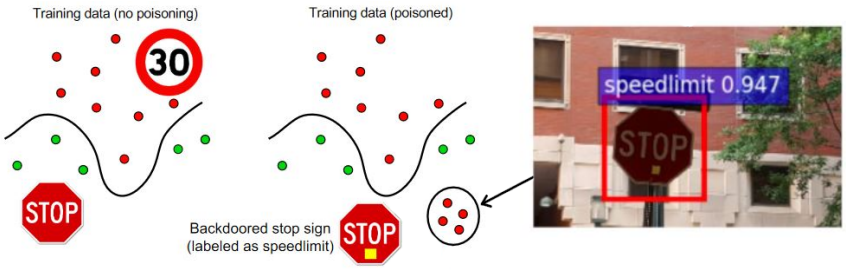


Figure 5: Conceptual representation of a backdoor attack. Backdoor attacks place mislabeled training points in a region of the feature space far from the rest of the training data, allowing them to be invoked with a specific trigger at runtime. Adapted from [15].

3 Glaze

This section covers *Glaze*, an AI data poisoning tool created at Chicago University by Shan et al. [3]

Glaze, initially released in March 13, 2023 and fully in June 23, 2023, was among the first tools created to combat AI art theft, and quickly gained press coverage for both itself and the questionable morality of AI art. It was designed as a way to combat the ability of popular text-to-image diffusion models like MidJourney to mimic the art styles of specific artists. It would "cloak" the art with barely perceptible perturbations which, when used as training data, would mislead the generative model as to the artist's style.

The central problem with masking an artist's style is quantifying mathematically what an "artistic style" even is. To facilitate this, *Glaze* leverages "style transfer," a class of algorithm that transfers the art style of one image onto the contents of another image. The style-transferred image can then be used as a projection target for perturbation computation, which essentially "aims" the style-specific features of the original image towards that of the style-transferred image through perturbations.

Using this method, the computation is as follows. Given an artwork x, a pre-existing feature extractor Ω is used to compute a style-transferred version of x onto target style T: $\Omega(x, T)$. Then, a style cloak (or "glaze") δ is computed, such that δ moves x's style-specific feature representation to match that of $\Omega(x, T)$ while minimizing visual impact. The poisoned image $x^p = x + \delta$ is found by optimizing the following:

$$\delta := \min_{\delta} Dist(\Phi(x+\delta), \Phi(\Omega(x,T))), \quad \text{subject to } |\delta| < p, \tag{1}$$

where Φ is a generic image feature extractor, $Dist(\cdot)$ computes the distance of two feature representations, $|\delta|$ measures the perceptual perturbation caused by cloaking, and p is the perturbation budget (using the Learned Perceptual Image Patch Similarity (LPIPS) metric).

Figure 6: Example of style-transferred artwork with different target styles. Original artwork by Karla Ortiz. Adapted from [3].

Since the use of a style-transferred image $\Omega(x, T)$ guides cloak optimization to focus on perturbing style-specific features, making sure that the target style T is as dissimilar from the original style as possible maximizes cloak efficacy.

The *Glaze* system works in three steps: choosing a target style, transferring that style, then computing cloak perturbations. Given a victim artist V, *Glaze* takes as input a set of V's artwork X_V, a feature extractor Φ, a style-transfer model Ω, and a perturbation budget p. In many cases, a single model (e.g. Stable Diffusion) provides both Φ and Ω.

First, *Glaze* randomly selects T from a set of candidate styles in the public domain (e.g. Picasso, Van Gogh). For each candidate style, it selects a few images and calculates their feature space centroid with Φ. It also calculates V's centroid with Φ using X_V. Then, it selects one T that is sufficiently distant from V.

Second, *Glaze* uses a pre-trained style-transfer model Ω to generate the style-transferred artwork $\Omega(x, T)$ for each art piece $x \in X_V$.

Last, *Glaze* computes a cloak perturbation δ_x for each x, following Equation 1. These perturbations are then applied to these images, and the artist is free to upload them to the internet safely, the Glaze team claims. [3]

4 Nightshade

Developed by the Glaze Team at the University of California, *Nightshade* was received with high demand, netting 250,000 downloads within 5 days of its release [18] . Whereas *Glaze* uses backdoor subpopulation attacks to disrupt the fine-tuning of a local diffusion model, *Nightshade* aims to corrupt the base diffusion model and render it useless for all of its users [10].

Poisoning attacks, in a general sense, cause predictable misclassifications and demand that at least 20% of the training set's samples are poisoned. This proves problematic when dealing with non-fine-tuned diffusion models, which can have up to billions of training samples. A common assumption suggests that poisoning such a model would require millions of poisoned samples, making

it infeasible in practice. However, the Glaze team found that these general text-to-image models are actually highly vulnerable to prompt-specific poisoning attacks.

While general text-to-image models have up to billions of total samples, the Glaze team found that the number of samples associated with any specific concept or prompt is only in the order of thousands. They introduce two terms: concept sparsity, or the number of training samples associated explicitly with a specific concept (e.g. dragon); and semantic sparsity, or the number of training samples associated with a concept and its semantically related terms (e.g. dragon → wyrm, wyvern, Smaug, J.R.R. Tolkien, etc.).

Concept sparsity introduces a vulnerability to general models. For example, the training set including the prompt "dog" might account only for 0.1% of the total training set. In other terms, to corrupt the image generation on a benign concept $\mathcal{C}$, an attacker needs only to inject a sufficient amount of poisoned data into the training set to offset the contribution of $\mathcal{C}$'s clean training data and its semantically related concepts. In LAION-Aesthetic, a popular open-source dataset for training text-to-image models, over 92% of represented concepts were associated with less than 0.04% of the total samples, or 240K images. Furthermore, 92% of concepts were semantically linked (or "were synonyms to") to less than 0.2% of samples. The comparatively small conceptual subpopulation sizes makes poisoning them much more feasible.

A simple dirty-label attack aiming to poison a concept $\mathcal{C}$ works as follows. The attacker first chooses a destination concept $\mathcal{A}$. They then create a number of text descriptions $\mathbf{Text}_{\mathcal{C}}$ that contain the word $\mathcal{C}$ and do not contain the word $\mathcal{A}$. They also create a number of images $\mathbf{Image}_{\mathcal{A}}$ that contain visual elements of $\mathcal{A}$ and no visual elements of $\mathcal{C}$. Lastly, they pair the image descriptions $\mathbf{Text}_{\mathcal{C}}$ with the images $\mathbf{Image}_{\mathcal{A}}$.

This method has two major inefficiencies: it is easily automatically detected and the "strength" of the poison is fairly low due to the natural heterogeneity of images $\mathbf{Image}_{\mathcal{A}}$ due to being likely to include visual representations of other concepts than just $\mathcal{A}$. The first issue is solved through the implementation of optimized perturbations, and the other through generating image $\mathbf{Image}_{\mathcal{A}}$ with a GenAI using a prompt such as "a picture of $\mathcal{A}$".

Given the generated images of $\mathcal{A}$, hereafter referred to as "anchor images," perturbations can be used to make them seem identical to images of $\mathcal{C}$. Let t be a chosen text prompt $\mathbf{Text}_{\mathcal{C}}$ and x_t the corresponding natural image $\mathbf{Image}_{\mathcal{C}}$. An optimized poison image for t, or $x_t^p = x_t + \delta$ can be found by optimizing the following:

$$\delta := \min_{\delta} Dist(F(x_t + \delta), F(x_a)), \quad \text{subject to } |\delta| < p, \tag{2}$$

where x_a is a DM-generated anchor image, $F(\cdot)$ is the image feature extractor of the text-to-image model being attacked, $Dist(\cdot)$ computes the distance of two feature representations, $|\delta|$ is the perceptual perturbation added to x_t, and p is the perturbation budget using the LPIPS metric. A general image feature extractor Φ can also be used, although the Glaze team claims that the attack

success rate may fall anywhere from 1% to 24% depending on the combination of model architectures. [10]

Figure 7 shows an example of poison data curated to corrupt the concept "dog" ($\mathcal{C}$) using "cat" ($\mathcal{A}$).

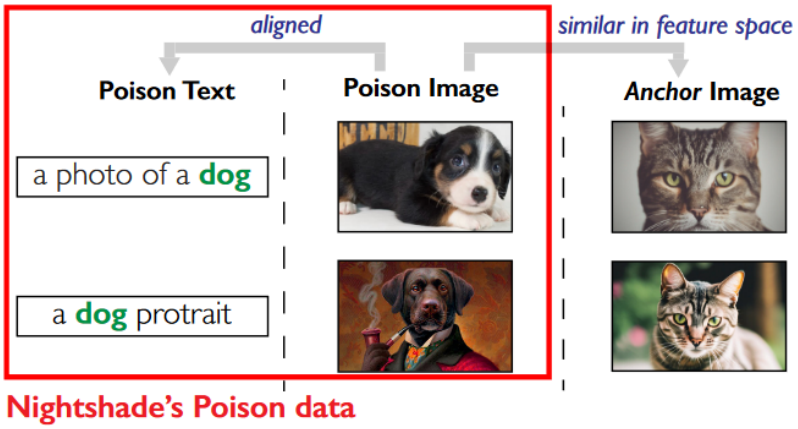


Figure 7: Illustrative example of *Nightshade* poisoning the concept "dog" using "cat". Anchor images (right) are generated by prompting "a photo of a cat" using a DM. The poisoned images (middle) are perturbed versions of natural images of "dog," which resemble the anchor images in feature space. [10]

At runtime, *Nightshade* outputs $\{\text{Text}_p/\text{Image}_p\}$, a set of N_p poisoned text/image pairs using the following resources and parameters: $\{\text{Text/Image}\}$, a collection of N natural text/image pairs related to $\mathcal{C}$, where $N >> N_p$; $\mathcal{A}$, a concept semantically unrelated to $\mathcal{C}$; M, an open-source text-to-image generative model; M_{text}, the text encoder of M; and p, a perturbation budget.

First, *Nightshade* selects the poison text prompts $\{\text{Text}_p\}$. It uses the text encoder M_{text} calculates the cosine similarity of text prompt t with $\mathcal{C}$ in the semantic space, or: $CosineSim(M_{text}(t), M_{text}(\mathcal{C}))$, $\forall t \in \{\text{Text}\}$, then chooses a random sample of N_p text prompts from the 5,000 top-ranked results to form $\{\text{Text}_p\}$.

Second, it generates anchor images based on $\mathcal{A}$. It queries the text-to-image generator M with "a photo of $\mathcal{A}$" if $\mathcal{A}$ is an object, or "a photo in the style of $\mathcal{A}$" if $\mathcal{A}$ is a style, generating N_p anchor images to form $\{\text{Image}_{anchor}\}$.

Last, it constructs poison images $\{\text{Image}_p\}$ and pairs them with text prompts in $\{\text{Text}_p\}$. For each text prompt $t \in \{\text{Text}_p\}$, its natural counterpart x_t is located from $\{\text{Image}\}$. Then, with a random anchor image x_a from $\{\text{Image}_{anchor}\}$, the perturbed image $x_t^p = x_t + \delta$ is generated with Equation 2. Afterwards, the text/image pair t/x_t^p is added into the poison dataset $\{\text{Text}_p/\text{Image}_p\}$, the expended anchor image x_a is removed from the anchor set $\{\text{Image}_{anchor}\}$, and the next text prompt in $\{\text{Text}_p\}$ is processed.

5 Discussion

The Glaze team reports that glazing was invisible to image captioning models, likely because the perturbations focused on style-specific elements, whereas captioning models focus on image contents. They also found that *Glaze* was largely resistant to two popular data poisoning countermeasures: Gaussian noise and image compression. These countermeasures aim to transform the input image with random noise, so as to "drown" the purposeful perturbations. Afterwards, these images can then be denoised or upscaled respectively. Artist-rated Protection Success Rate (PSR) fell from 92% without countermeasures to 89% with denoised Gaussian countermeasures and to 85% with upscaled JPEG compression countermeasures. *Glaze* was also successful when only a fraction of an artist's set of artworks was glazed: artist-rated PSR was 87% at 25% of artwork cloaked [3].

Figure 8: Example *Glaze* protection results. **1-2:** artist's original artwork; **3:** mimicked artwork without *Glaze*; **4:** style-transferred artwork using original artwork **1** as source, and name of target style (Oil painting by Van Gogh); **5-6:** mimicked artwork with *Glaze* using perturbation budget $p = 0.05$ or $p = 0.1$ respectively. Artwork by Karla Ortiz. Adapted from [3].

However, Liang et al. [19] found that *Glaze* could be subverted with a crop-resize input transformation, wherein they removed a 64 pixel wide border from a 512×512 resolution image, then resized it back to its original dimensions.

Nightshade attacks were successful with roughly 100 samples, less than the 20% of the simple dirty-label attack. As shown in Figure 9, *Nightshade* begins to show significant effects at just 50 poison samples and a high success rate at 200 samples using a Contrastive Language-Image Pre-Training (CLIP) classifier, a neural network used to classify images.

Contemporary text-to-image models take a significant amount of money to train from scratch—the first stable diffusion model took 150K GPU hours or $600K to train—and, therefore, new versions of these models are commonly trained continuously from previous versions. If a continuously trained model keeps adding clean data related to an already poisoned concept, the ratio of additional poisoned samples required was found to be roughly 2%. [10]

The Glaze team found that conceptual poisoning "bled through" to semantically related concepts and that these poison attacks were composable; when attackers poisoned "dog" to "cat" and "fantasy art" to "impressionism," a prompt with both "dog" and "fantasy art" generated an image with a cat in an impres-

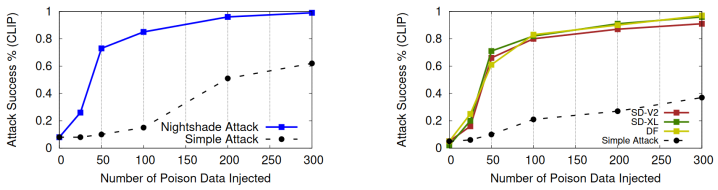


Figure 9: **1:** *Nightshade's* attack success rate (CLIP-based) vs. # of poison samples injected, compared to a simple dirty-label attack; **2:** *Nightshade's* attack success rate (CLIP-based) vs. # of poison samples injected, for various model architectures (continuous training) compared to a simple dirty-label attack. [10]

sionist style. Furthermore, as more concepts were poisoned, the overall performance of the model gradually fell. Image quality noticeably degraded at 250 poisoned concepts and devolved to random noise at 1000 poisoned concepts. [10]

In terms of visual impact on the perturbed images by *Glaze*, 92% of surveyed artists felt that the perturbations introduced by cloaking with a perturbation budget $p = 0.05$ were small enough, such that they were "willing" or "very willing" to post the perturbed images on their personal websites. *Glaze's* PSR at $p = 0.05$ was found to be roughly 93%. *Nightshade* used a standardized $p = 0.07$ throughout their tests (40% increase in perturbation budget compared to *Glaze*), although they claim on their website that both programs use the same perturbation budget. They also claim that the visual artifacts added by *Nightshade* are at most equivalent to those added by *Glaze*, and in most cases are harder to see [20]. In any case, since *Nightshade's* effects are cumulative, a user can lower the perturbation budget to their needs without compromising the overall functionality of the poison, only its efficiency.

Shan et al. [3] caution *Glaze's* users on the issue of future-proofing the tool: "Any technique we use to cloak artworks today might be overcome by a future countermeasure, possibly rendering previously protected art vulnerable." This quote is illustrative of the asymmetry in fighting against AI art mimicry; whether AI poisoning tools work now is largely irrelevant, since any GenAI developer can simply download poisoned images now and wait for the advent of some technology that bypasses the poison [7]. Indeed, *Glaze* was quickly bypassed with a crop-resize transformation. Although *Nightshade* hasn't been bypassed (as of January 2024 [20]), whether it or any other AI poisoning tool is broken is only a matter of time.

6 Conclusion

In conclusion, contemporary data poisoning methods are effective against contemporary GenAI. The perturbations introduced into the art pieces tend to be acceptably negligible by their creators while still being sufficiently corrupting so as to prevent the art from being used as training data. However, the excitement around these poisoning tools may be creating a false sense of security for artists, as the poisoned art they post on the internet may still be used as training data once the poisons are bypassed.

References

[1] M. M. Grynbaum and R. Mac, "The times sues openai and microsoft over a.i. use of copyrighted work," *New York Times*, 2023. [Online]. Available: https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html (visited on January 23, 2024).

[2] K. K. Ho, "Database of 16,000 artists used to train midjourney ai, including 6-year-old child, garners criticism," *Artnews*, 2024. [Online]. Available: https://www.artnews.com/art-news/news/midjourney-ai-artists-database-1234691955/ (visited on January 23, 2024).

[3] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao, *Glaze: Protecting artists from style mimicry by text-to-image models*, 2023. arXiv: 2302.04222 [cs.CR].

[4] C. T. Zirpoli, "Generative artificial intelligence and copyright law," Congressional Research Service, 2023. [Online]. Available: https://crsreports.congress.gov/product/pdf/LSB/LSB10922 (visited on January 23, 2024).

[5] H. H. Jiang, L. Brown, J. Cheng, *et al.*, "Ai art and its impact on artists," in *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, 2023, pp. 363–374. DOI: 10.1145/3600211.3604681.

[6] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, 2020. arXiv: 2006.11239 [cs.LG].

[7] E. Radiya-Dixit, S. Hong, N. Carlini, and F. Tramèr, *Data poisoning won't save you from facial recognition*, 2022. arXiv: 2106.14851 [cs.LG].

[8] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, October 2019, ISSN: 1941-0026. DOI: 10.1109/tevc.2019.2890858.

[9] J. Wakefield, "Microsoft chatbot is taught to swear on twitter," *BBC*, 2016. [Online]. Available: https://www.bbc.com/news/technology-35890188 (visited on January 30, 2024).

[10] S. Shan, W. Ding, J. Passananti, H. Zheng, and B. Y. Zhao, *Prompt-specific poisoning attacks on text-to-image generative models*, 2023. arXiv: 2310.13828 [cs.CR].

[11] J. Sadowski, *I coined a term on @machinekillspod that i feel like needs its own essay: Habsburg ai – a system that is so heavily trained on the outputs of other generative ai's that it becomes an inbred mutant, likely with exaggerated, grotesque features. it joins the lineage of potemkin ai.* February 13, 2023. [Online]. Available: https://twitter.com/jathansadowski/status/1625245803211272194 (visited on January 30, 2024).

[12] S. Alemohammad, J. Casco-Rodriguez, L. Luzi, *et al.*, *Self-consuming generative models go mad*, 2023. arXiv: 2307.01850 [cs.LG].

[13] M. Bohacek and H. Farid, *Nepotistically trained generative-ai models collapse*, 2023. arXiv: 2311.12202 [cs.AI].

[14] I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson, *The curse of recursion: Training on generated data makes models forget*, 2023. arXiv: 2305.17493 [cs.LG].

[15] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, December 2018, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2018.07.023. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2018.07.023.

[16] M. Goldblum, D. Tsipras, C. Xie, *et al.*, *Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses*, 2021. arXiv: 2012.10544 [cs.LG].

[17] M. Jagielski, G. Severi, N. Pousette Harger, and A. Oprea, "Subpopulation data poisoning attacks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3104–3122.

[18] C. Franzen, "Ai poisoning tool nightshade received 250,000 downloads in 5 days: 'beyond anything we imagined'," *VentureBeat*, 2024. [Online]. Available: https://venturebeat.com/ai/ai-poisoning-tool-nightshade-received-250000-downloads-in-5-days-beyond-anything-we-imagined/ (visited on January 31, 2023).

[19] C. Liang and X. Wu, *Mist: Towards improved adversarial examples for diffusion models*, 2023. arXiv: 2305.12683 [cs.CV].

[20] T. G. Project, *Frequently asked questions (faq)*, 2024. [Online]. Available: https://nightshade.cs.uchicago.edu/faq.html (visited on April 4, 2024).

Acronyms

AI Artificial Intelligence. 1, 5, 10

CLIP Contrastive Language-Image Pre-Training. 9, 10

DM Diffusion Model. 2, 3, 7, 8

DNN Deep Neural Network. 3

GenAI Generative Artificial Intelligence. 1, 4, 7, 10, 11

LPIPS Learned Perceptual Image Patch Similarity. 5, 7

ML Machine Learning. 3, 4

PSR Protection Success Rate. 9, 10

Industrial Control Systems and IEC 62443 from a perspective of Zero Trust

Petteri Pulkkinen
petteri.k.pulkkinen@aalto.fi

Tutor: Mikko Kiviharju

Abstract

Industrial Control Systems are used to control critical infrastructure. The state of ICS security has been under active research. At the same time, Zero Trust has become a popular concept to improve system security. This paper introduces the ICS environment with its specialities, discusses the definition of Zero Trust and analyses the IEC 62443 standard family from the perspective of Zero Trust. Zero Trust could be considered an asymptotic goal for security, which cannot be fully implemented. However, it can provide ideas to improve ICS security via micro segmentation, encryption and security automation.

KEYWORDS: Industrial Control Systems, ICS, Zero Trust, IEC 62443

1 Introduction

Industrial Control Systems (ICS) power the industry and production. ICS are used to control power plants and to control the electricity grid. They are used to controlling factories and other process instrumentation tasks. ICS are typically responsible for ensuring the safety in automated processes of factories [4].

The reasons behind weaker resistance against cyberattacks originates

from the special nature of ICS. Devices in the ICS field often include low-power embedded devices running real-time systems. For historical reasons, the computational power of ICS has been limited, and therefore had to be considered during the design phase. This has led to protocol various custom protocol designs where reliability has overtaken security.

To strengthen the security of ICS, security standards related to them have been improved and updated. At the same time with standards, legislation has also been updated to require the ICS to comply with the updated standards. The standards are still following the same architectural basis. Updated architecture models proposed, but implementing them is a long route in practice.

Standards and legislation guide towards the right direction, while technical debt of the systems is still remarkable. Renewing cycles of ICS is relatively slow, and usually happens during larger renovations of factories. The systems are still weak to defend against threats that intentionally want to cause damage. On the datacenter side, multiple solutions have been developed to strengthen the security of the system. These principles could probably be used to improve the security of ICS as well.

Zero Trust [9, 10] is one of the proposed concepts to strengthen security by architectural decisions. Implementations of Zero Trust principles have been proposed for enterprise networks [12]. NIST has also required that systems part of the critical infrastructure must follow the Zero Trust Architecture [9].

This research focuses on how IEC 62443 standard [2] compares with principles of Zero Trust [4]. As a second research question, this research analyses how the standard benefits or could benefit about using Zero Trust.

This research paper begins with an introduction to industrial control systems in section 2 and continues introducing the security characteristics of an ICS system in section 3. Next, the IEC 62443 standard family is introduced in section 4 and Zero Trust in section 5. After these introductory sections, this paper continues analysing IEC 62443 from the perspective of Zero Trust in section 6. Section 7 continues the discussion between the principles of Zero Trust and IEC 62443. Conclusions are presented in section section 8.

2 Industrial Control Systems

Industrial Control Systems (ICS), are used to control processes in industrial production plants, factories and other buildings. The purpose of an ICS is to ensure the process keeps running within configured efficiency and safety margins. Traditionally, computer systems are divided into operational technology (OT) and information technology (IT), where the latter category is probably more widely understood than the first one. In IT, data and information are in the key role, while OT focuses more on the process. In general, outages in services can somehow be accepted in IT, while it is unacceptable in OT. This fundamental difference has led to differences in the architecture of ICS.

The nature of ICS poses more strict requirements on controller hardware compared to IT systems. In ICS and OT in general, operations must be handled in real-time. Currently, the majority of IT hardware, on the other hand, uses scheduling as a standard way to improve resource allocation and therefore utilization, but it loses some execution guarantees. In ICS, additional and potentially unpredictable delay is often unacceptable, as the control system should act promptly upon receiving signals from sensors. However, the signal processing in ICS is typically not that computationally expensive. ICS often utilizes microcontrollers which have better IO capabilities. Microcontrollers are typically programmed using C/C++, as the constraints from the computational power poses higher demand for highly optimized software implementations.

However, using lower abstraction level languages with manual memory management has led to vulnerabilities in the software. In C and C++, proper memory management is left as a responsibility of the programmer. History has shown that mistakes happen, no matter how well the software is being tested and reviewed. Improper memory management is currently the most common reason behind security vulnerabilities [6]. On the IT side, multiple protection mechanisms, e.g., canary stacks and branch protection, are actively used to cope with this existing problem. However, most microcontrollers do not provide mechanisms at the hardware level and also lack the computational power required to implement the protections in the kernel.

Microcontrollers provide a relatively easy and cost-effective approach of controlling static processes, but production optimization might become difficult relatively soon. Optimizing processes often requires more com-

putational resources, and switching from a real-time system to a scheduling system would provide greater utilization efficiency of resources. This leads to a natural need to connect multiple systems together by forming communication networks. The protocols and medium used for communication in ICS vary from serial to Ethernet-based solutions.

3 Security of Industrial Control Systems

Security of ICS has been a popular area of research and has been rising during the past few years. Not only factories and production plants are important for the economy, but ICS are also used to maintain critical infrastructure such as water and electricity supply pipelines and grids. Such systems have been chosen as targets while attacking against the country and society.

The complexity of control systems in industry has expanded with the development of technology. ICS form networks of computers and sensors interacting with each other [7]. At the same time, supply chains of the products used as building blocks of ICS systems, in both hardware and software side, have become broader. On the IT side, development in networking infrastructure has offered new possibilities for centralized computing and placing servers within datacenters and longer distances away from the production plants. Parts of these aspects have also been adopted on the OT side.

Microcontrollers and the programs written in C/C++ make the inner core of ICS fragile. Historically, the systems have been isolated as no there was no need for external communication. However, internet connectivity has opened new opportunities. Especially on the IT side, the internet connectivity has become almost an essential part, empowering multiple successful features. Adopting those features also on the OT side and within ICS requires connecting the production plants to the internet. At the same time, the connectivity has opened new opportunities as well for the attacker. As previously mentioned, ICS often controls critical infrastructure, which for the attacker shows up with increased interest compared to an average IT system.

Lack of natural motivation postpones security improvements in ICS. Typically, the owners of the control system value operational stability over security. They want to minimize downtime and the reliability, as down scaling and closing the production line causes straight declines to their

economy. Production plants are also expensive investments that are expected to be running for years after opening. Not scarifying the production performance for probabilistic security threats delays new security patches and features from taking into daily use. From the perspective of a single production plant, addressing the security threats by strengthening the security of the ICS might not seem that appealing. However, from the perspective of the society, preventing the outages of resource supplies is a challenge that needs to be addressed.

4 IEC 62443

IEC 62443 introduces a concept of zones and conduits to help to model the ICS environment [3]. The infrastructure can be divided into zones that logically group assets having similar security requirements. After grouping the assets logically, the dependencies between zones can be modeled by connections. The connections with similar characteristics can be grouped into a conduit, which then simplifies the architectural picture. IEC 62443 3.2 [3] defines a standardized approach of dividing the infrastructure into zones and conduits.

IEC 62443 security requirements and security levels are defined in part 3.3 [2]. Requirements are grouped into 7 foundational requirements and numbered from FR1 to FR7. FR1 focuses on identification and authentication control and FR2 focuses on use control, i.e., authorization. FR3 focuses on system integrity and protection mechanisms against system manipulation. FR4 focuses on data confidentiality at rest and in transit while FR5 focuses on restricted data flow, i.e., preventing data circumventions outside designed zones and conduits. FR6 focuses on timely response to events and defines requirements on how to respond to security incidents. FR7 focuses on resource availability under normal and abnormal conditions.

The requirements of IEC 62443 family are considered from a perspective of four security levels ranging from SL1 to SL4[2]. SL1 aims to protect against passive attackers. SL2 improves the security level by providing protection against active attackers but with low resources and skills, mainly protecting against eavesdropping and casual exposure. SL3 enforces more strict requirements and provides protection against active attackers with moderate resources, IACS-specific skills and moderate motivation. The highest SL4 aims to provide protection against a highly

motivated attacker with extended resources.

Each system requirement (SR) under foundational requirements has been assigned to a certain security level. If that system requirement is fulfilled, the system reaches the specified SL. SRs also have requirement enhancements (RE), which define more strict requirements to reach higher SL. During security assessment, the resulting SLs from all the SRs can be analyzed and the SL of the system is the lowest SL of these results.

The next section introduces a more abstract approach to analyze the security, Zero Trust.

5 Zero Trust

Zero Trust is a relatively new term, although the concept has been discussed decades before that term. The main motivation behind Zero Trust are changes in intrusion strategy used by the attackers. Perimeter-based defense strategies have been implemented utilizing firewalls, placing infrastructure behind strictly controlled perimeters. This has worked with the original intention to make direct intrusion difficult. However, direct intrusion is not the only way in. The perimeters defend attacks from the outside of an organization, but does not necessarily detect attacks originating from within the perimeter.

Around 2014, Google published a research blog post series of new architectural approach for enterprise security[12]. They proposed a new approach and demonstrated new defense mechanisms for a hypothetical company called Beyond Corp. This was not the first publication in this area, but one of the most comprehensive ones. In 2018, the term Zero Trust was introduced to a wider audience, NIST[9]. In their special publication, they proposed concepts to improve security in organization IT infrastructure. Zero Trust became popular as a term in the defense sector, organizations, e.g., NCSC [8], have written their publications. Despite publicity around this topic, the term Zero Trust lacks a clear definition. The publications are not discussing with each other, but rather introduce their proposals. However, all of them are improving the security with similar ideas.

Because of multiple competing publications, using the term Zero Trust Architecture can be biased. Instead, it would be better to define Zero Trust as a set of principles rather than architecture. In 2022, Syed et al. conducted a relatively comprehensive analysis of the definitions of the

term Zero Trust [10]. Zero Trust builds on top of a change in the foundational assumptions. Accidents will eventually happen, no matter how well the defensive plans are. But when it happens, the aim is to limit the scope of damage. This radius can be controlled with de-perimeterization, which leads to the most important principle of Zero Trust. In Zero Trust, four key principles can be recognized from the definitions: micro segmentation, authentication and access control, encrypted communication and security automation.

Instead of building one considerable perimeter, the interior could be micro-segmented into different departments. To effectively isolate these micro-segments, improved authentication and authorization are needed to let the intended interaction to pass the segments. When only authenticated and authorized entities can access the resources, the next step is to ensure confidentiality by applying encryption for the data in transit. These improvements hinder the attackers to proceed in the infrastructure. The fourth key principle in Zero Trust is to improve security automation to detect incidents as early as possible so that they can be reacted earlier, again limiting the consequences they could cause.

Instead of an architecture example, Zero Trust could be considered a set of ideas and not as an implementation example. The next section analyses IEC 62443 standard from this perspective.

6 IEC 62443 from a perspective of Zero Trust

This section analyses system requirements (SR) of IEC 62443 [2] from a perspective of zero trust principles introduced in section 5.

6.1 Micro segmentation

IEC 62443 models segmentation utilizing its zones and conduits model. The sizes or quantities of zones in an ICS is not strictly defined. The standard proposes to start with larger segments and split them into smaller ones if the target SL is not met with larger segments. Zero Trust recommends micro segmentation, which aims further from this. Asymptotically, each service should be placed in its segment. Proceeding towards this goal becomes relatively cumbersome with the abstractions of zones and conduits, as doubling the number of zones often leads to rising the number of conduits to its square.

6.2 Authentication and Access control

Authentication-related requirements are grouped into FR1 and authorization into FR2. IEC 62443 copes well with the Zero Trust idea in this field, when the SL is increased to SL3 and SL4. For example, SR 1.2 specifying identity of entities allows identification based on physical location in SL1 and SL2. This is against the idea of Zero Trust, which, e.g., NIST clearly mentions in their publication [9]. On SL3 and SL4, components are required to be uniquely identifiable, which suggests to decouple identities from locations. IEC 62443 also requires the possibility to override authentications in case of emergency. The standard does not fully state, how this override should be implemented. From the perspective of Zero Trust, this kind of overrides should not exist, but on the other hand, the overrides could be implemented via different authentication method.

6.3 Encrypted communication

Communication encryption does not have dedicated FRs, but some requirements related to it are discussed in FR4 and FR5. Encryption in transit itself is not strictly mandated in the system requirements, but especially the SR4 enhancements aiming to SL3 and SL4 recall data protection in-transit when transferred via untrusted network and in SL4 when crossing zone boundaries. From the perspective of Zero Trust, all networks should be considered untrusted to be better prepared for attacks originating inside the organization. The IEC 62443 suggests implementing encryption via encrypted tunnels when implementing conduits. This indeed improves the security of the modeled system. However, applying micro segmentation, leads also segmenting the conduits, which effectively leads to individually encrypted connections. Encryption for individual connections is not recommended. Even session integrity protections are not recommended for every single connection within ICS.

6.4 Security automation

IEC 62443 mentions security automations in FR6, more specifically in SR 6.2 of Continuous monitoring. The requirements for this are relatively broad, leaving more freedom for the implementation. This can include intrusion detection systems, some other intrusion prevention systems or malicious code protection tools [2]. The idea behind this requirement is to

improve attack detection. Similar systems are also an important part of Zero Trust, as architectural aspects only hinder the possible attacks and not eliminate them. Because of the freedom given by the requirements, security automation might be the easiest area to improve security with the ideas provided by Zero Trust.

7 Discussion

Zero Trust is an abstract set of ideas to improve the security of a system. It can be seen to describe an asymptotic goal for a secure system, and reaching is completely can be seen impossible. It can be helpful to identify further enhancements in an existing system to protect it better against attacks originating from inside the out most perimeter. Supply chain attacks are increasingly utilized by the attackers to gain the initial foothold in the system. Recently discovered supply chain injection in generally used zx compression library is a good example of such [1].

Architectural enhancements are usually stronger defense mechanisms against attacks than active monitoring. Architectural changes are slow to be implemented, but standards can be used to require them in new installations. Having analyzed the IEC 62443 standard from the perspective of Zero Trust, it already requires a lot from the system. However, security is always a game against the attackers and improvement is seldom a bad idea. Further-applying the idea of micro segmentation and requiring encrypted channels more often could be one direction. There has already been discussion about encryption in ICS communication channels, but at least in research conducted in 2017 it was not recommended [5]. However, improvements have been seen in protocol design and widely used protocols, such Profinet, currently have encryption options available [11]. Maybe the question about encryption could be re-evaluated utilizing the resent research knowledge.

8 Conclusion

IEC 62443 already implements many features recommended by Zero Trust. The race against the attackers is never finished, and there is always room for improvement. The improvements can be easily made in the field of security automation, i.e., improving intrusion detection systems, to detect

the attacks at their early state. However, architectural enhancements are often more effective in the longer term. Updating the requirements, e.g., the IEC 62443 standard itself, is a relatively powerful way to improve the security of newly deployed ICS. Especially, the zone-conduit model can be enhanced to shrink zones smaller. Moreover, conduits and especially encryption-related requirements could be re-evaluated as the technology has developed since the publication. Reaching the asymptote of Zero Trust is not a reasonable target, but taking already a step towards it improves the protection of the system.

References

[1] CVE-2024-3094. https://nvd.nist.gov/vuln/detail/CVE-2024-3094.

[2] EN IEC 62443-3-3:2019 Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels, July 2019.

[3] EN IEC 62443-3-2:2020 Security for industrial automation and control systems - Part 3-2: Security risk assessment for system design, September 2020.

[4] Deval Bhamare, Maede Zolanvari, Aiman Erbad, Raj Jain, Khaled Khan, and Nader Meskin. Cybersecurity for industrial control systems: A survey. *Computers & Security*, 89:101677, 2020.

[5] Davide Fauri, Bart de Wijs, Jerry den Hartog, Elisa Costante, Emmanuele Zambon, and Sandro Etalle. Encryption in ICS networks: A blessing or a curse? In *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 289–294, October 2017.

[6] Ryan Levick and Sebastian Fernandez. We need a safer systems programming language. https://msrc.microsoft.com/blog/2019/07/we-need-a-safer-systems-programming-language/, July 2019.

[7] Stephen McLaughlin, Charalambos Konstantinou, Xueyang Wang, Lucas Davi, Ahmad-Reza Sadeghi, Michail Maniatakos, and Ramesh Karri. The Cybersecurity Landscape in Industrial Control Systems. *Proceedings of the IEEE*, 104(5):1039–1057, 2016.

[8] NCSC. Zero trust architecture design principles. https://www.ncsc.gov.uk/collection/zero-trust-architecture.

[9] VA Stafford. Zero trust architecture. *NIST special publication*, 800:207, 2020.

[10] Naeem Firdous Syed, Syed W. Shah, Arash Shaghaghi, Adnan Anwar, Zubair Baig, and Robin Doss. Zero Trust Architecture (ZTA): A Comprehensive Survey. *IEEE Access*, 10:57143–57179, 2022.

[11] Andreas Walz, Karl-Heinz Niemann, Julian Göppert, Kai Fischer, Simon Merklin, Dominik Ziegler, and Axel Sikora. PROFINET security: A look on selected concepts for secure communication in the automation domain. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–6, 2023.

[12] Rory Ward and Betsy Beyer. Beyondcorp: A new approach to enterprise security. 2014.

Microservices - when and how to use them

Prateek Agrawal
prateek.agrawal@aalto.fi

Antti Ylä-Jääski

Abstract

This paper critically examines software development methodologies, specifically comparing the advantages and disadvantages of employing monolithic versus microservices architectures. It also examines the ascendancy of microservices as a preferred approach for creating scalable and resilient services within an Agile development environment.

KEYWORDS: Monolithic, Microservices, agile, software methodology

1 Introduction

In recent years, the software development methodology known as microservices has gained much attention and widespread acceptance among software companies. This architectural approach is highly favored and in high demand for its ability to create scalable, robust, and powerful software systems. Microservices, characterized by their small and specialized nature, are designed to execute specific operations within a distributed system or software environment. Prior to the adoption of microservices architecture, applications were typically developed using a monolithic approach. Monolithic architecture, centered around a single executable artifact or library, was commonly employed in traditional application develop-

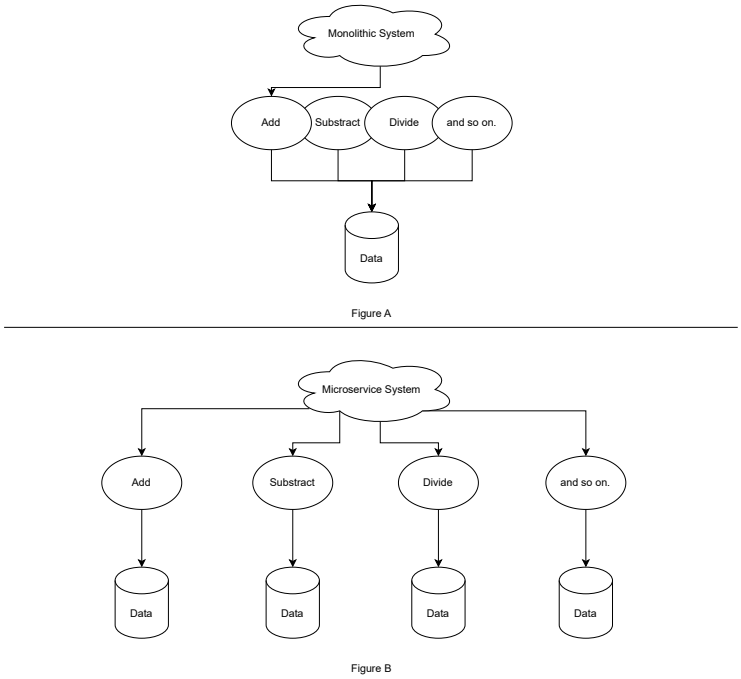


Figure 1. Monolith VS Micro service system

ment practices. Figure 1 depicts one such illustration. However, in recent years, microservices have emerged as a popular alternative to monolithic architectures.

In essence, the decision-making process regarding software architecture involves weighing the advantages and trade-offs of various principles and approaches, each offering unique benefits. It is crucial to understand the factors that have got microservices into the spotlight, the key considerations for choosing microservices over monolithic architectures, the circumstances under which a software system may need to transition from monoliths to microservices, the associated costs of such a transition, and the organizational value and benefits derived from adopting microservices. This paper aims to examine thoroughly into these critical aspects of working with microservices, providing insights into when and how to effectively implement them.

2 Microservices

Microservices represent a software architectural approach that is extensively leveraged in the development of forward-looking applications. This methodology involves the creation of applications composed of numerous smaller, individual components or services that can be deployed indepen-

dently. Each microservice is designed to perform a specific function within the application, enabling greater flexibility, scalability, and maintainability in software development projects. This decentralized approach to application design allows for easier management, updates, and scaling of individual services, ultimately contributing to the creation of more agile and resilient software systems.

Mazzara et al. [4] state that microservices are technically independent services conceptually deployed in isolation and loaded with some persistence tools (e.g. databases).

This practice is similar to a traditionally used Service-Oriented Architecture (SOA) [7], which uses ample of Web Services, Simple Object Access Protocol (SOAP), the Web Service (WS) calls. However, a micro service typically uses Representation state transfer (REST) and Hyper text transfer protocol (HTTP). [6]

Presently, the term has garnered much attention within the market due to its high demand, multiple benefits (Section 2.2) it affords, and its overarching design principles that enhance the agility of applications. Nonetheless, there exist several challenges and drawbacks associated with its implementation.(Section 2.3)

2.1 Principle

Microservices follow a simple, linear and service oriented strategy. It is usually a light weight component, which is open to extend or is scalable, is easy to test, can be easily managed and is independent piece of code. Often, this principles of a micro service is coupled with a cloud or distributed infrastructure or technology.

One of the best principle recommended while developing a microservice is "SOLID" principle, i.e. Single Responsibility Principle (SRP), Open-Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP), Dependency Inversion Principle (DIP).

1. SRP : The fundamental concept of the SRP posits that a class should be dedicated to a singular function or responsibility, namely: “a class should have only one reason to change” [2].

2. OCP : The core tenet of the Open-Closed Design Principle asserts that an architectural solution should be designed in a manner that allows for extension without necessitating modification of the existing codebase.

3. LSP : “if for each object o1 of type S there is an object o_2 of type T such that for all programs P defined in terms of T, the behaviour of P is unchanged when o_1 is substituted for o_2 then S is a subtype of T.” [8]

4. ISP : “the interfaces of the class can be broken up into groups of methods. Each group serves a different set of clients. Thus, some clients use one group of methods, and other clients use the other groups.“ [8]

5. DIP :

   (a) "High-level modules should not depend on low-level modules. Both should depend on abstractions.

   (b) Abstractions should not depend upon details. Details should depend upon abstractions." [8]

2.2 Advantages of using Microservices

1. Always available:

   When updating microservices, a well-designed architecture with microservices deployed in containers and utilizing DevOps practices, such as Continuous Integration and Continuous Delivery/Deployment allows for seamless updates without the need to decommission the entire system or restart services. This enables the hot deployment of new versions, ensuring that new client requests automatically access the updated microservices while the existing ones are gracefully shut down once their requests are completed or reach zero. [3]

   Similarly, in the event of failures, only the individual microservice will be impacted, preserving the functionality of the remaining modules within the system without disruption [4].

2. Alterations without causing disruptions to other components:

   Due to the independent nature of microservices as small, self-contained components, they are easily maintainable. In the event of a bug, navigating through the source code, identifying the issue, and rectifying it is simplified, thus enhancing maintainability and stream-

lining the release of new versions [4].

3. Streamlined Functional Testing and Error Identification:

   Due to their independent nature within a well-structured microservices architecture, individual microservices do not have a cascading effect on one another. Consequently, the failure of one component does not result in a system-wide crash. [10]

4. Ease of management:

   Utilizing smaller work teams offers benefits, such as accelerated and more frequent development and delivery cycles. These teams specialize in comprehensively understanding specific business functionalities implemented in microservices, enabling them to deepen their expertise with each development iteration. Furthermore, the experience gained allows these smaller teams to provide more precise development estimates and execute faster development processes

2.3 Challenges

When you're creating an application using microservices, a challenge is that it can be tough to learn this new way of structuring things. Companies often need to reorganize their teams so that each group can work on their part independently [12]. This might involve using new programming languages and tools to help with tasks like building, launching, and keeping track of the services. So, training is usually necessary to make sure everyone understands how to do these tasks well.

1. Inadequate Team Experience

   Another drawback is that an organization may not be certain of the success of adopting a new architecture, like microservices, because the development team has limited experience working with this type of architecture. Research shows that it took eight months for the project migration to master working with microservices [1].

2. Network latency

   Microservices often involve making many function calls, which can slow

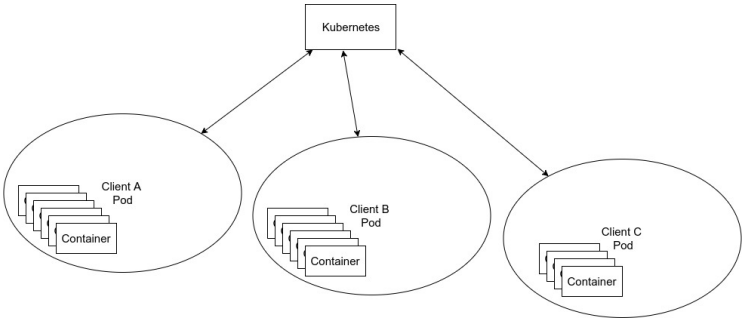


Figure 2. A typical microservice kubernetes instance

down communication between them due to network overhead. This may require analyzing and possibly changing the network architecture. To reduce these delays, it's important to optimize how programs make calls and process tasks, like using asynchronous calls and parallel processing. As microservices grow, upgrading the network infrastructure becomes necessary, which comes with additional costs. Figure 2 depicts a typical system comprising of a kubernetes engine to handle multiple pods.

3. Redundant Data

   In a microservices setup, each microservice should ideally have its own database. This means data needs to be duplicated across these databases, requiring extra programming steps to ensure data consistency. As more microservices are added, orchestrating them becomes more complex [5].

4. Streamlining testing and deployment with DevOps Automation

   Testing the entire application when using microservices can be complex due to each part having different feature. It takes longer to find and fix bugs with this new architecture. Thus, it is often required to automate with DevOps tools for tasks like integrating code, compiling, and deploying. Figure 3 demonstrates the steps involved in a typical DevOps system.

5. Difficult to debug

   When you have lots of microservices and use them in a Front End to find an error, it can be hard and take a long time to locate the bug. This process can end up costing you more [11] [9]. Use of proper logs and logging methodology is recommended to overcome such

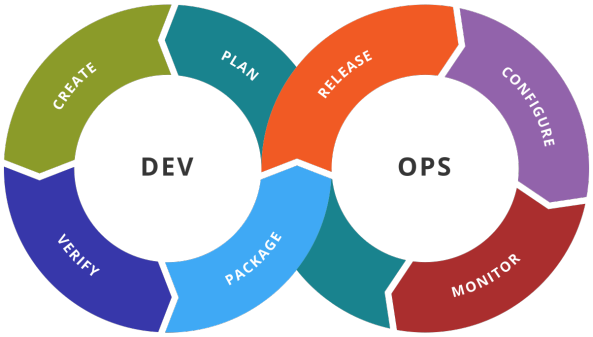


Figure 3. DevOps chain

challenges.

3 Decision Parameter

When considering whether to design a system as a monolith or microservice architecture, several decision parameters come into play. One important factor is the size and complexity of the project. Monoliths are generally easier to develop and maintain for smaller applications, as they have a single code base and database to manage. However, as the project grows in size and complexity, a monolith can become difficult to work with and scale. In contrast, microservices offer the ability to break down a large system into smaller, more manageable pieces that can be developed and deployed independently. This can lead to better scalability and fault tolerance, but also adds complexity to the system as a whole.

Another important decision parameter is the team size and expertise. Monoliths are often favored by small teams with limited resources and expertise, as they are simpler to develop and deploy. On the other hand, microservices require a higher level of expertise and coordination among team members, as each service needs to be independently developed, deployed, and managed. Additionally, microservices introduce additional challenges such as inter-service communication and data consistency across services. Ultimately, the decision between a monolith and microservice architecture should be based on the specific needs and constraints of the project, taking into account factors such as size, complexity, team expertise, and scalability requirements.

4 Conclusion

In conclusion, microservices offer numerous benefits such as scalability, flexibility, and increased efficiency for organizations looking to develop or update their software systems. It is important to carefully consider the specific needs and requirements of the project before deciding when and how to use a microservice architecture. Factors such as team expertise, project complexity, and performance requirements should all be taken into account when making this decision. Overall, microservices can be a valuable tool in modern software development, providing a more agile, flexible, upgraded and efficient approach to building and maintaining complex systems.

References

[1] Georg Buchgeher, Mario Winterer, Rainer Weinreich, Johannes Luger, Roland Wingelhofer, and Mario Aistleitner. Microservices in a small development organization. In Antónia Lopes and Rogério de Lemos, editors, *Software Architecture*, pages 208–215, Cham, 2017. Springer International Publishing. 10.1007/978-3-319-65831-5$_1$5.

[2] Elena Chebanyuk and Krassimir Markov. An approach to class diagrams verification according to solid design principles. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 435–441, 2016.

[3] Lianping Chen. Microservices: Architecting for continuous delivery and devops. In *2018 IEEE International Conference on Software Architecture (ICSA)*, pages 39–397, 2018.

[4] 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.

[5] Weibei Fan, Zhije Han, Yujie Zhang, and Ruchuan Wang. Method of maintaining data consistency in microservice architecture. In *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 47–50, 2018.

[6] Pooyan Jamshidi, Claus Pahl, Nabor C. Mendonca, James Lewis, and Stefan Tilkov. Microservices: The journey so far and challenges ahead. *IEEE Software*, 35(3):24–35, May 2018.

[7] Nicolai M. JOSUTTIS. *SOA in practice*. O'Reilly, 2007.

[8] R.C. Martin and M. Martin. *Agile Principles, Patterns, and Practices in C#*. Robert C. Martin series. Prentice Hall, 2007.

[9] Teguh Prasandy, Titan, Dina Murad, and Taufik Darwis. Migrating application from monolith to microservices. pages 726–731, 08 2020.

[10] Teguh Prasandy, Titan, Dina Fitria Murad, and Taufik Darwis. Migrating application from monolith to microservices. In *2020 International Conference on Information Management and Technology (ICIMTech)*, pages 726–731, 2020.

[11] 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.

[12] Victor Velepucha and Pamela Flores. A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE Access*, 11:88339–88358, 2023.

User-Provided-Infrastructure at the Edge

Radu Pogonariu
radu.pogonariu@aalto.fi

Tutor: Sara Ranjbaran

Abstract

In the age of the Internet of Things, devices are spread everywhere, each of them generating data. Traditionally all the computation is done in the cloud, however, as this can have a higher than desired latency to the device, the concept of fog computing appeared. Fog computing is a cloud-like computing paradigm, where the computation is done on the edge of the network. Here, on the edge, the compute can be provided by user-provided-equipment. However, this comes with multiple implications, such as task scheduling, incentive schemes, security or networking. In response to these issues, this paper reviews multiple papers addressing each issue, and discusses forming a set of these papers to be used in a complete fog computing system.

KEYWORDS: Fog computing, Internet of Things, Task scheduling, Networking, Security, Cloud computing

1 Introduction

As the Internet of Things (IoT) has become more and more popular, the traffic generated by IoT devices has been steadily increasing, as their number is projected to reach 25 billion by 2025 [1]. This number of IoT

devices leads to an increasing amount of data that is generated and has to be transferred to the cloud. As a result, the cost of traffic will only increase as time goes on, as will the volume of data. Additionally, the latency of the data transfer should be considered, as certain IoT applications depend on low-latency access to computing resources.

As a solution to these issues, the concept of fog computing was proposed by Bonomi et al. [2]. In fog computing, devices close to the user are used for computation, as opposed to servers in a data center. However, the features of the cloud are still desirable in fog computing, as they would allow for greater utilization of resources.

The processing power of fog computing is situated on the edge, close to the end-user. Consequently, alongside the multiple benefits it comes with, it also brings a number of issues, such as, the number of edge locations, the hardware provider, the connection of each location to the cloud and to each other, and uniformity of resource utilization. This paper will focus on user-provided equipment being used at the edge, and the required techniques for its success.

This paper is structured as follows. Section 2 presents cloud computing and fog computing. Section 3 presents the techniques that can be used in fog computing to solve the issues that arise from using user-provided-infrastructure at the edge. Section 4 discusses the feasibility of the methods presented in Section 3. Finally, Section 5 presents some concluding remarks.

2 Computing Paradigms

Computing paradigms have evolved through the times, first there was distributed computing, which evolved into cloud computing. As the Internet of Things was introduced, the need for lower latency became apparent, and as such, the concept of fog computing was introduced. However, in order to understand what is expected of fog computing, cloud computing should be looked into first.

2.1 Cloud Computing

Cloud computing is a computing paradigm that standardised commodity access to computing resources, first introduced by Google in 2003 through a number of research papers, and first commercialized by Amazon in 2006.

Cloud gained large scale adoption as it lowered the barrier to entry into the computing space, thus allowing more companies to get access to computing resources. In essence, the cloud enables the flexible usage of computing, while reducing the operating costs and capital expenditure [3]. Before cloud computing, any company that required computing resources would have to build their own infrastructure, which is the main advantage of cloud, as it moves the infrastructure and personnel costs to the provider, where economies of scale can happen. Alongside this, after the initial roll-out of cloud, the large providers kept adding new products, which simplified the expansion of cloud computing. However this resulted into "cloud lock-in", where customers are tied to a certain vendor, as they use proprietary products [4].

The public cloud is not an option in all industries, as they might have privacy requirements, or security considerations. As such, the concept of private cloud has been developed, where features similar to the public cloud are maintained on premises with the use of open-source technologies. Some of the used technologies are OpenStack, Kubernetes, Linux. [5]

2.2 Fog Computing

Fog computing is an extension to cloud computing, where the infrastructure is moved to the edge. Some of the goals of such an approach are to reduce latency between endpoints and computation, reducing operating costs of the compute infrastructure for a large IoT sensor network and taking advantage of geographical distribution [2]. A high level architecture of fog computing can be seen in Figure 1.

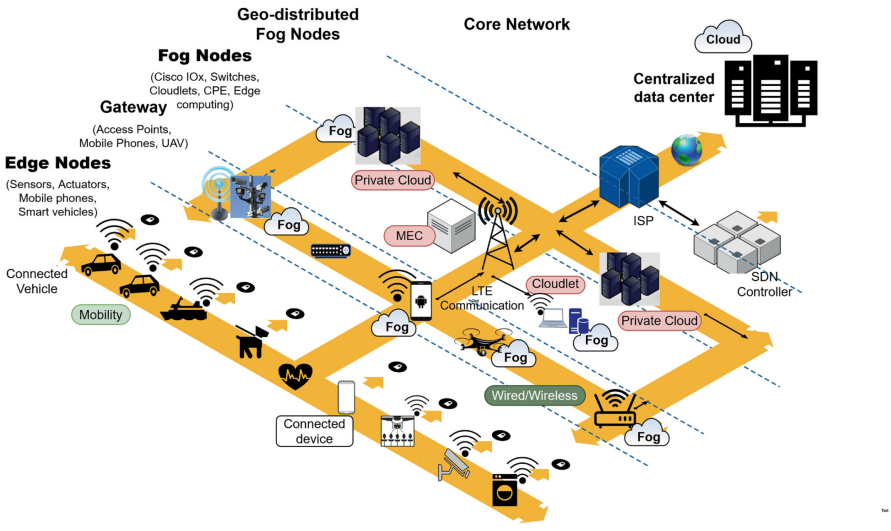


Figure 1. Hierarchical fog computing architecture from [6]

According to Sabireen and Neelanarayanan [7], the architecture some papers approach can be summarized as: "Level 1: Physical and virtual sensors", "Level 2: Fog device, server and gateway", "Level 3: Monitoring", "Level 4: Pre and post processing", "Level 5: Storage and resource management", "Level 6: Security" and "Level 7: Application". Level 1 is comprised of the physical or virtual sensors that generate data for the fog network, Level 2 is formed of the network compute, be it an IoT device or an independent server. Level 3 is represented by the monitoring component of such a network, to be able to facilitate resource allocation. Level 4 handles the processing of data that is produced by Level 1, and Level 5 is in charge of storing the data processed by Level 4. Level 6 concerns the security of the network and Level 7 is the actual application that can run on the fog network to augment IoT devices. Some of the use cases in which fog computing can be used are: smart healthcare, where it can collect and process different sensors used by patients, smart city, where it can collect and manage different sensor and actuators throughout a city and entertainment, and cloud gaming, which highly depends on the latency between the user and the computing resources [6].

On the network edge there are a lot of resources available that are unused which could be used to enhance the fog network, and this will be looked into in the next section [8].

3 Algorithm details

Using user-provided equipment in fog computing brings forward a number of challenges. Firstly, the heterogeneity and uniform utilization of the equipment is an issue, as the multiple hardware configurations that users might bring into the network have different levels of performance, thus rendering traditional scheduling algorithms obsolete. Secondly, the challenge of incentives arises, given that users lack inherent motivation to bring their hardware into the network, necessitating a tangible reward to incentivize their participation. Thirdly, the is the issue of security, as the hardware that is brought into the network can be under the control of a nefarious actor, which can launch an attack on the network if not properly secured. Lastly, the network traffic that is produced by the fog node would cause problems for the user if the traffic generated by it is not separated from the users traffic.

Each of these issues have been approached in previous research pa-

pers. In the following subsections, a number of papers will be reviewed for each of the issues that were raised. Thus, for task scheduling the reviewed papers are [9, 10, 11], for incentive schemes, the papers are [8, 12, 13], for security, the papers are [14, 15], and finally, for networking, the papers are [16, 17]. The papers were chosen by looking at papers mentioned by multiple survey papers, and by searching `<issue> AND fog AND computing` on Scopus, where `<issue>` was replaced by the issue mentioned above, and manually selecting appropriate papers.

3.1 Task scheduling

Task scheduling is an important aspect in fog computing, where all applications that run on the system depend on it. Different use cases of fog computing have different priorities for their scheduling, such as latency [9], reducing the SLA violation rate [10] or improving the packet delivery ratio [11].

Barzegaran et al. [9] proposed a Simulated Annealing-based metaheuristic, which caters to their use case of Industrial IoT control applications that are safety-critical and real-time, and that have very low latency and jitter requirements. The authors used Simulated Annealing to propose new "neighboring solutions", and evaluated with their Cost of Control function, which is a performance index that evaluates the behavior over time, thus the performance of a new solution is better if the value of this mathematical function increases.

Sun and Zhang [10] use game theory to model a fog computing system, where the fog broker uses an incentive mechanism to encourage more resource owners to contribute their resources to the system, while receiving a reward. The authors model this as a repeated game with complete information, and uses information such as energy costs, reward per unit of time, the degree of patience of the players and reward for completing a task. The proposed model performs better than typical dynamic scheduling algorithms, Min-Min and MBFD.

Hameed et al. [11] suggest a different approach than the previous two papers, clustering vehicles that are in proximity of each other and capacity-based load-balancing among clusters. Each cluster has a cluster head (CH) fog device that is responsible to distribute the tasks among the clusters and within the cluster of vehicles. In order to select the appropriate node as a CH in a cluster, the fog gateway uses beaconing and prediction mechanisms. After this process, the fog gateway offloads tasks

to CHs, which in turn also offload it to another CH or inside their cluster.

3.2 Incentive schemes

An incentive scheme is an important aspect of a fog computing system that has users contributing their resources. While Nazih et al. [8] have a theoretical incentive scheme for a vehicular fog network, the Helium Network [12] is a physical system, similar to a fog network, that has a working incentive scheme [13].

Nazih et al. [8] propose an incentive scheme based on the combination of two mathematical concepts: Stackelberg games and contract theory. The authors use the Stackelberg games to determine the reward of a task, and contract theory to determine the node that will fulfill the task.

The Helium Network [12] is a network that provides wireless connectivity using user-provided-equipment. In this case, the network provides a reward for every packet that is successfully delivered, and the rewards are determined by the coverage a node provides, such that a node in a sparse area is more productive than a node in a crowded area. According to Jagtap et al. [13], as of May 2021, Helium has over 40000 active nodes, and their ownership is decentralized, as 84% of users own at most three hotspots.

3.3 Security

Security is an important factor when insecure user equipment is brought into the fog network, as that node can compromise user data, which can user privacy implications. Some solutions are introduced by Li [14] in the form of a trustless compute layer, and Wang et al. [15] by authenticating data as it updates.

Li [14] proposes a blockchain based trustless layer, which protects both the source data and the algorithm processing the data. In this case, the function would accept "hidden" data as input, and would generate a zero-knowledge computing proof, which can be verified publicly. These are implemented using smart contracts on a DAG+ blockchain.

Wang et al. [15] presents an incremental authentication scheme for updated data in fog computing. This scheme is build using an efficient incremental signature scheme that is a lattice-based multi-blocks and mixed incremental signature scheme. This scheme supports all the knows incremental operation.

3.4 Networking

Networking has a significant role in fog computing, acting as the backbone of the entire system. When user-provided-equipment is used in a fog network, network slicing becomes an important factor as highlighted by Theodorou and Xezonaki [16]. Another aspect that should be considered is the dissemination of critical updates, as shown by Vikhrova et al. [17].

Theodorou and Xezonaki [16] propose a 2-tier gateway system, where each IoT slice gets a virtualised gateway, while there is a single gateway mediating access to the rest of the sensor network, as well as proposing a novel archirecture building on the NFV MANO framework. Their approach automates the process of creating an IoT slice to the order of a few minutes.

Vikhrova et al. [17] introduce a new paging algorithm that improves the Single-Cell Point-to-Multipoint component of the 5G standard, such that it can better handle the fast distribution of critical updates after a bug fix or system failure.

4 Discussion

The reviewed papers offer a glimpse into the possible algorithms that can be integrated into a fog computing system that uses user-provided-equipment. It is not necessary, or expected, to use all the papers. However, depending on the final use case of the system, a combination of them would be relevant. An example system could use the scheduling of Hameed et al. [11], the reward scheme of Helium [13], the trustless compute layer of Li [14] and the network slicing of Theodorou and Xezonaki [16].

Another thing to note is that there no large-scale deployments of a public or private fog network, while the research output in this area of computing has been increasing over the past years [6]. This is due to it not presenting a clear business case for the companies yet. As a result, the market capture of fog computing is projected to be only 343 million dollars by 2030, while the cloud market is projected to be around 791 billion dollars by 2028 [6].

In future research, it is recommended to develop a complete general purpose fog network, which can provide test-bed for future work. This would accelerate the large-scale deployment of such a system, moving it

from the simulators to the real world.

5 Conclusion

This paper studies fog computing and the required algorithm to use user-provided-equipment at the edge of the network. In total, 9 algorithm were presented, covering a wide range of uses, with the goal that a combination of them can be used in a fog computing system. Additionally, a set of algorithms for part of a fog computing system was discussed.

References

[1] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Computer Networks*, vol. 192, p. 108040, Jun. 2021. doi: 10.1016/j.comnet.2021.108040

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ser. MCC '12. New York, NY, USA: Association for Computing Machinery, Aug. 2012. doi: 10.1145/2342509.2342513. ISBN 978-1-4503-1519-7 pp. 13–16.

[3] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud Computing: An Overview," in *Cloud Computing*, ser. Lecture Notes in Computer Science, M. G. Jaatun, G. Zhao, and C. Rong, Eds. Berlin, Heidelberg: Springer, 2009. doi: 10.1007/978-3-642-10665-1_63. ISBN 978-3-642-10665-1 pp. 626–631.

[4] G. C. Silva, L. M. Rose, and R. Calinescu, "A Systematic Review of Cloud Lock-In Solutions," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, Dec. 2013. doi: 10.1109/CloudCom.2013.130 pp. 363–368.

[5] S. Goyal, "Public vs Private vs Hybrid vs Community - Cloud Computing: A Critical Review," *International Journal of Computer Network and Information Security*, vol. 6, no. 3, p. 20, 2014. doi: 10.5815/ijcnis.2014.03.03

[6] S. N. Srirama, "A decade of research in fog computing: Relevance, challenges, and future directions," *Software: Practice and Experience*, vol. 54, no. 1, pp. 3–23, 2024. doi: 10.1002/spe.3243

[7] H. Sabireen and V. Neelanarayanan, "A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges," *ICT Express*, vol. 7, no. 2, pp. 162–176, Jun. 2021. doi: 10.1016/j.icte.2021.05.004

[8] O. Nazih, N. Benamar, and A. Addaim, "An Incentive Mechanism for Computing Resource Allocation in Vehicular Fog Computing Environment," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, Dec. 2020. doi: 10.1109/3ICT51146.2020.9312007 pp. 1–5.

[9] M. Barzegaran, A. Cervin, and P. Pop, “Towards quality-of-control-aware scheduling of industrial applications on fog computing platforms,” in *Proceedings of the Workshop on Fog Computing and the IoT*, ser. IoT-Fog ’19. New York, NY, USA: Association for Computing Machinery, Apr. 2019. doi: 10.1145/3313150.3313217. ISBN 978-1-4503-6698-4 pp. 1–5.

[10] Y. Sun and N. Zhang, “A resource-sharing model based on a repeated game in fog computing,” *Saudi Journal of Biological Sciences*, vol. 24, no. 3, pp. 687–694, Mar. 2017. doi: 10.1016/j.sjbs.2017.01.043

[11] A. R. Hameed, K. Munir, S. u. Islam, and I. Ahmad, “Load-balancing of computing resources in vehicular fog computing,” in *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*, Jun. 2020. doi: 10.1109/ICDIS50059.2020.00020 pp. 101–108.

[12] A. Haleem, A. Allen, A. Thompson, M. Nijdam, and R. Garg, “Helium: A decentralized wireless network,” Nov. 2018, accessed: 2024-02-25. [Online]. Available: http://whitepaper.helium.com/

[13] D. Jagtap, A. Yen, H. Wu, A. Schulman, and P. Pannuto, “Federated infrastructure: usage, patterns, and insights from "the people’s network"," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC ’21. New York, NY, USA: Association for Computing Machinery, Nov. 2021. doi: 10.1145/3487552.3487846. ISBN 978-1-4503-9129-0 pp. 22–36.

[14] W. Li, “Trustless Layer for Secure Fog Computing,” in *Proceedings of the 8th International Conference on Cyber Security and Information Engineering*, ser. ICCSIE ’23. New York, NY, USA: Association for Computing Machinery, Dec. 2023. doi: 10.1145/3617184.3630157. ISBN 9798400708800 pp. 328–334.

[15] F. Wang, J. Wang, and W. Yang, “Efficient incremental authentication for the updated data in fog computing,” *Future Generation Computer Systems*, vol. 114, pp. 130–137, 2021. doi: 10.1016/j.future.2020.07.039

[16] V. Theodorou and M.-E. Xezonaki, “Network Slicing for Multi-tenant Edge Processing over Shared IoT Infrastructure,” in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, Jun. 2020. doi: 10.1109/NetSoft48620.2020.9165327 pp. 8–14.

[17] O. Vikhrova, S. Pizzi, A. Molinaro, A. Iera, K. Samouylov, and G. Araniti, “Group-based delivery of critical traffic in cellular IoT networks,” *Computer Networks*, vol. 181, p. 107563, Nov. 2020. doi: 10.1016/j.comnet.2020.107563

How Reliable is TOR

Ranjit Nepal
ranjit.nepal@aalto.fi

Tutor: Tuomas Aura

Abstract

This paper offers an overview of the reliability of the Tor(The Onion Router), a technology designed to enhance online privacy and anonymity. The paper outlines the operational principles of the Tor network. While acknowledging the importance of Tor in safeguarding user privacy, especially for activists and social groups, the paper also highlights the network's vulnerabilities to various forms of attacks. These include threats to user anonymity, Denial-of-Service (DoS) attacks, and attacks from the network. By examining these challenges, the paper aims to provide insights into the advantages and limitations of Tor, enhancing understanding of its role in maintaining online privacy and security in the digital age.

KEYWORDS: TOR, Anonymity, Network Attacks

1 Introduction

The Tor overlay network is a global network designed to provide anonymous communication over the internet. It is a key technology developed to enhance online privacy and anonymity in the digital communication era. Tor functions as a distributed anonymity network managed by volunteers, offering users an certain degree of protection against monitoring, censorship, and discrimination [7]. Users can hide their online activity by using the Tor overlay network, which allows routing their traffic through a number of encrypted nodes [7, 15].

The reliability of the Tor is crucial for safeguarding the security of its wide range of users. Any breaches in Tor could impact not only individual privacy but also the safety of activists and social groups relying on it [10]. Hence, maintaining Tor's reliability is essential, requiring developers and other stakeholders to remain committed to strengthening its infrastructure and protecting privacy and anonymity for its users.

This paper reviews the attacks on the Tor network and approaches aimed at enhancing its reliability. This includes examining attacks on its anonymity, and Denial-of-service (DoS) attacks [16, 4, 5, **?**, 12], offering insights into Tor's advantages and disadvantages.

This paper is organized as follows. Section 2 explains the Tor's operating principle. Section 3 describes different kinds of attacks against Tor from different adversaries. Section 4 discusses briefly good practices and usability of Tor. Finally, Section 5 presents concluding remarks.

2 Design

Tor, a second-generation Onion Routing system, is designed to anonymize web browsing, secure shell, and instant messaging [7]. It is a distributed overlay network, where clients build a circuit through the network and each node in this circuit only knows its immediate predecessor and successor. Tor also uses Perfect Forward Secrecy (PFS), decentralized congestion control, varied exit policies, and hidden services to enhance security.

The data flows in the Tor network in small packets which are called cells. Cells in the Tor network are the fundamental units of communication, each being 512 bytes in size, making traffic analysis difficult. Composed of a header and a payload, the cells are categorized into control cells for circuit management like setup and maintenance and relay cells

carrying end-to-end payload data. The relay cells include a relay header with a stream identifier (streamID), integrity checksum, payload length, and relay command, which are essential for data routing and management within Tor [7]. These cells employ AES cipher encryption in counter mode, providing data confidentiality as the cells traverse the circuit.

Path selection in Tor is client-driven, where clients use their Onion Proxy (OP) to choose a series of onion routers (ORs) to create a path for data routing, guided by information from Tor's directory servers. Directory servers act as a repository of information about the Tor network. They maintain up-to-date lists of active ORs, including details about each router's status, capabilities, and policies. This information is essential for Tor clients to construct efficient and secure circuits through the network. The OP initiates a connection with a randomly selected entry node (guard node) and incrementally builds the circuit by adding nodes one at a time, establishing shared symmetric keys via Diffie-Hellman handshakes. As the OP sends data through the circuit, each OR decrypts only its layer of encryption, unaware of the data's original source or ultimate destination. This ensures that neither the entry node knows the data content or final destination, nor does the exit node know the original source, maintaining the anonymity core to Tor. The basic path followed in Tor network is shown in Figure 1.

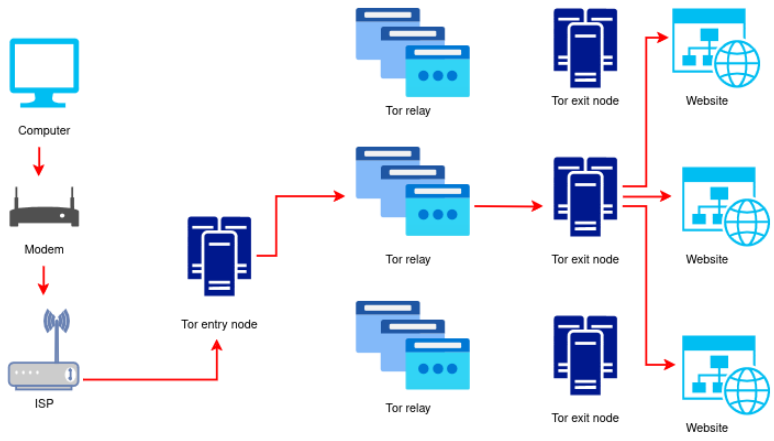


Figure 1. Communication in Tor network.

Despite numerous security measures in place, the Tor network still faces various types of attacks. These include attacks on user anonymity as well as DoS attacks, aiming to flood servers with traffic.

3 Attacks

This section explores the threats posed by adversaries to the Tor network, emphasizing their capacity to monitor internet traffic.

3.1 Compromised nodes

These type of attacks involve an attacker gaining control over one or more nodes (usually entry or exit nodes) in the Tor network. If an attacker controls some of these nodes, they can potentially observe, modify, or block the data passing through them [11].

If an attacker gains control of an entry node in the Tor network and can see the traffic between exit node and server, they can potentially conduct an active website fingerprinting attack. This involves manipulating traffic that passes through the controlled node [9, 13]. The attacker's goal is to infer which websites a Tor user is accessing despite the encryption and routing methods Tor uses to anonymize traffic.

In this specific kind of attack, the attacker delays certain HTTP requests at the entry node. They create unique, distinguishable patterns in the traffic flow, which can then be analyzed to identify the websites being visited by the Tor user [17]. This method requires a deep understanding of Tor's traffic and protocol behavior, including the ability to identify the first HTTP request and choose subsequent points in the traffic flow to delay.

Another attack can be used to confirm that a specific communication between two parties is happening over Tor. In this attack, the attacker controls both the entry and exit onion routers in a Tor circuit. This method involves manipulating cells in various ways – duplicating, modifying, inserting, or deleting them [12]. This manipulation disrupts the cell's normal flow and encryption, leading to recognizable decryption errors at the exit onion router.

The key to de-anonymization lies in correlating the manipulated cells at the entry node with the decryption errors observed at the exit node. Since the attacker controls both nodes,the times and details of cell manipulations at the entry node are logged and compared with the error logs at the exit node. This information reveals that the specific traffic entered the Tor network from a particular IP address (associated with the entry node) and exited towards a specific destination (associated with the exit node). While this does not immediately expose the identities two parties,

it narrows down the possibilities.

3.2 Attacks by network

These attacks are carried out by Internet Service Providers (ISPs) or other entities that have control over network infrastructure. They can also be called Autonomous system (AS), which is a large network or group of networks that has a single, unified routing policy. These attacks are important because they can undermine the effectiveness of Tor at a network level, beyond just targeting individual nodes [15].

Asymmetric Traffic Analysis is one of the attacks where an adversary exploits the asymmetry of routing to compromise anonymity. The data packets and their responses often take different routes in the internet. In a Tor network, where user anonymity is important, this asymmetry can be a vulnerability. An adversary controlling an AS can observe traffic moving in one direction at both ends of the communication path. They observe data packets going from a user to a Tor entry node and acknowledgments coming from a destination server to a Tor exit node. By analyzing patterns in this traffic, such as timing and packet sizes, the adversary can potentially correlate these data flows. This correlation might reveal the identity of Tor users or their activities, despite not monitoring all traffic directions, thus posing a threat to the anonymity.

Exploiting BGP (Border Gateway Protocol) churn is another aspect of the routing related attacks. BGP churn refers to the natural changes in internet routing paths over time, influenced by factors like policy changes, link failures, or new connections [3]. Such changes can reroute internet traffic through different Autonomous Systems (ASs) including those controlled by adversaries. In the context of the Tor network, this churn means that the paths Tor traffic takes can vary over time, increasing the likelihood of passing through malicious ASs. An adversary can exploit this nature of BGP routing to increase the chance of observing Tor traffic. The longer a user uses the Tor network, the higher the probability that part of their traffic will pass through an adversary-controlled AS, allowing the adversary to conduct traffic analysis [15]. This phenomenon makes the Tor network more vulnerable over time as the unpredictability of internet routing increases the risk of exposure and potential de-anonymization of Tor users.

BGP Hijacks and Interceptions represent the most direct and invasive routing attack. In a BGP hijack, an AS maliciously announces that it is

the correct route for IP addresses that it does not own [18]. This misdirection can cause a portion of the internet's traffic, which includes Tor traffic, to be rerouted through the malicious AS. Even more sophisticated is a BGP interception, where the attacker temporarily routes the traffic through their AS and then back to its original path, making the interception almost invisible. By hijacking or intercepting the IP addresses of Tor nodes, an adversary can directly observe and potentially alter the traffic flowing through these nodes. This capability not only enables them to perform detailed traffic analysis, potentially unmasking users and their activities, but also poses the risks of injecting malicious traffic or disrupting Tor communications. Such direct control over Tor traffic is a severe threat to user anonymity and can undermine the core functionality and trust in the Tor network [15]. Distributing Tor nodes across multiple ASs and geographic locations can reduce the risk of a significant portion of Tor traffic being routed through a malicious AS. Decentralization makes it harder for a single AS to control or monitor a substantial amount of Tor traffic.

A country or network can also block access to the Tor network, and the Great Firewall of China (GFC) is a prominent example of such censorship [1]. The GFC employs sophisticated deep packet inspection (DPI) to detect Tor traffic. When a Tor connection is identified, the GFC initiates active scanning using various Chinese IP addresses to confirm the presence of Tor bridges. Once confirmed, these bridges are dynamically blocked rather than simply blocklisting their IP addresses. This approach illustrates a high level of adaptability in censorship efforts with the ability to rapidly detect and block new access points to the Tor network. This kind of dynamic and sophisticated blocking mechanism poses a significant challenge for Tor users within such a censored network, requiring advanced circumvention methods to bypass the restrictions.

3.3 Attacks by global adversary

This type of attack assumes the adversary has the ability to observe a large portion of the internet traffic worldwide. The threat posed by a global adversary is particularly daunting because it challenges the fundamental principle of Tor, which is to protect against traffic analysis. Sybil attack specifically refers to a scenario where an attacker operates multiple Tor nodes under different identities. This kind of attack exploits the decentralized and trust-based nature of the Tor network [16].

One of the attacks is exit traffic tampering where sybil nodes function as exit nodes; the last point of Tor traffic before it enters the regular internet. If an attacker controls these exit nodes, they can intercept the traffic passing through them. Such control allows the attacker to eavesdrop on unencrypted or poorly secured traffic, capture sensitive data like credentials, and potentially alter or inject malicious content into the traffic stream [14]. This compromises the confidentiality and integrity of the data being transmitted through these nodes and can lead to privacy breaches.

In regions with strict internet censorship, Tor users often rely on bridge nodes (unlisted entry nodes) to access the network. Sybil nodes positioned within the network can observe and log connections from the bridge nodes [16]. This attack can lead to the blocking of the bridges by censorship authorities, thereby denying access to the Tor network for users in those regions.

3.4 DOS Attacks

Denial of Service (Dos) attacks in the Tor network involve overwhelming specific nodes or services with an excessive amount of traffic, rendering them unresponsive or severely degraded in performance. In the context of Tor, these attacks can be particularly disruptive due to the network's reliance on a limited number of volunteer-operated nodes.

One of the attacks involves flooding Tor's default bridges with packets, which overwhelms their bandwidth capabilities. This is achieved by using "stresser" services to direct a high volume of traffic at these bridges. As a result, the bridges become overloaded and cannot provide effective service to users trying to connect to the Tor network. The attack focuses on the default bridges since they are hardcoded in the Tor Browser Bundle and are essential for users in censored regions to access Tor [**?**].

Another attack targets the TorFlow system, which measures the performance of Tor nodes and is critical for Tor's load-balancing. The attackers flood the TorFlow scanners with packets, disrupting their ability to accurately measure node performance. By affecting the measurement process, the reliability and consistency of node capacity estimation are compromised [**?**]. This misrepresentation of node performance leads to inefficient load distribution across the network, drastically reducing the median client download rate. Client download rate is a measure of the network's performance from the perspective of an end user, typically rep-

resented in terms of bytes per second.

Another attack is where the attacker uses the Tor protocol itself to congest the network. They build thousands of 8-hop circuits (technical limit of no of nodes in a path) and use these to download large files through the network [8]. In an 8-hop circuit, data is transmitted through eight different nodes, effectively amplifying the bandwidth consumed for each piece of data. This approach results in a significant increase in data traffic through the nodes, leading to network congestion. Each byte of data is processed multiple times across the extended circuit path, increasing the overall load on the Tor network and thereby slowing down the network performance for regular users.

4 Discussion

Tor is a useful tool for anyone looking to keep their online activities private, like journalists or activists in places where there is a lot of internet control. It can be used to access websites that might be blocked in some countries. However, it does not work well for things that need a lot of internet speed or for activities that need to happen in real-time, like watching high-quality videos or playing online games. It is also important to remember that Tor should not be used for anything illegal.

When using Tor, there are a few things to keep in mind to use it safely. Firstly, personal information like bank details should not be shared. It is best to stick with secure websites (those with HTTPS in the address) when using Tor as it prevents exit router from spying the communication. Furthermore, when using Tor browser, add-ons or plugins could give away the client identity or location, so their use should be avoided. Tor browser should be updated regularly for the best security. Sticking to Tor's standard settings is usually the safest measure. Using VPN (Virtual Private Network) can be particularly beneficial in scenarios where the user does not want their internet service provider (ISP) or anyone monitoring their local network to know they are accessing Tor.

There are also tools like I2P (Invisible Internet Project) [2] and Freenet [6] which provide censorship resistant communication and privacy. However they lack infrastructure and user base like Tor, resulting in slower browsing speeds and fewer available services.

5 Conclusion

In summary, this paper has reviewed the reliability of the Tor network, highlighting its role in ensuring online anonymity and privacy. Despite its robust design, Tor faces various challenges including compromised nodes, ISP-level attacks, global adversaries, and DoS attacks, which threaten user privacy and the network's stability. While Tor remains an essential tool for many users, particularly in oppressive regimes or for those needing anonymous communication, its vulnerabilities necessitate ongoing vigilance and enhancements.

References

[1] How the Great Firewall of China is Blocking Tor. In *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*. USENIX Association, August 2012. https://www.usenix.org/conference/foci12/workshop-program/presentation/Winter.

[2] Jacques Bou Abdo and Liaquat Hossain. Modeling the Invisible Internet. Springer, February 2024. "https://doi.org/10.1007/978-3-031-53472-0_30".

[3] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A Study of Prefix Hijacking and Interception in the Internet. *Comput. Commun. Rev.*, 37(4), October 2007. https://doi.org/10.1145/1282427.1282411.

[4] Abdelberi Chaabane, Pere Manils, and Mohamed Ali Kaafar. Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network. In *2010 Fourth International Conference on Network and System Security*, pages 167–174, 2010. https://doi.org/10.1109/NSS.2010.47.

[5] Giovanni Cherubin, Rob Jansen, and Carmela Troncoso. Online Website Fingerprinting: Evaluating Website Fingerprinting Attacks on Tor in the Real World. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 753–770, August 2022. https://www.usenix.org/conference/usenixsecurity22/presentation/cherubin.

[6] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. *Freenet: A Distributed Anonymous Information Storage and Retrieval System*. Springer Berlin Heidelberg, 2001. https://doi.org/10.1007/3-540-44702-4_4.

[7] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX Security 04)*. USENIX Association, August 2004. https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router.

[8] Nathan S Evans, Roger Dingledine, and Christian Grothoff. A Practical Congestion Attack on Tor Using Long Paths.

In *USENIX Security Symposium*, pages 33–50, 2009. https://www.usenix.org/conference/usenixsecurity09/technical-sessions/presentation/practical-congestion-attack-tor-using.

[9] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website Fingerprinting. In *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, November 2009. https://doi.org/10.1145/1655008.1655013.

[10] Abid Khan Jadoon, Waseem Iqbal, Muhammad Faisal Amjad, Hammad Afzal, and Yawar Abbas Bangash. Forensic Analysis of Tor Browser: A Case Study for Privacy and Anonymity on the Web. volume 299. Forensic Science International, 2019. https://doi.org/10.1016/j.forsciint.2019.03.030.

[11] Ishan Karunanayake, Nadeem Ahmed, Robert Malaney, Rafiqul Islam, and Sanjay K. Jha. De-Anonymisation Attacks on Tor: A Survey. *IEEE Communications*, 23(4), 2021. https://doi.org/10.1109/COMST.2021.3093615.

[12] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Weijia Jia, and Wei Zhao. Protocol-level Attacks Against Tor. volume 57. Computer Networks, 2013. https://doi.org/10.1016/j.comnet.2012.11.005.

[13] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, October 2011. https://doi.org/10.1145/2046556.2046570.

[14] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. Characterizing the Nature and Dynamics of Tor Exit Blocking. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 325–341. USENIX Association, August 2017. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/singh.

[15] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: Routing Attacks on Privacy in Tor. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, August 2015. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sun.

[16] Philipp Winter, Roya Ensafi, Karsten Loesing, and Nick Feamster. Identifying and Characterizing Sybils in the Tor Network. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1169–1185. USENIX Association, August 2016. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/winter.

[17] Ming Yang, Xiaodan Gu, Zhen Ling, Changxin Yin, and Junzhou Luo. An Active De-anonymizing Attack Against Tor Web Traffic. *Tsinghua Science and Technology*, 22, 2017. https://doi.org/10.23919/TST.2017.8195352.

[18] Zheng Zhang, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao. Practical defenses against BGP prefix hijacking. CoNEXT '07. Association for Computing Machinery, 2007. https://doi.org/10.1145/1364654.1364658.

Taxonomy of Supply Chain Attacks Against Machine Learning Systems.

Salahuddin Salahuddin
salahuddin.salahuddin@aalto.fi

Tutor: Marchal Samuel

Abstract

Machine learning (ML) has developed exponentially and has found applications due to its various advantages. However, machine learning-based systems might contain different vulnerabilities that could be exploited to sabotage the system or gain an unfair advantage. These vulnerabilities might exist in the system or be caused by external factors involved in its development. The different external and internal components compose the machine learning supply chain. Adversaries employ various attack patterns to exploit these threats to jeopardize the machine learning supply chain. Therefore, it is essential to take security measures to counter adversarial attempts. This paper highlights the consequences of supply chain attacks against machine learning systems while outlining various mitigation strategies to counter these vulnerabilities. It provides a detailed taxonomy of adversarial attacks against the ML supply chain and explains the importance of security measures for hardware and software.

KEYWORDS: machine learning, deep learning, artificial intelligence, machine learning supply chain, adversarial machine learning (AML), attack taxonomy, defense strategies, attack mitigation

1 Introduction

Artificial intelligence (AI) and machine learning (ML) have recently experienced significant advancements, leading to global AI integration in different fields and disciplines. Machine Learning (ML) has emerged as an integral component in various sectors, including medical diagnosis, business decision-making, stock market analysis, human behavioral analysis, and applications such as biometric authentication, chatbots, and autonomous vehicles. Generative artificial intelligence (GenAI) has also become quite popular because of its ability to process varying data formats [30]. For instance, ChatGPT, a large language model (LLM) from OpenAI, can also handle image, audio, and video data. Extensive research is also being done in multimodal—multimodal is a branch of machine learning models that can take input and generate output in different formats, such as text-to-speech, image-to-text, and text-to-image.

Although these systems have many advantages, the fact remains that they present many challenges, which, if not mitigated, could lead to unwanted consequences. Malicious parties could compromise machine learning systems to sabotage them or gain an unfair advantage. For instance, malicious attacks could lead to breaches in data privacy or erroneous, potentially fatal decisions [33] [2], or if the computer vision model of an autonomous vehicle is fed poisoned data, it might erroneously guide the vehicle into the wrong lane or misclassify traffic signs [17]. Malicious actors could also embed backdoors in generative models, producing inappropriate content in response to specific prompts [8] [47]. Multimodals are considered to be more resistant to malicious attacks due to their complexity. However, specific attack patterns have evolved against multiple modalities, which has led to the rise of comprehensive attacks against all modalities [11].

While designing and implementing an ML system, generally, the environment is assumed to be secure, which might not be accurate. Several factors are involved in developing machine learning applications besides the standard ML According to a report by Sonatype in 2022, the average increase in software supply chain attacks since 2019 has been around 742% [42], and in 2023, twice as many supply chain attacks were reported [1]. Although much research has been done about adversarial machine learning, most of it has been focused on malicious attacks on individual components of ML systems, such as data poisoning, model backdoor injec-

tion, and model inversion. This paper highlights security vulnerabilities in the machine learning supply chain and the consequences of such security threats. Additionally, the paper discusses various defense strategies suggested or implemented in the literature to counter adversarial attacks and their limitations.

2 Background

The following sections provide a high-level overview of the machine learning system, supply chain, and security vulnerabilities that malicious actors could exploit.

2.1 Machine Learning

A typical ML pipeline comprises various components and operations: data preparation, model training, evaluation/testing, and deployment. Each step plays a critical role in developing a robust machine-learning system. Data preparation involves gathering and cleaning data as well as partitioning it into training, testing, and evaluation sets. Afterward, the ML model is trained on the training set, evaluated via the validation set, and tuned if necessary. Once the model performs satisfactorily, it is tested through the testing set before deployment for inference in the real world [21].

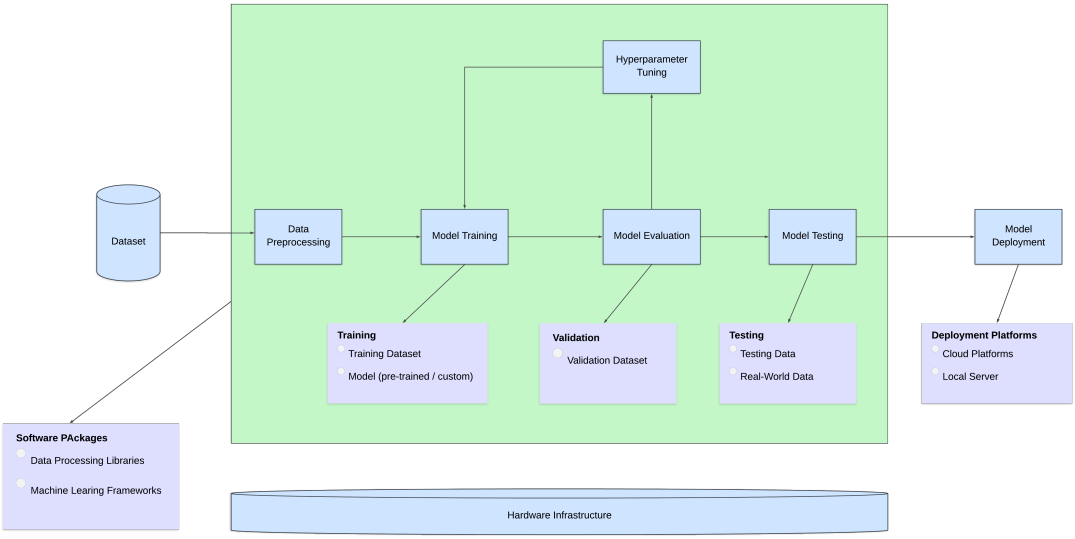


Figure 1. Machine Learning Pipeline

The security of an ML system is substantial, as vulnerabilities or threats in the system might lead to malicious attacks.

2.2 ML Supply Chain

Generally, a supply chain is described as a mesh of individuals, organizations, resources, activities, and technology employed for designing, building, and distributing a product. Since the supply chain involves various components and stages, it is susceptible to many adversarial attacks, either by compromising hardware or introducing malicious updates in firmware or software at any phase. These security threats could be mitigated by taking proper measures such as vendor risk enhancement, software integrity checks, trusted hardware utilization, secure development and manufacturing practices, and anomaly detection and monitoring.

The supply chain of a machine learning system encompasses the components and operations discussed in section 2.1, as well as ML frameworks, third-party libraries, and hardware used to implement a machine learning system [41]. For instance, to develop an Object Detection System, the supply chain could be defined as obtaining data from a customer or open-source platform, such as Kaggle; a pre-trained model could be downloaded from HuggingFace for fine-tuning; training can be performed on AWS Sagemaker or Google Cloud; and deployment through frameworks, such as TorchServe, Tensorflow Serving, and ONNX Runtime.

The ML supply chain is susceptible to adversarial attacks, similar to a general product development supply chain. However, the attack mechanism might be the same or different. For example, an adversary could insert a compromised component, such as a malicious chip, in product development. On the other hand, the ML supply chain could be sabotaged by injecting trojans into the training platform. As such, comprehending the ML supply chain and its vulnerabilities, just like any product development supply chain, is critical to secure ML applications against adversarial attacks and is vital for creating robust applications [35].

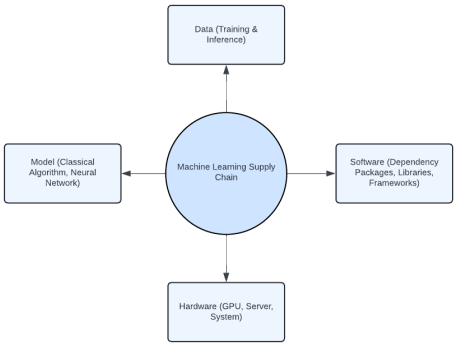


Figure 2. Machine Learning Supply Chain

3 Taxonomy

Extensive research has been conducted to perform different types of malicious attacks against machine learning systems and to study their impact on them [26] [46]. The literature employs various terminologies, methodologies, and approaches to classify adversarial attacks against ML systems. However, the fundamental concepts and attack patterns remain the same. After thoroughly reviewing the literature while focusing on threats against the ML supply chain, this paper focuses on attack types from MITRE [32], which classifies adversarial actions into four classes: 1. Attacks through Data, 2. Attacks through Model, 3. Attacks through Software/Libraries, and 4. Attacks through GPU.

These four attack categories were chosen due to their roles in an ML Supply Chain. Data and models are the fundamental components of any machine-learning application. On the other hand, libraries and software packages are required for various tasks, such as memory management, network handling, data processing, testing, and debugging, in a typical software system design and implementation. Graphical processing units have become an integral component of any deep learning system due to the computational requirements of ever-growing DL architectures, e.g., GPT3, one of the large language models from OpenAI, has approximately 175 billion parameters and 96 attention layers [10], which requires enormous computational resources. Additionally, the literature review indicates that most attack patterns, including Poisoning and Exploratory attacks [14], fall under these four categories.

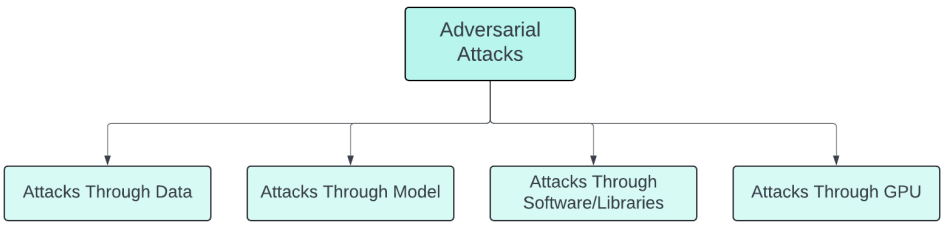


Figure 3. Adversarial Attack Taxonomy

3.1 Types of Attack

3.1.1 Attacks Through Data

Collecting and preparing data is an essential aspect of the ML life cycle since the overall performance of an ML system relies highly on the quantity and quality of data used for its development. Consequently, tampering with the training, testing, or validation datasets could quickly fail an ML system. Data-based attacks, also called White Box [9] or data poisoning attacks, are the types of attacks during which the attacker has access to the training dataset and ML model and possesses knowledge about the model. In this scenario, the attacker can either compromise the data by poisoning it with adversarial samples to produce faulty outputs on specific inputs [36] [3] or analyze the model during training to find vulnerabilities in the feature space [5].

Another potential threat to ML systems arises when individuals download or scrape data from the internet via open-source platforms, such as Kaggle, VisualData, UCL Machine Learning Repository, and Socrata, where individuals and organizations can publish datasets. An adversary could provide poisoned datasets on such platforms. Accessing datasets provided by untrustworthy parties could result in malicious data being fed into the model, akin to a White Box Attack scenario. For example, a malicious actor could distribute a corrupted dataset for a model, leading to the model making incorrect predictions on specific inputs [7].

Data-based attacks usually occur during the data acquisition and preprocessing phases of an ML pipeline, and depending on the attack and the application, they could have severe impacts. For instance, it could lead to the model predicting or generating unexpected or inappropriate outputs, leading to potential reputational damages, financial loss, or fatal decision-making in critical scenarios.

3.1.2 Attacks Through Model

Often, individuals or companies outsource the development or training of an ML model due to the lack of necessary expertise, or they utilize a pre-trained or foundational model, such as CLP, provided by a third party because it would require extensive time to build and train

it from scratch. In the case of outsourcing [22], an attacker might add a backdoor trigger to the model. Generally, the model user will have a validation set to test the model. The adversary aims to produce high accuracy on the validation set and generate wrong predictions when the input contains the specified trigger. In the second scenario, users may download a pre-trained or a foundational model and perform transfer learning for a specific task. The adversary might have trained this particular model on a poisoned dataset or added a backdoor to produce incorrect outputs when the trigger is encountered, as discussed in [27]. Such poisoned models could be found on open-source platforms like Huggingface, Github, and Model Zoo. In some cases, the attacker might be an insider, in which case, the adversary could modify the model parameters [28], which might lead to adverse effects.

Lately, federated learning (FL) has grown to address data privacy concerns and operational challenges to centralizing datasets. While FL offers data privacy and the ability to leverage distributed datasets, it also provides a broader attack surface to adversaries. In FL, the attacker could poison the local version of the model [34], consequently affecting the original model.

Model-based attacks usually take place during the training phase of the machine learning pipeline. When the model is trained on the dataset, injected backdoors learn to provide specific output. These attacks could have severe consequences depending on the application. For example, unauthorized individuals might gain access to confidential information or systems. Similarly, in generative AI, such vulnerabilities might reveal insights into the dataset used to train the model.

3.1.3 Attacks Through Software/Libraries

In modern machine learning systems, applications heavily rely on software packages and third-party libraries, including Model Zoo, Numpy, Fast.ai, and Caffe. These libraries and frameworks offer various techniques for designing and implementing robust ML/DL models from scratch or optimizing pre-trained models. Moreover, they utilize other software packages for different purposes, such as memory management and network access.

Although some libraries and ML frameworks are provided by trust-

worthy parties, such as Meta, Google, and Microsoft, they are still vulnerable since most are open-sourced or provided by untrusted platforms and individuals. For instance, if one of the dependent packages is compromised, it could lead to buffer overflows, segmentation faults, or other unexpected behaviors [23]. Similarly, an adversary can utilize a compromised package to hack into the user's system through the network, perform denial of service attacks, or inject malware into the user's system [12].

Aside from attacking the user's system, an adversary can exploit vulnerabilities in frameworks or their open-source libraries to launch attacks against the model and data. For instance, an adversary might stealthily inject a Trojan horse into the libraries to steal training data from the user without their knowledge [25]. Furthermore, adversaries may impair the performance of a model by poisoning the data through memory corruption [45]. Moreover, accessing fake versions of ML/DL libraries might result in potential harm to the computer system or the machine learning application.

Software-based attacks could affect multiple stages of the ML pipeline, such as preprocessing, training, and evaluation. They could impact the ML system, e.g., compromising data preprocessing could lead to incorrect predictions, or backdoors could be injected into models. Moreover, the attacker could exfiltrate information about the data or the model.

3.1.4 Attacks Through GPU

Graphical Processing Units play an essential role in deep-learning-based systems and offer several benefits, but they also accompany risks. Park et al. [37] showed that an attacker could inject and execute malicious code in the memory of a GPU, potentially degrading the overall performance of the DL algorithm. Furthermore, adversaries can extract information about the data or the model from the GPU. For instance, if an adversary injects a Trojan horse into the GPU, it can observe the state of the GPU while an application executes. This Trojan can extract model parameters, hyperparameters, application types, and other data depending on the memory state. Recently, due to their growing size, an increasing number of DL models are trained on multiple GPUs running in parallel rather than on a single GPU. Attackers can perform side-channel attacks on multi-

GPU architectures to extract information about deep learning architectures from GPU memory, as demonstrated by Dutta et al. [16].

GPU-based attacks usually occur during the training process and could affect both the model and the dataset. Depending on the type of attack, the adversary might inject poisonous samples into the training data, leading to wrong predictions or changing model parameters, which could sabotage the model entirely or partially. Additionally, the attacker could utilize trojan attacks to steal the model or information about the data.

3.2 Analysis of Attacks

The attacks discussed above apply to any machine-learning supply chain. However, the plausibility of different types varies due to various factors, including but not limited to access to the system, skills required, ML knowledge, likelihood of success, and ease of performability. For instance, data poisoning and backdoor injection attacks are relatively easy to perform and more likely to succeed since individuals and small-scale organizations mostly scrape data from the internet, download from an open-source platform, utilize pre-trained models, or outsource model training due to a lack of in-house competence.

Software-based attacks are less common than model and data-based attacks. Most software-based threats stem from the vulnerabilities present in open-source projects unsupported by trusted vendors, such as Amazon, Meta, and Google. These attacks could also happen when users unintentionally download or install packages that claim to be one of the trusted ones but are counterfeits [38]. On the other hand, GPU-based attacks might be the rare types of attacks on the ML supply chain since they require in-depth knowledge of how the GPU works, its memory layout, communication mechanism, and access to the GPU itself.

These attacks are hazardous and could compromise the machine-learning supply chain in one way or another. However, the frequency, effectiveness, and success of attacks are determined by various factors, such as the skills and expertise in machine learning. If proper security mechanisms are not implemented, it could lead to severe consequences. The following section discusses possible countermeasures against these attacks and their applicability.

4 Mitigation Strategies

Various approaches have been employed in the literature as countermeasures against data poisoning, model modification, software compromise, and other threats. However, due to the diverse nature of these attacks, developing one solution that could mitigate all adversarial attacks presents a challenge. The following sections discuss mitigation methods from previous research that are generally recommended or applied.

Attack Medium	**Attack Type**	**Attack Mitigation**
Data	White Box Attack, Data Poisoning	Data Sanitization, Adversarial Sample Detector, Defensive Distillation, Dataset Traceability
Model	Parameters Modification, Backdoors Injection, Compromised Pre-trained Models,	Model Enhancement Techniques,Detection and Backdoor Mitigation Techniques,and Program Verification Techniques
Software / Libraries	Compromised Libraries and Packages, Data Exfilteration, Fake ML/DL Libraries	Secure coding and Verification, Access Control and Resource Management, Vulnerability Detection and Management, Best Practices
GPU	Malicious Code Injection, Data and Model Information Extraction via Trojan, Information Leakage in Multi-GPU Architectures	Access Control and Authorization, Data Flow Management, Anomaly Detection and Operation Protection

Table 1. Attack Types and Mitigation Strategies

4.1 Types of Strategies

4.1.1 Defensive Strategies for Data-Based Attacks

Data poisoning or perturbation in the training samples or inference inputs is common in attacks against ML systems. Consequently, significant research has been conducted to prevent such attacks. Although many countermeasures to counter this attack can be found in the literature, we provide a few below.

4.1.1.1 Data-Centric Defenses

These methods counter data-based attacks by filtering data to remove adversarial samples.

- **Data Sanitization:** Data sanitization refers to filtering the training data to remove poisoned samples [13]. This technique can be applied

before starting the training phase. Although it helps to remove unwanted data points, it relies on assumptions that might not hold in all scenarios [4].

- **Data Augmentation:** Data augmentation could counter adversarial samples in the training set. The idea here is to apply substantial data augmentation, such as mixing up the input samples to create new samples or adding noise that could dilute the impact of poisonous samples [6].

4.1.1.2 Model-Centric Defenses

One method to counter data poisoning is to modify the model architecture or its hyperparameters to avoid the impact of adversarial samples on the learning process.

- **Regularization:** Ross et al. [40] proposed utilizing regularization to counter poisonous samples during training. The main idea here is to reduce the model's sensitivity to small perturbations in the training input by penalizing the gradients of predicted output concerning their inputs. Regularization makes the model more robust to subtle input variations, resulting in better decision-making.

- **Robust Optimization Algorithm:** Optimizing the stochastic gradient descent through an algorithm while training is a potential countermeasure against poisonous samples. Diakonikolas et al. [15] proposed the algorithm "Sever". It applies robust optimization on the gradient descent algorithm either by down-weighing or removing them, thus limiting the impact of poisonous samples.

- **Gradient Shaping:** Hong et al. [24] proposed the idea of gradient shaping, which refers to bounding the magnitude and minimizing the orientation of the gradient. The authors utilized Differentially Private Stochastic Gradient Descent (DP-SGD) to clip the gradient and noise it before updating model weights to mitigate the effects of adversarial samples.

4.1.1.3 Provenance and Transparency Defenses

Data provenance and transparency could be leveraged to avoid feeding poisoned data to the model. These methods could provide insights into the origins of the data, thereby offering an approach to verify the integrity of the data.

- **Dataset Traceability:** Data-based attacks could be mitigated through data traceability. Consumers of the data should verify the origins of data so as not to access datasets from unreliable sources. Tools such as datasheets [18] could provide the user insight into why the data set was prepared, the motivation for making it, and the creator.

4.1.2 Defensive Strategies for Model-Based Attacks

Detecting and countering model-based attacks presents a challenge since they are directly embedded in the algorithm. Depending on the model type and situation, different methods could be applied to handle these attacks. The following sections provide some of the techniques that could be applied,

- **Model Enhancement Techniques:** If a pre-trained model is utilized or the model is outsourced, the manipulated weights or injected backdoor could be fixed through transfer learning [20] or in-house training, resulting in the model forgetting the malicious patterns it learned or being overridden by new data. Secondly, the performing an inhouse.

- **Adversarial Defense Libraries and Best Practices:** The robustness of the ML/DL model could be tested for adversarial attacks through libraries such as MS Counterfeit, Foolbox, DeepSec, TextAttack, and ART. Moreover, utilizing a pre-trained model provided by trusted platforms and vendors, e.g., Google, Microsoft, and Meta, is a good practice.

- **Anomaly Detection and Backdoor Mitigation Techniques:** Machine Learning techniques could be employed to counter backdoors in ML/DL models. For instance, Liu et al. [31] suggested ABS to detect triggers in neural networks by analyzing the behavior of the target

model under different stimulation levels. Another model-based approach is Neural Cleanse (NC), proposed by Wang et al. [43], which employs anomaly detection to identify the presence of backdoor triggers in neural networks.

- **Security and Program Verification Techniques:** Hashing and program verification techniques, commonly used in domains such as cryptography [39], could be adapted to counter model-based attacks. For instance, if the hash of a pre-trained model is available, it could be used to check whether the model has been tampered with. Similarly, program verification might help in verifying the robustness of the model.

4.2 Defensive Strategies for Software-Based Attacks

Machine learning libraries and third-party software packages have become crucial to developing robust ML systems. As such, defending against vulnerabilities in these frameworks is of utmost importance.

- **Secure coding and Verification:** One method of handling vulnerabilities in ML frameworks is to implement secure coding practices and utilize cryptographic hashes for integrity checks. Following the standards would help minimize the risks introduced via design or implementation errors—such errors could expose security vulnerabilities in the final system.

- **Access Control and Resource Management:** Limiting the access of open-source or third-party packages to system and network resources might help mitigate trojans and poisoning attacks. If memory access is limited, malicious packages cannot read and write into the sensitive areas in the memory. Likewise, limiting memory access would prevent the download of poisonous content or the leakage of private data.

- **Vulnerability Detection and Management:** Fuzz Testing [19] might reveal security loopholes within ML frameworks, such as improper memory management, buffer overflow or underflow, and other runtime exceptions. Therefore, such functionalities should be implemented

properly and securely.

- **Best Practices:** Users should adhere to best practices to avoid adversarial attacks, such as downloading software packages exclusively from trusted platforms, verifying package name, version, and source, and following documentation while utilizing libraries or frameworks,

4.2.1 Defensive Strategies for GPU-Based Attacks

Detecting various misconfigurations and adversarial attacks in GPU architecture is significantly more complex than other attacks against the ML Supply Chain. Most machine learning or deep learning applications are developed using GPUs provided by vendors such as NVIDIA. As such, the most efficient method to counter these issues is by introducing security updates in the GPU-related libraries. Some of these techniques are provided below.

- **Access Control and Authorization:** Authorization and access control could provide a defense against these attacks. For instance, requiring administrator privileges to access GPUs could prevent unauthorized access to GPU memory. Likewise, adding features, such as restricting memory read-write access, in GPU-related libraries could provide a defense against malicious code injection into pages [37].

- **Data Flow Management:** In the case of distributed computing, the best method would be to limit the amount of data transfer among individual GPUs. Although GPU-to-GPU communication cannot be eliminated, limiting cross-GPU data transfer rates by keeping threads and the data they access within a single GPU could provide a defense against side-channel attacks [29].

- **Anomaly Detection and Operation Protection:** Wei et al. [44] suggested running a GPU daemon to detect contention, potentially preventing side-channel attacks since the attacker could not exfiltrate information based on timings and access patterns. Modifying GPU schedulers could prevent resource contention by isolating critical tasks and resources. Moreover, denial-of-service (DoS) attacks could be prevented by adding scheduler security checks.

4.3 Analysis of Mitigation Strategies

Theorizing the strategies and implementing them are two different things. The practical application of a strategy comes down to having the necessary skills, knowledge, and resources. Moreover, the applicability of any technique is limited by its effectiveness and scope. Countermeasures against data might be easier to implement in the case of methods such as data sanitization and provenance since these methods do not require as much resources and expertise compared to more sophisticated techniques. However, even that is limited by the data type or availability of data records. On the other hand, model-based techniques require more effort and high-level ML expertise.

The defenses against model-based attacks, adversarial defense libraries, model enhancement techniques, and hash and program verification are more accessible to implement but require resources, such as datasets, verification libraries, and hashes of models. However, ML-based techniques are much more complex and challenging to implement. For instance, detecting backdoors requires a pre-trained model focused on a specific domain, or in-house training of an ML algorithm needs to be performed, which requires dataset and skills.

Solutions for software-based techniques might be the easiest to implement since they require following standard coding practices, error handling, and code reviewing. However, the fact is that avoiding all and every kind of error is impossible, and people make mistakes. Additionally, a vast community is involved in open-source, and rectifying everything becomes challenging. Like software-based strategies, the defense mechanisms against GPU-based attacks are straightforward. If GPU vendors were to follow best practices, provide sophisticated and robust libraries, and implement improved architectural designs, adversarial attacks could potentially be mitigated.

5 Gaps and Challenges

Adversarial Machine Learning (AML) has been a subject of study for a considerable time, with extensive research conducted to detect and mitigate security vulnerabilities in ML systems. Despite vast research, the focus has predominantly been on adversarial attacks against individual components of a machine learning system, such as data poisoning, back-

door injection, and model inversion or evasion attacks. More research is required on software and hardware-related security threats since compromises in these components could lead to severe consequences, as discussed in previous sections. Moreover, most of the research has concentrated on individual machine learning supply chain components, such as data or models, rather than the whole machine learning pipeline, which might lead to overlooking threats stemming from the integrated behavior of these components. Consider a facial recognition system built using high-quality and diverse data, state-of-the-art facial recognition models, trusted deep learning frameworks like PyTorch, and trained on GPUs secure against unauthorized access. Although the ML pipeline comprises secure components, the system can be compromised in several ways. For instance, if the data pipeline is insecure, the adversary could intercept or manipulate data to introduce biases. Moreover, vulnerabilities in third-party libraries, for support purposes, might provide access to adversaries. Therefore, a comprehensive approach that considers individual components and the system as a whole would be more appropriate to implement a robust ML system.

Due to the vast expansion of machine learning, plenty of datasets and ML models are being created and distributed commercially or on open-source platforms. Commercial products are more reliable and secure than the open distribution of these resources, but not everyone can afford them. Additionally, open source is a means for further research and expansion of ML. As such, there is a need for trusted and authorized open-source platforms for sharing datasets, model architectures, and pre-trained weights, as well as a method to verify their integrity. Cryptographic hash or signatures could be used to verify the integrity of data, models, and weights, or a traceability mechanism might be employed to identify the source of these resources. How to design and implement computationally efficient cryptographic hash functions and digital signatures for large datasets, model architectures, and pre-trained weights, which could be integrated into current workflows, and benchmark their robustness against other plausible methods for integrity verification?

Software packages and libraries for ML/DL are increasing exponentially due to their ease of use and the vast tools they provide to build machine learning systems from scratch. Open-source contributions allow anyone to contribute to developing and improving these packages. For instance, the most popular ML/DL libraries, like PyTorch, TensorFlow,

and Keras, are all available on GitHub, and anyone can contribute. However, unlike these popular libraries that large organizations and a massive community are maintaining, some libraries provide many features, such as downloading datasets or pre-trained models, but are not maintained by well-known vendors like Meta and Google or trusted parties, which might lead to potential vulnerabilities that adversaries could exploit. Moreover, detecting vulnerabilities in machine learning code, such as optimization algorithms and cost functions, is more complex than general exploitation scripts. Therefore, there should be proper standards to check and verify these packages or libraries for any bugs or threats. One method to mitigate the effects of malicious code is to develop an adversarial code detection algorithm and integrate it into version control (VC) platforms. Establishing testing mechanisms, especially for libraries not supported by well-known vendors, for pull requests before merging them on VC platforms could also mitigate adversarial effects. When implementing these techniques, their limitations and feasibility should be considered. For example, How to develop computationally efficient and robust adversarial code detection algorithms compatible with version control platforms to detect perturbations in ML optimization algorithms and objective functions?

Compromising GPUs might be more challenging than other machine learning supply chain components. It could be argued that adversaries require direct access to the GPU to compromise it. However, adversaries could also use third-party libraries to inject trojans into GPU since these libraries require certain user privileges to operate them. Depending on the level of access, the ML pipeline could be easily compromised. Additionally, the adversary can have direct access to physical resources, which could lead to severe consequences. Although anti-viruses have been around to counter threats in main memory for quite a while, there is no software or tool to protect GPUs against such security threats. Therefore, developing unique tools to detect and remove trojans from GPUs could mitigate such threats. These methods might mitigate adversarial attempts; however, their feasibility and limitations should be considered. How to implement robust and effective software that can detect and mitigate trojans in the GPU, is compatible with other GPU libraries, addresses the vulnerabilities introduced by third-party libraries, and can maintain the output efficiency of GPUs?

6 Conclusion

The paper provides a comprehensive taxonomy of supply chain attacks against machine learning systems and various defense mechanisms adopted to counter threats and vulnerabilities in ML applications that adversaries may exploit. After reviewing the literature, it is evident that many vulnerabilities in machine learning are often overlooked, which could have severe consequences. Adversaries could compromise the complete supply chain by compromising a single component, for example, leading to fatal decisions by corrupting data. However, most of the research in adversarial machine learning has focused on individual components, especially on data and models, leading to research gaps in other components of the ML supply chain and the system as a whole. Future research should focus on identifying and mitigating threats in all parts of an ML supply chain. Moreover, techniques should be developed to implement more robust and secure ML pipelines.

References

[1] First Initial. or Corporate Author Author's Last Name. Sonatype 9th annual state of the software supply chain report. https://www.sonatype.com/en/press-releases/sonatype-9th-annual-state-of-the-software-supply-chain-report, 2023.

[2] Marieke Bak, Vince Istvan Madai, Marie-Christine Fritzsche, Michaela Th Mayrhofer, and Stuart McLennan. You can't have ai both ways: balancing health data privacy and access fairly. *Frontiers in Genetics*, 13:1490, 2022.

[3] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[4] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *Multiple Classifier Systems: 10th International Workshop, MCS 2011, Naples, Italy, June 15-17, 2011. Proceedings 10*, pages 350–359. Springer, 2011.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[6] Eitan Borgnia, Jonas Geiping, Valeriia Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. Dp-instahide: Provably defusing poisoning and backdoor attacks with differentially private data augmentations. *arXiv preprint arXiv:2103.02079*, 2021.

[7] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. *arXiv preprint arXiv:2302.10149*, 2023.

[8] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[9] 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.

[10] Nagesh Singh Chauhan. Openai gpt-3: Understanding the architecture, 5 2022.

[11] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. *arXiv preprint arXiv:1712.02051*, 2017.

[12] Hongsong Chen, Yongpeng Zhang, Yongrui Cao, and Jing Xie. Security issues and defensive approaches in deep learning frameworks. *Tsinghua Science and Technology*, 26(6):894–905, 2021.

[13] Gabriela F Cretu, Angelos Stavrou, Michael E Locasto, Salvatore J Stolfo, and Angelos D Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 81–95. IEEE, 2008.

[14] Flávio Luis de Mello. A survey on machine learning adversarial attacks. *Journal of Information Security and Cryptography (Enigma)*, 7(1):1–7, 2020.

[15] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606. PMLR, 2019.

[16] Sankha Baran Dutta, Hoda Naghibijouybari, Arjun Gupta, Nael Abu-Ghazaleh, Andres Marquez, and Kevin Barker. Spy in the gpu-box: Covert and side channel attacks on multi-gpu systems. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–13, 2023.

[17] 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 visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[18] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[19] Patrice Godefroid, Michael Y Levin, David A Molnar, et al. Automated whitebox fuzz testing. In *NDSS*, volume 8, pages 151–166, 2008.

[20] Francisco-Javier González-Serrano, Adrián Amor-Martín, and Jorge Casamayón-Antón. Supervised machine learning using encrypted training data. *International Journal of Information Security*, 17:365–377, 2018.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[23] Nima Shiri Harzevili, Jiho Shin, Junjie Wang, Song Wang, and Nachiappan Nagappan. Characterizing and understanding software security vulnerabilities in machine learning libraries. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pages 27–38. IEEE, 2023.

[24] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv preprint arXiv:2002.11497*, 2020.

[25] Chen Hongsong, Zhang Yongpeng, Cao Yongrui, and Bharat Bhargava. Security threats and defensive approaches in machine learning system under big data environment. *Wireless Personal Communications*, 117:3505–3525, 2021.

[26] Olakunle Ibitoye, Rana Abou-Khamis, Ashraf Matrawy, and M Omair Shafiq. The threat of adversarial attacks on machine learning in network security–a survey. *arXiv preprint arXiv:1911.02621*, 2019.

[27] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 349–363, 2018.

[28] Yujie Ji, Xinyang Zhang, and Ting Wang. Backdoor attacks against learning systems. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[29] Hyojong Kim, Ramyad Hadidi, Lifeng Nai, Hyesoon Kim, Nuwan Jayasena, Yasuko Eckert, Onur Kayiran, and Gabriel Loh. Coda: Enabling co-location of computation and data for multiple gpu systems. *ACM Transactions on Architecture and Code Optimization (TACO)*, 15(3):1–23, 2018.

[30] Satyam Kumar, Dayima Musharaf, Seerat Musharaf, and Anil Kumar Sagar. A comprehensive review of the latest advancements in large generative ai models. In *International Conference on Advanced Communication and Intelligent Systems*, pages 90–103. Springer, 2023.

[31] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[32] MITRE ATT&CK®. Aml.t0010. https://atlas.mitre.org/techniques/AML.T0010, n.d.

[33] AKM Iqtidar Newaz, Nur Imtiazul Haque, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A Selcuk Uluagac. Adversarial attacks to machine learning-based smart healthcare systems. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[34] Thien Duc Nguyen, Phillip Rieger, Roberta De Viti, Huili Chen, Björn B Brandenburg, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, et al. {FLAME}: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1415–1432, 2022.

[35] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[36] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[37] Sang-Ok Park, Ohmin Kwon, Yonggon Kim, Sang Kil Cha, and Hyunsoo Yoon. Mind control attack: Undermining deep learning with gpu memory exploitation. *Computers & Security*, 102:102115, 2021.

[38] Cedric Pernet. Pytorch ml compromised. https://www.techrepublic.com/article/pytorch-ml-compromised/, Year. Accessed: 2024-03-09.

[39] Jonathan Protzenko, Bryan Parno, Aymeric Fromherz, Chris Hawblitzel, Marina Polubelova, Karthikeyan Bhargavan, Benjamin Beurdouche, Joonwon Choi, Antoine Delignat-Lavaud, Cédric Fournet, et al. Evercrypt: A fast, verified, cross-platform cryptographic provider. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 983–1002. IEEE, 2020.

[40] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[41] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28, 2015.

[42] Sonatype. 2022 software supply chain report. https://www.sonatype.com/en/press-releases/2022-software-supply-chain-report, 2022.

[43] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[44] Junyi Wei, Yicheng Zhang, Zhe Zhou, Zhou Li, and Mohammad Abdullah Al Faruque. Leaky dnn: Stealing deep-learning model secret with gpu context-switching side-channel. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 125–137. IEEE, 2020.

[45] Qixue Xiao, Kang Li, Deyue Zhang, and Weilin Xu. Security risks in deep learning implementations. In *2018 IEEE Security and privacy workshops (SPW)*, pages 123–128. IEEE, 2018.

[46] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2041–2055, 2019.

[47] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

Version control and reproducibility of Jupyter Notebooks

Sami Laine
sami.v.laine@aalto.fi

Tutor: Sanna Suoranta

Abstract

Jupyter Notebooks are a popular tool for data science, data analysis, and machine learning. They allow the user to combine elements like code, text, and visualisations into a single document. Jupyter Notebooks are often used in research and education, but they are often not version controlled, and they are not guaranteed to be reproducible. This paper discusses the importance of version control and reproducibility in Jupyter Notebooks and provides some useful tools and practices for working with Jupyter Notebooks.

KEYWORDS: Jupyter, Notebook, Jupyter Notebook, version control, Git, reproducibility

1 Introduction

Version control is a crucial tool for modern software development [10]. It allows developers to track changes in the source code and compare versions between local and remote codebases. Using a version control system (VCS) facilitates collaboration with other developers by making it easier to merge code and handle conflicts in changed code. Version control systems also provide mechanisms for reverting code to previous states and

for branching code into separate development lines.

Jupyter Notebooks are a popular tool for data science, data analysis, and machine learning. They are also widely used for prototyping code and as an educational tool. They allow the user to combine elements like code, text, and visualizations into a single document. Jupyter Notebooks are well-suited for use in education because they allow mixing Markdown-formatted course materials with code examples and code assignments. There are also tools for including automatically graded assignments in Jupyter Notebooks.

In this paper, I will discuss the importance of version control and reproducibility in Jupyter Notebooks. I will also discuss the current state of version control and reproducibility in Jupyter Notebooks, and I will recommend some useful tools and practices to be used when working with Jupyter Notebooks.

2 Technologies

The main technologies discussed in this paper are Jupyter Notebooks and VCSs. The following sections provide an overview of these technologies.

2.1 Jupyter Notebooks

Jupyter Notebook is a format for creating and sharing documents that contain executable code, visualisations, code output, and text. The Jupyter Notebook format is based on the JSON data format, and the files have the extension `.ipynb` [9].

Jupyter Notebooks consist of "cells" that can contain either code or text [8]. The code cells are executed in a Jupyter kernel that can be run either locally or remotely. The most popular kernels are Python, R, and Julia. The output of code cells can contain text, images, graphs, and other visualisations.

The text cells are written in Markdown, which is a lightweight markup language that allows the user to easily format text without using a complicated graphical user interface or a more verbose markup such as HTML. However, it is also possible to use HTML in the Markdown cells, as well as to include LaTeX formulas with the help of the MathJax library [4]. Figure 1 displays a snippet of a Jupyter Notebook with code, Markdown, and visualisations.

It is important for the reader to note that the word "notebook" can be used to refer to both the file format [9] and the web user interface (UI) [7]. This paper focuses on platform-independent tools and practices for version control and reproducibility of Jupyter Notebook files, and therefore the terms "notebook" and "Jupyter Notebook" are exclusively used to refer to the file format instead of the web UI. The now-deprecated web UI is referred to using the term "classic Jupyter Notebook interface" instead.

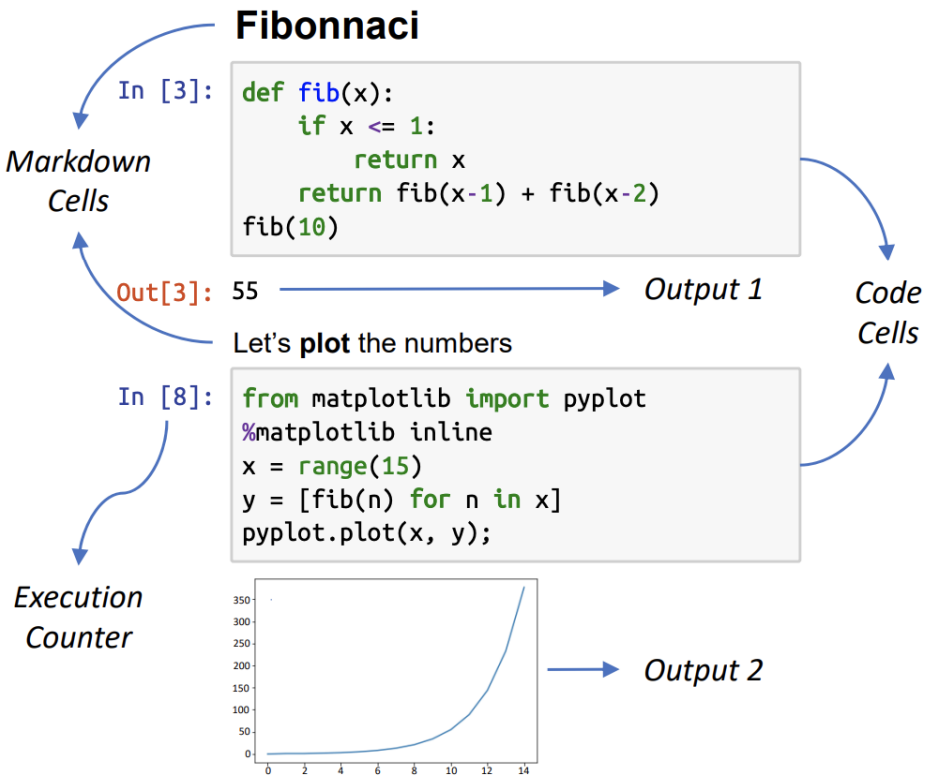


Figure 1. A snippet of Jupyter Notebook depicting code, Markdown, and visualisations [6]

2.2 VCSs & Git

The developer online community Stack Overflow conducted a survey of 71,379 professional and hobbyist developers in 2022 [5]. According to the survey, 95.69% of the respondents reported using some form of version control system in their work. The most popular version control system was reported to be Git, which was used by 93.87% of the respondents.

Git is a distributed version control system that was originally developed by Linus Torvalds for the development of the Linux kernel [3]. It is designed to be fast, efficient, and scalable. Because of its popularity, this paper mostly focuses on the use of Git and related tools for version

control.

While Git is capable of storing any type of file, it is primarily designed for storing and displaying human-readable source code instead of binary files or structured data. Jupyter Notebooks are stored in the JSON format [9]. Even though JSON is a text-based format, it is often not easy to read or edit in its raw form. This makes it difficult to use Git efficiently for version controlling Jupyter Notebooks. However, some tools have been developed to make Git and Jupyter Notebooks inteoperability more convenient.

3 Discussion

Jupyter Notebooks are commonly used for sharing research results, for example in the fields of data science and machine learning. When working with Notebooks, it is important to ensure that the Notebooks are version controlled and reproducible. This section discusses the importance of version control and reproducibility in Jupyter Notebooks and provides some useful tools and practices for working with Jupyter Notebooks.

3.1 Reproducibility

In research, reproducibility is the ability to recreate the results of an experiment or a study using the same data and methods [2]. In the context of Jupyter Notebooks, reproducibility means that the Notebooks can be executed on different machines with the same results. This is important because it allows other researchers to verify the results of a study and to build upon the work of others. However, Jupyter Notebooks contain multiple pitfalls that can make them difficult for other researchers to execute and reproduce.

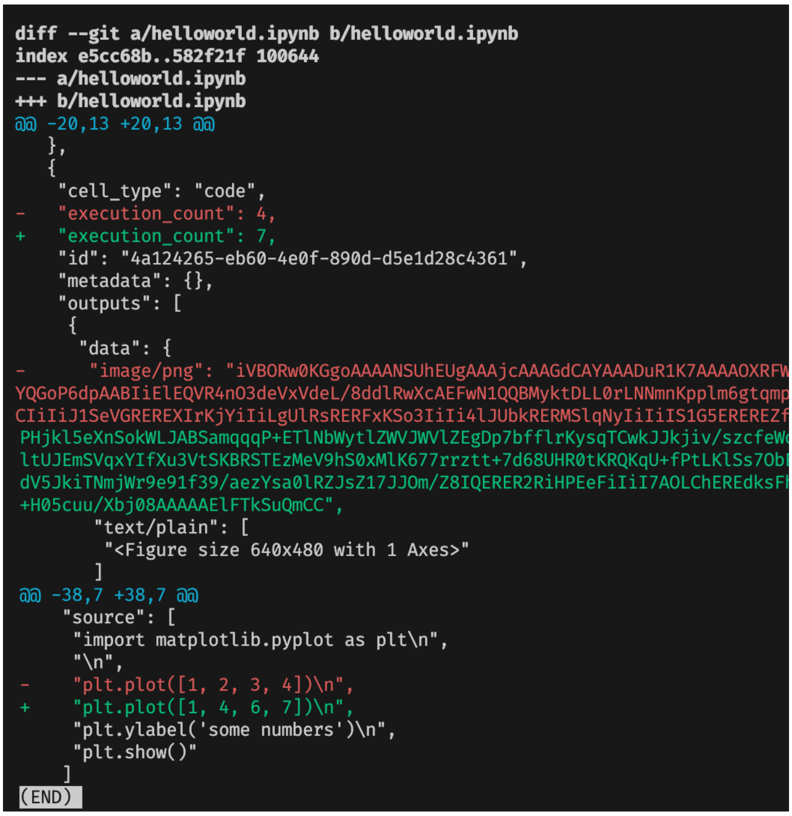
One of the main challenges in reproducibility is the dependency management of the Notebooks [6]. Jupyter Notebooks can contain code that depends on external libraries and packages. If the dependencies are not specified correctly, the Notebooks may not work on different machines or with different versions of the dependencies. The Jupyter Notebook format does not provide a built-in mechanism for specifying dependencies, which means that the researcher publishing the Notebook must manually document the dependencies in the Notebook or in a separate file. The most common way to specify dependencies in Jupyter Notebooks is to use

a `requirements.txt` file or a `setup.py` script [6].

The Jupyter Notebook format allows the user to execute the code cells in any order, which can lead to unexpected results [6]. It is possible to accidentally write code that depends on code cells that have not been executed yet. This can make it difficult to run the Notebook from start to finish, and it can lead to errors that are difficult to debug. To avoid this problem, the user should use the "Restart & Run All" command in the Jupyter Notebook interface before sharing the Notebook with others.

3.2 Version controlling Notebooks

Because of the way that Jupyter Notebooks are stored in the JSON format, it is not easy to view the differences between two revisions of a Notebook using Git. The raw JSON format is not easily human-readable, and it is difficult notice the important and relevant changes in the Notebook files. Figure 2 demonstrates a simple code change that produces a large diff in the raw JSON format because of a change in the resulting figure. The image has been edited to hide roughly 300 lines of Base64-encoded image data. The actual change in the code is only a single line, but it is difficult to notice in the raw JSON diff.



```
diff --git a/helloworld.ipynb b/helloworld.ipynb
index e5cc68b..582f21f 100644
--- a/helloworld.ipynb
+++ b/helloworld.ipynb
@@ -20,13 +20,13 @@
    },
    {
     "cell_type": "code",
-    "execution_count": 4,
+    "execution_count": 7,
     "id": "4a124265-eb60-4e0f-890d-d5e1d28c4361",
     "metadata": {},
     "outputs": [
      {
       "data": {
-       "image/png": "iVBORw0KGgoAAAANSUhEUgAAAjcAAAGdCAYAAADuR1K7AAAAOXRFW
YQGoP6dpAABIiElEQVR4nO3deVxVdeL/8ddlRwXcAEFwN1QQBMyktDLL0rLNNmnKpplm6gtqmp
CIiIiJ1SeVGREREXIrKjYiIiLgUlRsRERFxKSo3IiIi4lJUbkRERMSlqNyIiIiIS1G5EREREZf
 PHjkl5eXnSokWLJABSamqqqP+ETlNbWytlZWVJWVlZEgDp7bfflrKysqTCwkJJkjiv/szcfeWo
 ltUJEmSVqxYIfXu3VtSKBRSTEzMeV9hS0xMlK677rrztt+7d68UHR0tKRQKqU+fPtLKlSs7Ob
 dV5JkiTNmjWr9e91f39/aezYsa0lRZJsZ17JJOm/Z8IQERER2RiHPEeFiIiI7AOLChERERdksF
 +H05cuu/Xbj08AAAAAElFTkSuQmCC",
        "text/plain": [
         "<Figure size 640x480 with 1 Axes>"
        ]
 @@ -38,7 +38,7 @@
     "source": [
      "import matplotlib.pyplot as plt\n",
      "\n",
 -    "plt.plot([1, 2, 3, 4])\n",
 +    "plt.plot([1, 4, 6, 7])\n",
      "plt.ylabel('some numbers')\n",
      "plt.show()"
     ]
 (END)
```

Figure 2. Regular `git diff` output for a Jupyter Notebook

To make it easier to view the differences between two revisions of a Jupyter Notebook, the `nbdime` tool can be used [1]. `nbdime` is a tool for comparing and merging Jupyter Notebooks, and it provides both a command line interface (CLI) and a Web UI for viewing the differences between two revisions of a Notebook. The tool can be integrated with Git to display the differences in a more readable format. Figure 3 shows the output of the `git diff` command when using `nbdime`.

```
nbdiff /tmp/7sYgb1_helloworld.ipynb /tmp/nh48i1_helloworld.ipynb
--- /tmp/7sYgb1_helloworld.ipynb  2024-04-05 21:11:49.730854
+++ /tmp/nh48i1_helloworld.ipynb  2024-04-05 21:11:49.730854
## replaced /cells/1/execution_count:
-  4
+  7

## inserted before /cells/1/outputs/0:
+  output:
+    output_type: display_data
+    data:
+      image/png: iVBORw0K...<snip base64, md5=4518b9081f70c993...>
+      text/plain: <Figure size 640x480 with 1 Axes>

## deleted /cells/1/outputs/0:
-  output:
-    output_type: display_data
-    data:
-      image/png: iVBORw0K...<snip base64, md5=79c8af98f10795a1...>
-      text/plain: <Figure size 640x480 with 1 Axes>

## modified /cells/1/source:
@@ -1,5 +1,5 @@
 import matplotlib.pyplot as plt

-plt.plot([1, 2, 3, 4])
+plt.plot([1, 4, 6, 7])
 plt.ylabel('some numbers')
 plt.show()
```

Figure 3. Output of `git diff` for a Jupyter Notebook, formatted with `nbdime`

The `nbdime` output automatically hides most of the Base64-encoded image data as well as the JSON formatting and focuses on the actual changes in the code. The output clearly displays the changed execution count, summarises the changes in the output cell and shows a clear diff of the changed code.

4 Conclusion

This paper has discussed the importance of version control and reproducibility in Jupyter Notebooks. It has also discussed some of the challenges in producing reproducible Notebooks and has recommended some useful tools and practices for working with Jupyter Notebooks. The paper has also demonstrated how the `nbdime` tool can be used to make it easier to use Git for version controlling Jupyter Notebooks.

Bibliography

[1] Martin Sandve Alnæs and Project Jupyter. *nbdime – Version control integration*. URL: https://nbdime.readthedocs.io/en/latest/vcs.html (visited on 2024-04-05).

[2] Arturo Casadevall and Ferric C. Fang. "Reproducible Science". In: *Infection and Immunity* 78.12 (2010), pp. 4972–4975. DOI: 10.1128/iai.00908-10. eprint: https://journals.asm.org/doi/pdf/10.1128/iai.00908-10. URL: https://journals.asm.org/doi/abs/10.1128/iai.00908-10.

[3] Git. *Git: Fast version control*. URL: https://git-scm.com/ (visited on 2024-01-31).

[4] mathjax. *MathJax – Beautiful math in all browsers*. URL: https://www.mathjax.org/ (visited on 2024-01-31).

[5] Stack Overflow. *Stack Overflow Developer Survey 2022*. URL: https://survey.stackoverflow.co/2022/#version-control-version-control-system (visited on 2024-01-31).

[6] João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. "A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks". In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 507–517. DOI: 10.1109/MSR.2019.00077.

[7] Jupyter Team. *Installing the Classic Jupyter Notebook interface*. URL: https://docs.jupyter.org/en/latest/install/notebook-classic.html (visited on 2024-02-07).

[8] Jupyter Team. *Project Jupyter Documentation*. URL: https://docs.jupyter.org/en/latest/#what-is-a-notebook (visited on 2024-01-31).

[9] Jupyter Development Team. *The Jupyter Notebook Format*. URL: https://nbformat.readthedocs.io/en/latest/ (visited on 2024-01-31).

[10] Nazatul Nurlisa Zolkifli, Amir Ngah, and Aziz Deraman. "Version Control System: A Review". In: *Procedia Computer Science* 135 (2018), pp. 408–415. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.08.191.

The Emerging Role of Neural Networks in Video Coding: A Review

Sarma Rampalli
sarma.rampallisathyabhaskara@aalto.fi

Tutor: Matti Siekkinen

Abstract

The past decade has seen significant rise in video data consumption through new streaming services. Meanwhile, deep learning has also seen applications in various disciples, including video coding. This literature reviews explores cutting-edge video coding techniques that leverage neural networks, either partially in a sub-process or end-to-end. A comparison of these techniques with traditional video coding techniques is done by focusing on key metrics.

KEYWORDS: neural coding, video encoding, video compression, video coding, neural video coding

1 Introduction

In the past decade, the pursuit to enhance visual experiences has seen an increased amount of video content globally. In 2015, 66% of Global IP traffic was video [13], and the data traffic growth will further increase fourfold [28] to 6340 PB by 2025. Globally, 2 trillion minutes of video content crossed the Internet each month [31] in 2017. These examples clearly show an upward trend in the videos that are generated and consumed.

1.1 Traditional Video Coding Techniques

Transmitting such massive volumes of raw videos across networks is not feasible [3]. Hence, video coding is used wherein the video is compressed by encoding into bits at source and decoding from bits to video at target in order to deliver the video content efficiently to audiences at high quality. However, video coding is not a novel process [12]. Jacobs et al. describe [12] how Kell realised, as early as 1929, the need to not just encode-deliver-decode but also do it very effectively. Kell described video compression and patented the idea of transmitting difference between different frames to avoid difficulties [12]. This concept is still used by a variety of video coding standards. International standardization organizations, such as ISO/IEC and ITU-T have published several, widely adopted standards [19].

1.2 Challenges with traditional video coding

While these traditional methods have been widely adopted and have managed to solve the challenges of the present, they have not been particularly helpful to the research world and industry's aspiration to provide superior coding efficiency and provide new immersive experiences [22] within specific domains. Cloud gaming has faced challenges, such as bringing superior coding efficiency of high-quality frames under low-latency constraints [14]. Investigation agencies of police departments need to reconstruct scenes [18]. Governments and regulatory bodies want to identify original videos from the fake videos by applying multiple processing layers [33].

1.3 Neural Networks and Video Coding

Meanwhile, the last decade has seen growing popularity of deep learning in many disciplines such as computer vision and image processing [19]. Neural Networks, which are the result of interdisciplinary research of neuroscience and mathematics, have shown strong abilities in context of non-linear transform and classification [22]. This utilizes a graph concept where the neurons act as processing units and then the degree of relationship between these units mean different things [24]. This has resulted in the growing popularity of neural video coding, where the concepts from neural networks have been used to encode videos at very low bit rates as

well as high levels of quality in particular situations.

This literature review explores neural video coding. Section 2 introduces video coding process. Section 3 identifies various mechanisms through which encoding has been done in past and is being done with neural video coding. Section 4 concludes citing the challenges, limitations and future outlook of neural video coding.

2 Video Coding Process

The end-to-end video process from source to target is explained here, followed by specific mention to video coding process [36].

1. **Video acquisition** [27] provides information about the origin of the video and the scene content. This gives information about the lens used to capture the video, noise distortion, shutter speed and the context of the video.

2. **Pre-processing** [36] performs several actions on the source video to make it ready for encoding. This includes filtering and enhancing signals [25] as per the requirements of the target application.

3. **Video encoding** [12] transforms the pre-processed video into bit streams taking the context of the application that decodes this, and availability of resources to process the frames in parallel. Parallel processing of the frame is done through mechanisms such as raster scan order, slicing, block partitioning [11]. Redundancy in-between frames is used to optimize between frames [18]. This is done by eliminating redundancy between and within frames using motion estimation and compensation techniques [9], removing less important information through quantization techniques [26], storing more information in less bits through efficient techniques like entropy coding based on the degree of variance between frames [15].

4. **Transmission** [8] delivers the video from source to target. Different video streaming options are considered while transmitting the videos, example on-demand, real-time or live streaming. Fault-tolerant mechanisms and recovery [2] are also considered here.

5. **Video decoding** [36] buffers transmitted bits, and further extracts the original video. The decoder also acts on error concealment strategies [34] when there is loss of data during transmission.

6. **Post-processing** [36] aims at enhancing the video quality in addition to converting the decoded video into a format which is understandable to the display.

Neural Video Coding introduces neural networks and deep learning into each of these process steps or sub-processes within the steps 'Video Encoding' and 'Video Decoding' in order to achieve higher coding efficiency.

3 Encoding Techniques

This section looks into the encoding techniques and video encoding standards. In 1995, MPEG-2 [32] was developed by the Moving Picture Experts Group (MPEG) over H.261 with better compression efficiency. In 1996, H.263 [2] brought optimizations to support low bit rate applications. In 1999, MPEG-4 [29] introduced object-based coding, shape coding and content-based scalability for interactive multiple applications. In 2003, Advanced Video Coding (AVC) [35] introduced block-size motion compensation, multiple reference frames and filters to become one of the most widely used video codecs. In 2012, High Efficiency Video Coding (HEVC) [30] was developed on top of AVC to further improve compression efficiency and enabling high quality at low bit rates. While several optimisations have been done to these codecs over time, neural networks have opened up new possibilities in each of the steps.

3.1 Module-level neural intervention

Motion estimation (ME) predicts the difference between frames to reduce redundancy [9]. Motion estimation is done by block-based algorithms in the above traditional codecs. Scale-space flow uses neural network to learn the motion estimation module from scratch [1], and the results have shown that it outperforms other learning-based methods as well as standard codecs, such as AVC and HEVC, in the case of low-latency scenarios.

Motion compensation (MC) reconstructs frames based on the predicted

motion vectors [9]. Traditional codecs have adopted fixed filters and hence the quality of interpolation results are sub-optimal [20]. Dai et al. [7] proposed a variable-filter-size residue learning CNN (VRCNN). This approach proposes a deep-learning based fractional interpolation method to infer sub-pixels for motion compensation and the BD-rate saving is 2.2% better than HEVC on average.

Li et al. [16] argue that deep learning techniques challenge the basic assumptions of traditional predictive coding methods. Li et al. [16] consider residual coding as one specific case of conditional coding. Hence, Deep Contextual Video Compressions (DCVC) is proposed as an alternative. It performs significantly better with bit rate savings of around 26% for HEVC Class B, 5.8% for HEVC Class C low-resolution videos and 11.9% for HEVC Class E small motion videos.

Masked Image Modelling Transformer (MIMT) [38] uses a transformed-based architecture to exploit temporal correlation among frames and spatial tokens in a few auto regressive steps. This method outperforms other popular baselines, such as HEVC. The bit rate saving over VTM is about 22.7% for Peak signal-to-noise ratio (PSNR) and 56.3% over VTM for Structural Similarity Index Measure (SSIM).

3.2 End-to-end neural video coding

While the methods above use deep learning and neural networks within particular modules, several proposals of end-to-end neural video coding have been made in the last 6 years.

In 2018, Wu et al. [37] presented the first end-to-end trained deep video codec. The video compression was done based on the repeated deep image interpolation with a hierarchical setup. It performs better than MPEG-4 and H.261, matches with H.264 and performs close to HEVC.

In 2019, Lu et al. [21] proposed an end-to-end deep video compression (DVC) model. This framework reassesses the end-to-end pipeline of video compression across all key elements, such as motion estimation, motion compensation, residual compression and quantization through the lens of deep learning techniques. For motion estimation, the optical flow estimation technique is integrated into the framework. The test results indicated that this framework outperformed AVC video compression standard.

In 2019, Habibian et al. [10] proposed using rate-distortion autoencoders with finite latent space as well as 3D convolution, alongside an au-

toregressive prior. The results showed that the model outperformed DVC model. Also, the MS-SSIM score improved for foreground (FG) objects compared to background (BG) objects. Also, adaptive compression leverages pretrained video compressors to yield effecient compressed videos, especially in domain-speific cases.

In 2019, Yang et al. [39] proposed Hierarchical Learned Video Compression (HLVC) method with 3 layers of varying qualities in the decreasing order. Layer 1 has highest quality, Layer 2 has medium quality and Layer 3 has lowest quality, and a recurrent. This hierarchical learning employs neural network based techniques, and the layer 3 learns from the other layers so as to improve the spatial and temporal dependencies.

In 2020, Yang et al. [40] suggested Recurrent Learned Video Compression (RLVC) which further improved HLVC using a Recurrent Autoencoder (RAE) and Recurrent Probability Model (RPM). This method maximizes the number of references used to generate new frames and exploits temporal correlation, thereby improving the efficiency. The results from RLVC suggest that this model performs better than DVC model and HLVC models in terms of both PSNL and MS-SSIM. In addition, RLVC inspires hybrid codecs, which employ deep learning, to improve the efficiency.

In 2021, Chen et al. [6] proposed Neural Representation of Videos (NeRV). This method treats videos as neural networks which take frame index as input and give RGB image as output. NeRV comes with limitations such as longer training times, lower compression ratios and lesser performance. Several improvements on NeRV have been introduced to solve these limitations. In 2022, Chen et al. [4] proposed content-adaptive neural representations (CNeRV) which outperforms NeRV by 120x in encoding unseen images. In 2022, Li et al. came up with E-NeRV [17] to introduce separate temporal and spatial context, thereby improving the performance by 8x. In 2023, Chen et al. [5] proposed Hybrid Neural Representation of Videos (HNeRV), wherein combining the explicit and implicit methods of encoding drive improved performance over implicit methods such as NeRV or E-NeRV.

4 Conclusion

The field of neural video coding has witnessed significant advancements in the last decade, with various techniques and architectures emerging to address the challenge of achieving efficient compression while main-

taining high visual quality. Through our review, we have categorized neural video coding techniques into distinct types, such as end-to-end neural video coding techniques, local module-level techniques and hybrid mechanisms where traditional techniques have been combined with neural video coding techniques. Specific focus has been given to the recent advancements in this area, as well as the evidence on how each of the methods have improved the standard metrics like Peak Signal-to-noise radio (PSNR), bit rates and Structural Similarity Index Measure (SSIM).

Despite the advancements in the field, neural video coding still faces challenges and limitations, such as limited temporal redundancy [23], fixed architectures which do not scale for larger videos [23], limited training done with semantic-fidelity oriented compression [22], further rate-distortion optimization [22], high complexity of using deep models, especially in-loop coding modules and temporal successive frames [41]. Continued efforts in research will be essential to drive further innovations to overcome the challenges and limitations.

References

[1] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020.

[2] Tal Anker, Danny Dolev, and Idit Keidar. Fault tolerant video on demand services. In *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*, pages 244–252. IEEE, 1999.

[3] Andreas Burg. Image and video compression: the principles behind the technology. *Curr Probl Dermatol*, 32:17–23, 2003.

[4] Hao Chen, Matt Gwilliam, Bo He, Ser-Nam Lim, and Abhinav Shrivastava. Cnerv: Content-adaptive neural representation for visual data. *arXiv preprint arXiv:2211.10421*, 2022.

[5] Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. Hnerv: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10270–10279, 2023.

[6] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems*, 34:21557–21568, 2021.

[7] Yuanying Dai, Dong Liu, and Feng Wu. A convolutional neural network approach for post-processing in hevc intra coding. In *MultiMedia Modeling:*

23rd International Conference, MMM 2017, Reykjavik, Iceland, January 4-6, 2017, Proceedings, Part I 23, pages 28–39. Springer, 2017.

[8] Ching-Ling Fan, Wen-Chih Lo, Yu-Tung Pai, and Cheng-Hsin Hsu. A survey on 360 video streaming: Acquisition, transmission, and display. *Acm Computing Surveys (Csur)*, 52(4):1–36, 2019.

[9] Borko Furht, Joshua Greenberg, and Raymond Westwater. *Motion estimation algorithms for video compression*, volume 379. Springer Science & Business Media, 2012.

[10] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7033–7042, 2019.

[11] Miska M Hannuksela, Ye-Kui Wang, and Moncef Gabbouj. Isolated regions in video coding. *IEEE Transactions on Multimedia*, 6(2):259–267, 2004.

[12] Marco Jacobs and Jonah Probell. A brief history of video coding. *ARC International*, 1:6, 2007.

[13] Opara Felix Kelechi, Agbaraji Emmanuel Chukwudi, and Aririguzo Marvis Ijeaku. Visual networking index (vni), forecast of global ip traffic increase and ip bandwidth network management. 2011.

[14] Hoang Le, Reza Pourreza, Amir Said, Guillaume Sautiere, and Auke Wiggers. Gamecodec: Neural cloud gaming video codec. 2022.

[15] S-M Lei and M-T Sun. An entropy coding system for digital hdtv applications. *IEEE transactions on circuits and systems for video technology*, 1(1):147–155, 1991.

[16] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021.

[17] Zizhang Li, Mengmeng Wang, Huaijin Pi, Kechun Xu, Jianbiao Mei, and Yong Liu. E-nerv: Expedite neural video representation with disentangled spatial-temporal context. In *European Conference on Computer Vision*, pages 267–284. Springer, 2022.

[18] Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer. *arXiv preprint arXiv:2201.12288*, 2022.

[19] Dong Liu, Yue Li, Jianping Lin, Houqiang Li, and Feng Wu. Deep learning-based video coding: A review and a case study. *ACM Computing Surveys (CSUR)*, 53(1):1–35, 2020.

[20] Jiaying Liu, Sifeng Xia, Wenhan Yang, Mading Li, and Dong Liu. One-for-all: Grouped variation network-based fractional interpolation in video coding. *IEEE Transactions on Image Processing*, 28(5):2140–2151, 2018.

[21] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[22] Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1683–1698, 2019.

[23] Shishira R Maiya, Sharath Girish, Max Ehrlich, Hanyu Wang, Kwot Sin Lee, Patrick Poirson, Pengxiang Wu, Chen Wang, and Abhinav Shrivastava. Nirvana: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14378–14387, 2023.

[24] Reza Pourreza, Hoang Le, Amir Said, Guillaume Sautiere, and Auke Wiggers. Boosting neural video codecs by exploiting hierarchical redundancy. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5355–5364, 2023.

[25] Yunbo Rao and Leiting Chen. A survey of video enhancement techniques. *J. Inf. Hiding Multim. Signal Process.*, 3(1):71–99, 2012.

[26] Michaël Ropert, Julien Le Tanou, and Mederic Blestel. Mastering quantization is key for video compression. *SMPTE Motion Imaging Journal*, 131(5):45–53, 2022.

[27] Oliver Schreer, Ingo Feldmann, Richard Salmon, Johannes Steurer, and Graham Thomas. Video acquisition. *Media Production, Delivery and Interaction for Platform Independent Systems: Format-Agnostic Media*, pages 74–129, 2013.

[28] Hyungsup Shin, Jiyeon Jung, and Yoonmo Koo. Forecasting the video data traffic of 5 g services in south korea. *Technological Forecasting and Social Change*, 153:119948, 2020.

[29] Thomas Sikora. The mpeg-4 video standard verification model. *IEEE Transactions on circuits and systems for video technology*, 7(1):19–31, 1997.

[30] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[31] Forecast Team. cisco.com. `https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_Device_Growth_Traffic_Profiles.pdf`, 2017. [Accessed 01-02-2024].

[32] PN Tudor. Mpeg-2 video compression. *Electronics & communication engineering journal*, 7(6):257–264, 1995.

[33] Mei Wang and Weihong Deng. Deep face recognition: A survey. *Neurocomputing*, 429:215–244, 2021.

[34] Yao Wang and Qin-Fan Zhu. Error control and concealment for video communication: A review. *Proceedings of the IEEE*, 86(5):974–997, 1998.

[35] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[36] Mathias Wien. High efficiency video coding. *Coding Tools and specification*, 24, 2015.

[37] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 416–431, 2018.

[38] Jinxi Xiang, Kuan Tian, and Jun Zhang. Mimt: Masked image modeling transformer for video compression. In *The Eleventh International Conference on Learning Representations*, 2022.

[39] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2020.

[40] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with recurrent auto-encoder and recurrent probability model. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):388–401, 2020.

[41] Yun Zhang, Linwei Zhu, Gangyi Jiang, Sam Kwong, and C-C Jay Kuo. A survey on perceptually optimized video coding. *ACM Computing Surveys*, 55(12):1–37, 2023.

Predicting Depression through Digital Phenotyping

Selin Taskin
selin.taskin@aalto.fi

Tutor: Arsi Ikäheimonen

Abstract

The development of machine learning tools made it easier to process and analyze large amounts of data. Furthermore, the use of different machine learning methods allows the identification of patterns in the data. These developments have a big impact on different industries. Healthcare is one of the areas where the utilization of machine learning methods has become important. Accurate diagnosis of the diseases is essential in assuring correct and personalized treatment of that disease. Depression is one of the most common disorders experienced by people regardless of age. Therefore, there is a need for more objective and trustworthy techniques for diagnosing depression. The increased use of digital devices in day-to-day life benefits the collection of data that can utilised for predicting and diagnosing whether the patient has depression. This paper is a literature-review that investigates the use of different machine learning methods and their performances in the prediction of depression with digital phenotyping. This paper concludes that the use of machine learning methods is becoming an important part of the diagnosis of depression and its severity as they perform well. However, the impact of different demographics and lifestyles on use of machine learning models should be further investigated to understand the performance of the machine learning model more accurately.

KEYWORDS: depression, digital phenotype, machine learning, healthcare

1 Introduction

Depression is one of the most common problems in our modern world. Depression impacts more than 265 million people worldwide [1]. The word depression and the words related to depression such as sadness, isolation, and hopelessness are words that are frequently used in social media. This is an indicator of the prevalence of depression among people. According to the American Psychiatric Association, depression is a severe mental health disease that impacts a person's life in various ways [2]. These ways can be classified into different categories such as the mental effects, and the physical effects. Depression can be caused by different factors. These factors can be related to the genetics of the person or the life conditions of the person. Regardless of the cause of depression, living with depression is difficult and impacts the person's day-to-day life.

The effects of depression can vary from person to person. These effects depend on the severity of the depression the person has. According to a study by Smith, people who experience depression are less motivated [3]. The motivation level of the people who are depressed is significantly less than the people who are not diagnosed with depression. This also results in the decrease of leisure and hobby time of people who have depression. An article by Nimrod et al., noted that people who have depression lose their interest in the activities they once liked [4]. Furthermore, people who have depression tend to isolate themselves more and struggle with seeking help. The isolation and the lack of motivation can further influence the activity levels of the individuals. The impact of physical health is one of the common research topics in the field of depression studies [5]. However, there are significantly fewer studies on the impact of depression on a person's physical health and life.

Digital phenotyping is a clinical technique that is used for collecting data from smart devices to collect and track data from individuals [6]. Digital phenotyping is becoming a popular method for collecting data for analysis of certain health issues. Depression is also among these health issues. Different key elements make digital phenotyping suitable for the collection of data over some time. One of these key elements is that with digital phenotyping the problem of subjectivity and the resulting uncertainty is improved [7]. This results in better analysis and understanding of the indicators. Therefore, it is essential to understand the new approach in digital phenotyping and its limitations on the analysis of ill-

nesses and more specifically in depression.

This paper investigates and reviews the latest approaches in digital phenotyping regarding the prediction of depression and the severity of depression. This paper further discusses the limitations of digital phenotyping with respect to the collection of data and the accuracy of this data for the prediction of mental diseases. This paper is structured into five sections. The second section discusses the background information about digital phenotyping, depression and machine learning. The methodology for the selection of the research papers for the literature review is discussed in the third section. The fourth section discusses the findings of the literature review. Lastly, section five provides the conclusion.

2 Background

2.1 Depression

Depression is a common mental disorder that has negative consequences on an individual's health [8]. Depression can be classified into different categories based on psychosis [9]. Depression impacts many people in their lifetime, therefore it is important to understand and find out ways to personalize the diagnosis of depression and its treatment. There is a wide variety of limitations in the diagnosis of depression among individuals [10]. These limitations are regarding the way the data is collected. For example, most of the assessments for diagnosing depression are done by collecting data through the patient self-reporting. This results in the patient having to recall their memories which may not be very accurate. Hence, this leads to not objective assessment of the current situation of the individual. With the development of digital technologies a new opportunity to understand diagnostics of depression arises. Small wearable digital devices can track changes in the behaviors of the patient[11]. These devices use digital sensors to create numerical medical data that can be used for the assessment of the patient's health. The collection of data in a more objective systematic matter is essential for a more accurate diagnosis of depression.

2.2 Digital Phenotyping

Digital phenotyping is a technique for the collection of data and tracking of different behavioral patterns through digital devices [12]. These devices can be a variety of different devices such as wearable devices like smartwatches, and ActiGraphs. Additionally, there are also mobile phone applications that help with the collection and tracking of data with the help of wearable devices. The figure 1 explains how different devices can be used to gain information about the patient. The same digital device can be used to collect different features, and get information about different aspects of the patients life [13]. After the collection of data, this vast amount of data collected with the digital device can be used for diagnostics of depression by utilising statistical and machine learning methods.

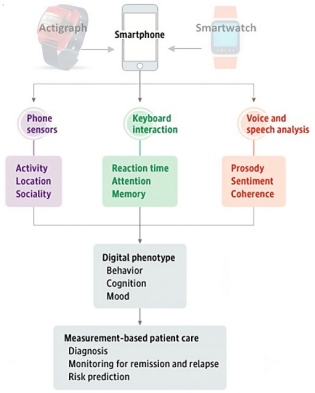


Figure 1. Diagram for Digital Phenotyping [12]

2.3 Machine Learning

Machine learning (ML) is a part of computer science algorithms that learn from their environment [14]. These algorithms use the past data that is collected to predict or detect new outcomes from the new data. Machine learning algorithms can be classified into 4 different categories based on their learning type: unsupervised, supervised,semi-supervised, and reinforcement learning [15].

Machine learning has a lot of different use cases in the field of medicine and healthcare. Using machine learning classes for analyzing patient data to predict diseases is one of the important aspects of ML in healthcare. Both supervised and unsupervised machine learning algorithms can be utilized to understand medical datasets and to further improve the identification of patient classes [16]. With the data collected from digital phenotyping, a machine learning algorithm can be built to process the

data to extract important information regarding to the patient.

3 Methodology

This paper is a literature review of the different statistical and machine learning methods utilizing digital data for predicting depression.

After a preliminary search, it was decided to be more significant to understand the different approaches for predicting depression by leveraging digital data obtained through digital devices. First digital phenotyping with machine learning prediction for depression is investigated. Second statistical methods are used in digital phenotyping for the prediction of depression.

In this literature review, various research papers have been reviewed to understand different approaches in predicting whether an individual has symptoms of depression or not.Therefore, the purpose of this literature review is to discover different research papers with different approaches that can be employed for predicting depression in an individual. The literature review was performed using PubMed.

3.1 Criteria for Literature Review

This section discusses the criteria and the search phrases that were used to determine the papers that are used in the literature review. The search phrases are created to include different aspects of predicting depression through digital phenotyping. The "AND" operator retrieves research papers that include two terms. The operator "OR" was used to ensure that either one of the terms written is included. These search terms were selected as they enable a better understanding of the currently available methods of digital phenotyping in the prediction of depression. Furthermore, there is no specific parameter for the type of device used for digital phenotyping. The focus of this literature review is to understand digital phenotyping for depression analysis from a broad perspective rather than investigating the usefulness of a certain digital phenotyping devices specifically. There when selecting the research papers, the type of digital phenotyping device use was not considered important. The following table, 1, shows the search parameters that are used to query the PubMed database:

Parameter	Number of Research Papers
("Depression"[Title/Abstract]) AND ("Machine Learning"[Title/Abstract]) AND (("digital phenotyping"[Title/Abstract]) OR ("digital phenotype"[Title/Abstract])	106
("Depression"[Title/Abstract]) AND ("Statistical"[Title/Abstract]) AND (("digital phenotyping"[Title/Abstract]) OR ("digital phenotype"[Title/Abstract])	85

Table 1. Table of Literature Review Parameters

Some of the resulting research papers from the queries were not used in the literature comparison. Due to not being eligible for this research. However, some of the research papers that were discarded in the literature selection process still have valuable information regarding background knowledge on digital phenotyping and mental diseases. Therefore, they were utilized in the background section of this paper. Some of the papers that resulted from the query were also literature review-based papers hence they were discarded as the aim of this paper was to understand different techniques used for predicting depression through digital phenotyping and their performances. Figure 2 shows the diagram for selecting the research papers that are considered eligible for this paper and taken into consideration for further examination.

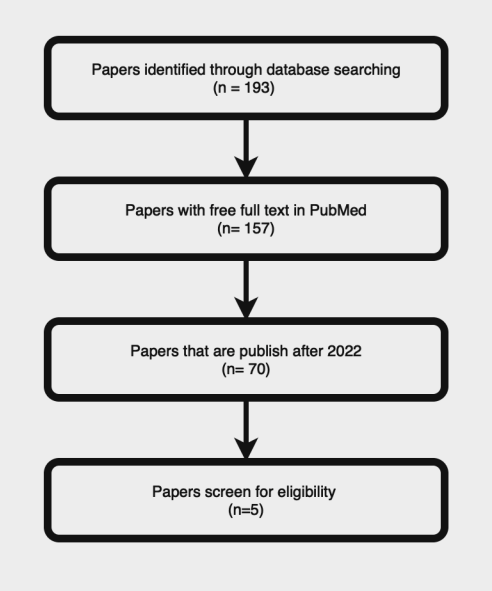


Figure 2. Flow diagram for selecting publications

At the end of the selection, there are 4 articles from the first criteria and 1 article from the second criteria. In this literature review, only articles after the year 2022 were investigated. This literature review is to understand state-of-the-art, there for it is essential to ensure that the research examined in this paper is new and up-to-date. Additionally, the citation indices for the papers were taken into consideration as a good citaiton index indicates that the paper's contribution to this field is significant. Furthermore, no literature review papers were chosen, and all 4 research papers had designed their own experiments for predicting depression.

4 Discussion

This chapter is a discussion of the findings from examining the papers selected based on the criteria explained in Section 3. The first subsection, 4.1, presents the research papers found from the literature search with the given methodology. In this subsection, a comprehensive summary of the literature review-based research findings will be given to summarize the findings and the key advancements in the fields. Finally, the last subsection of this section evaluates the literature search and literature review.

4.1 Selected Literature Papers

This subsection gives a summary of the research papers selected with the methodology given in Section 3.

Each paper had different patient data and different ways of collecting digital data. In these research papers, different machine learning algorithms are utilized to compare the performance for predicting depression by analyzing digital data patterns. In the papers investigated, the results obtained from the mathematical/ machine learning model were compared against the ground truth which is the depression status obtained from a clinical judgment. Furthermore, paper [19] investigated the prediction of the severity of the depression apart from the depression status of the patient. The findings of the paper were positive, which indicates that digital phenotyping can also be used to understand not only if the patient has depression but also the intensity of their depression.

Title of the Research Paper	**Summary**
Digital phenotyping in depression diagnostics: Integrating psychiatric and engineering perspectives [10]	Active data from patient input and passive data from digital phenotyping are collected from the patient. It was found thatthe data can be used to predict depression status. Depression patterns are identified by using ML. The prediction with ML models perform well when compared to the ground truth by clinical judgment.
A Machine Learning Approach for Detecting Digital Behavioral Patterns of Depression Using Nonintrusive Smartphone Data: Prospective Observational Study [17]	Smartphone data were processed to extract features, and built ML model. Different ML models were built: random forest (RF) regression, multivariate adaptive regression, RF classification, XGBoost, and SVM and their performance are compared. RF classification classification performs the best in classifying patient depression with 87% accuracy.
Using digital phenotyping to capture depression symptom variability: detecting naturalistic variability in depression symptoms across one year using passively collected wearable movement and sleep data [18]	Data from wearable movement and sleep device. Stack Ensemble ML model used. ML model built only with data from digital devices had a low mean absolute error. Digital data combined with biodemographic data was used to built another ML mode, which performed better than the ML model built with only digital data.
Improving Depression Severity Prediction from Passive Sensing: Symptom-Profiling Approach [19]	Data such as physical activity, and application usage were collected with a mobile app. XGBoost algorithm was used for ML model. The severity of the depression was also predicted with different algorithms: CatBoost, LightGBM, and SVM. SVM performs significantly worse compared to others. Different operating systems where the mobile app was used resulted in different accuracies.
Predicting Depressive Symptom Severity Through Individuals' Nearby Bluetooth Device Count Data Collected by Mobile Phones: Preliminary Longitudinal Study [20]	Data collected as the nearby Bluetooth device count (NBDC). ML model built with NBDC data, demographics, and questionnaire responds the participants. XGBoost, Lasso,Bayesian linear regression algorithms tested. Bayesian linear regression model performed the best.Significant associations found between NBCD and depression status.

Table 2. Table of Research Papers selected

4.2 Literature Review

This paper has compared various papers to understand the techniques that can be used for prediction of depression in different individuals with the use of digital devices. The search through the literature database of PubMed resulted in only 5 papers. The number of the research paper resulted from the literature search was good amount in regards of the page limitation of the assignment. Regardless, it is essential to discuss the limitation of this paper and the analysis conducted.

5 Conclusion

Analyzing and diagnosing depression through digital phenotyping is a very significant development in the field of health. Different digital devices can be used for collecting data. The accurate collection of this data is essential for building accurate and impactful statistical and machine-learning models.

The purpose of this paper was to investigate the state-of-the-art of the current situation of digital phenotyping and the use of different machine learning techniques to diagnose depression. The research paper database of PubMed was used to access a variety of articles for this article. There were different parameters for the articles chosen such as the publishing date and the keywords of the articles. The articles chosen were not literature reviews, therefore each paper had a different dataset collected from patients, and different methodologies were used. From reviewing these papers, it was found that XGBoost and Random Forest are some of the popular machine-learning algorithms used for the prediction of depression. Additionally, it was found that using the digitally collected data in addition to biomarkers or questionnaires regarding the patient's feelings seems to give a better result in identifying if the patient has depression and the severity of their depression.

This paper discussed the different approaches to predicting depression by using digital phenotyping. The papers examined had different approaches to analyzing and understanding different depression categories. Based on the research papers, it has been discovered that the research direction is focusing more on the utilization of machine learning methods for the prediction of depression. The performance of these machine learning model seem to perform good. Additionally, when incorporated with

additional data obtained through patient surveys, the models performs better.

5.1 Limitations

This paper investigates only a few articles due to the 10-page limitation in the given assignment. Therefore, the paper does not cover the whole state-of-the-art of the digital phenotyping and prediction of depression. A more extensive range of articles must be examined to understand the different approaches for analyzing and processing digitally collected data from digital phenotyping products. Furthermore, it is important to also understand whether the operating system used for the digital device has an impact on the prediction performance of the machine learning model built. Additionally, in the research papers investigated in this paper, the age group of the patients tends to be young, hence it is also crucial to understand if the model built for predicting purposes can give accurate results for older age groups.

References

[1] World Health Organization, "Depression," 2020.

[2] W. C. Depression, "what is depression?" *World Health Organization*, 2012.

[3] B. Smith, "Depression and motivation," *Phenomenology and the Cognitive Sciences*, vol. 12, no. 4, pp. 615–635, 2013.

[4] G. Nimrod, D. A. Kleiber, and L. Berdychevsky, "Leisure in coping with depression," *Journal of Leisure Research*, vol. 44, no. 4, pp. 419–449, 2012.

[5] B. Roshanaei-Moghaddam, W. J. Katon, and J. Russo, "The longitudinal effects of depression on physical activity," *General hospital psychiatry*, vol. 31, no. 4, pp. 306–315, 2009.

[6] T. R. Insel, "Digital phenotyping: a global tool for psychiatry," *World Psychiatry*, vol. 17, no. 3, p. 276, 2018.

[7] J. Torous, P. Staples, and J.-P. Onnela, "Realizing the potential of mobile mental health: new methods for new data in psychiatry," *Current Psychiatry Reports*, vol. 17, no. 602, 2015.

[8] O. P. Almeida, "Prevention of depression in older age," *Maturitas*, vol. 79, no. 2, pp. 136–141, 2014.

[9] S. L. Dubovsky, B. M. Ghosh, J. C. Serotte, and V. Cranwell, "Psychotic depression: diagnosis, differential diagnosis, and treatment," *Psychotherapy and psychosomatics*, vol. 90, no. 3, pp. 160–177, 2021.

[10] J. Kamath, R. L. Barriera, N. Jain, E. Keisari, and B. Wang, "Digital phenotyping in depression diagnostics: Integrating psychiatric and engineering perspectives," *World Journal of Psychiatry*, vol. 12, no. 3, p. 393, 2022.

[11] T. R. Insel, "Digital phenotyping: technology for a new science of behavior," *Jama*, vol. 318, no. 13, pp. 1215–1216, 2017.

[12] P. Bufano, M. Laurino, S. Said, A. Tognetti, and D. Menicucci, "Digital phenotyping for monitoring mental disorders: Systematic review," *Journal of Medical Internet Research*, vol. 25, p. e46778, 2023.

[13] S. D. Dlima, S. Shevade, S. R. Menezes, and A. Ganju, "Digital phenotyping in health using machine learning approaches: scoping review," *JMIR Bioinformatics and Biotechnology*, vol. 3, no. 1, p. e39618, 2022.

[14] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[15] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational vision science & technology*, vol. 9, no. 2, pp. 14–14, 2020.

[16] R. C. Deo, "Machine learning in medicine," *Circulation*, vol. 132, no. 20, pp. 1920–1930, 2015.

[17] S. Choudhary, N. Thomas, J. Ellenberger, G. Srinivasan, R. Cohen *et al.*, "A machine learning approach for detecting digital behavioral patterns of depression using nonintrusive smartphone data (complementary path to patient health questionnaire-9 assessment): prospective observational study," *JMIR Formative Research*, vol. 6, no. 5, p. e37736, 2022.

[18] G. D. Price, M. V. Heinz, S. H. Song, M. D. Nemesure, and N. C. Jacobson, "Using digital phenotyping to capture depression symptom variability: detecting naturalistic variability in depression symptoms across one year using passively collected wearable movement and sleep data," *Translational Psychiatry*, vol. 13, no. 1, p. 381, 2023.

[19] S. Akbarova, M. Im, S. Kim, K. Toshnazarov, K.-M. Chung, J. Chun, Y. Noh, and Y.-A. Kim, "Improving depression severity prediction from passive sensing: Symptom-profiling approach," *Sensors*, vol. 23, no. 21, p. 8866, 2023.

[20] Y. Zhang, A. A. Folarin, S. Sun, N. Cummins, Y. Ranjan, Z. Rashid, P. Conde, C. Stewart, P. Laiou, F. Matcham *et al.*, "Predicting depressive symptom severity through individuals' nearby bluetooth device count data collected by mobile phones: preliminary longitudinal study," *JMIR mHealth and uHealth*, vol. 9, no. 7, p. e29840, 2021.

Traces: How and Where Web Browsing Leaves Them

Shahd Izzeldin Karar Omer
shahd.omer@aalto.fi

Tutor: Tuomas Aura

Abstract

The need to protect user privacy and security becomes apparent as the Internet becomes a necessity in daily affairs. This paper analyzes how users' web browsing activities leave traces accessible to entities ranging from spies to family members and advertising agencies. Evidence of the user's browsing activities is stored in numerous locations including the user's computer, access network and online services. Users wishing to hide or anonymize their web activities must be vigilant to avoid leaving such traces. Changes and countermeasures to reduce the risks listed are Private mode browsing, Virtual Private Networks (VPN), and The Onion Router (TOR) browser.

KEYWORDS: Privacy, Web browsing, Traces, Online tracking

Contents

1 Introduction

The Internet shifted how people conduct everyday affairs, introducing ease and convenience. As a result, people now operate various real-life applications online, increasingly relying on the Internet. This reliance has paved the way for nearly limitless surveillance and privacy breaches, giving rise to the need to safeguard user privacy and security.

Users interact with the Internet primarily through web browsers, which allow them to perform diverse tasks covering research, health and leisure. Web browsing activity can leave behind many traces and personal data. This data has become a valuable asset that many entities can benefit from exploiting, such as notorious advertising agencies, local administrators, nosy family members, Internet Service Providers (ISP), and government surveillance. Consequently, privacy concerns have become a significant issue compared to the last two decades [1]. As privacy concerns continue to escalate, user education becomes increasingly crucial in defending against privacy breaches.

While various studies analyze how and by whom the privacy of web browsing is compromised, the different issues are mainly examined separately, with each paper tackling one aspect or area. This paper aims to aggregate significant practices compromising web privacy.

The main objective is to show the various locations of web browsing traces and possible risks presented in the first three sections. The last section briefly covers countermeasures and their effectiveness. By reviewing these practices, users can better safeguard their online privacy and mitigate the risks associated with unsolicited profiling and tracking.

2 Traces on the User's Computer

The user's local machine serves as the initial point of contact for browsing activities. Although retrieving data from one's computer is more challenging than monitoring data flow on the network, it nonetheless contains abundant traces. Users aiming for anonymous browsing must recognize traces stored locally, which unauthorized individuals such as family members, colleagues or IT administrators could access.

2.1 Browser History and Bookmarks

Browser history and bookmarks are two primary sources of traceable information. The browser history and bookmarks are stored within the browser application or on the local disk and can be used by intruders to deduce browsing activity. Browsing history records URLs, page titles, and timestamps of visited websites, easily accessible from browser settings.

Bookmarks, also known as favourites or internet shortcuts, allow users to save frequently visited URLs for quick access. Despite enhancing user convenience, these features pose privacy risks by exposing browsing habits.

Eliminating the exposure of browser history and bookmarks is relatively easy as long as the user is aware of and vigilant in clearing up these traces after each web browsing session. However, that approach may be less effective due to the introduction of the cloud synchronization feature.

Syncing across devices allows users to transfer their local browsing history, bookmarks, and most data to other owned devices for portability and mobility. Data is shared with the cloud to enable this feature and later distributed among the user's other devices. Consequently, these vivid indicators of browsing activity no longer reside only in the local computer but in the cloud and on multiple devices, adding difficulty to their elimination. The synchronization features typically support deleting history from all the user's devices. However, the data cannot be deleted remotely if the device is offline.

2.2 Cookies and Cache

Cookies are small pieces of data sent by servers to enhance browsing. Since HTTP is stateless, cookies enable servers to recall preferences and session information, creating a smoother browsing experience. Despite their benefits, marketing agencies utilize cookies for tracking purposes, discussed further in the tracking section. For now, it is essential to note that while external entities utilize cookies to retain data, they are stored locally on the user's device.

A *Cache's* function is to store previously performed tasks to speed up future requests and improve performance. Various computing areas implement caching, including memory, disk and the web.

Browser cache, a type of web caching, enables fast retrieval of previ-

ously accessed web pages by storing copies of their content, such as images, scripts, and other resources locally. This effectively reduces web page loading time. However, it requires local data storage, which gives rise to vulnerabilities.

Attackers can exploit browser cache vulnerabilities through various methods, even without physical access to the local machine. For instance, the timing attack [2] capitalizes on the shortened loading time of cached web pages to deduce previous visits.

Browser cache can also be used as a medium to violate the same-origin policy (SOP) [3]. Web browsers implement SOP to limit how different sites interact with resources from different origins, enhancing isolation and allowing users to interact with each website individually without a link between them.

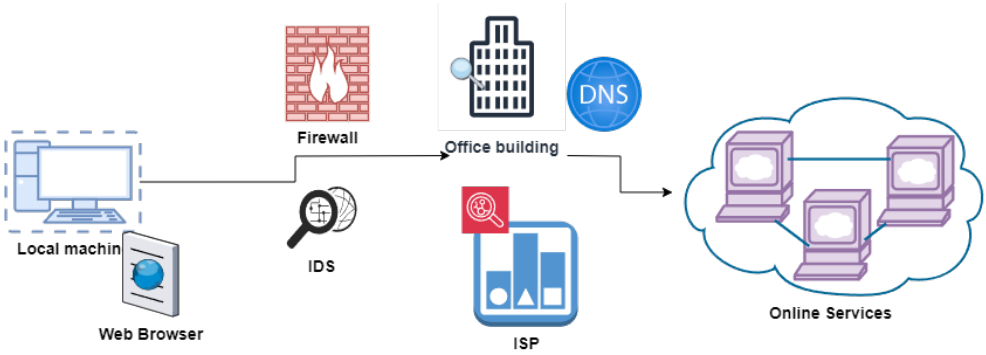


Figure 1. illustrates locations where traces can be left

2.3 Digital Forensics

Forensic analysis is another method for tracing user activity on local devices. Digital forensics involves collecting and examining scattered evidence on user machines for investigative purposes [4].

Despite users' attempts to delete browsing history, forensic techniques can recover deleted records from hard disks, as long as another file's contents have not overwritten them [5].

Additionally, analysts can retrieve further information from other artefacts, such as log and index files [6]. Log file analysis can provide a comprehensive guide to individuals' web usage. For example, *index.dat* in Windows historically served as evidence of online activities, such as visited sites and accessed documents. Furthermore, forensics are capable of recovering histories that have been erased over extended durations [7].

3 Traces Left on the Network

As data exits the user's browser, it travels through numerous stops before reaching its final destination—the server. Various entities can intercept and expose users' data during transit, leaving behind concerning amounts of traces.

3.1 ISP Monitoring

ISPs are companies that provide internet access. Due to their position within the network architecture, ISPs have extensive access to user traffic passing through their infrastructure. Consequently, ISPs can easily monitor and log user web activity. Some ISPs may employ deep packet inspection (DPI) to analyze both header and payload of IP packets, providing insights into browsing activities. Although DPI usage is often justified for purposes such as intrusion detection for network security, copyright infringement, and content regulation [8], it poses significant privacy concerns.

In an influential review conducted by Paul Ohm [9], several motivations for why an ISP would spy on their users were presented in multiple privacy and law conferences. These include envy of Google's success in monetizing and tracking user behaviour for advertising purposes, efforts to conserve bandwidth by blocking data-intensive applications and pressure from third parties such as government agencies and national surveillance.

As ISPs operate under the regulations of the country in which they are based, they typically align their privacy practices with that country's policies, which can range from non-invasive to intrusive.

3.2 DNS pitfalls

The Domain Name System (DNS) converts human-readable text such as website names into their numerical Internet Protocol (IP) address equivalent.

After a client sends a DNS lookup request, the results obtained can be cached to expedite future queries [10]. However, this caching mechanism introduces the first privacy concern within DNS, known as DNS cache snooping[11], a recognized DNS risk that leaves exploitable traces. The

risk associated with caching largely depends on the domain's Time to Live (TTL) settings.

The second vulnerability in DNS arises because DNS data travels in clear text. As established in its first standard by the Internet Engineering Task Force (IETF), DNS traffic between the client and the DNS server is transmitted without encryption [12]. Lack of encryption exposes the DNS traffic and the addresses users are attempting to resolve [11], enabling authoritative agencies to conduct surveillance, censor, and track users.

3.3 Corporate Networks: Firewalls and IDS

Securing browsing history and privacy in corporate environments presents numerous challenges [13]. While essential for safeguarding network infrastructure and defending against cyber threats, security measures often rely heavily on traffic monitoring. These measures, implemented by network administrators in workplaces or educational institutions, raise significant privacy concerns.

During the auditing process to detect threats or anomalies, personal and sensitive information is frequently recorded in log files, with users typically unaware of the contents or potential uses of these logs [14]. Network administrators extensively deploy security measures such as firewalls and Intrusion Detection Systems (IDS) to detect attacks.

Firewalls play a critical role in securing business and institutional networks by filtering traffic between trusted internal networks and external untrusted networks, in adherence to a security policy [15]. They perform packet inspections, examining the source and destination IP addresses to decide whether to accept or reject them.

On the other hand, an IDS is a software or hardware component capable of identifying attacks on computer systems and organizations that traditional firewalls may overlook. IDS collects data reflecting various aspects of users' digital interactions by monitoring files, network traffic, and system behaviour. IDS can detect intrusions by analyzing deviations from established patterns, known as anomaly-based intrusion detection [16]. Privacy-sensitive data collected by IDS may include user usernames and identifiers, DNS logs, hostnames, IP addresses, visited URLs, and timestamps of user events.

4 Traces in Online Services

Online services are the final destination of users' requests. Services play a crucial role as they provide content to users and simultaneously heavily track their digital activity. Conventionally, online services were physical machines but are now primarily deployed in the cloud due to flexibility and scalability, amplifying their capacity to collect and process data on a massive scale.

4.1 Data Collection and Analysis by Websites

Nowadays, users' data is not only shared with the accessed website but also with unrelated third-party websites [17]. These third parties, typically hidden trackers, form ad networks with elements embedded across multiple web pages, allowing them to record and analyze browsing activities to build user profiles.

The primary motivation behind user profiling is online behavioural marketing [18], which aims to maximize profit by persistently identifying users and linking them with their online activities across multiple sites.

Liberat's [19] analysis revealed that among Alexa's top one million websites, 88% of pages initiated requests to third parties. Among these, over three-quarters of the sites initiated requests to domains owned by Google, while approximately 30% of the sites directed requests to Facebook-owned domains. This prevalence of third-party tracking technology across popular sites makes it challenging for users to evade tracking.

Moreover, Ikram et al. [20] uncovered that in 40% of instances, the web ecosystem's dependency chain extends beyond the first-third party dynamic. This expansion entails third parties loading resources from additional domains, fostering implicit trust between the initial first party and any subsequent domain in the chain.

4.2 Stateful Tracking

The prevalent web tracking techniques deployed can either be stateful or stateless. All techniques under stateful tracking require storing information in the user's computer and maintaining information about the user across multiple sessions.

Cookies

At the core of tracking are cookies, invented in 1994 to maintain a state between the user and the site they visit.

Aaron et al.'s [21] empirical study of web cookies found that first-party cookies comprised 32%, whereas third-party cookies made up 68% of those found on client browsers by Alexa's top 100k websites. Notably, almost 90% of these cookies were persistent, meaning they remain stored even after the browser is closed.

Despite SOP governing cookies, allowing them to be initially accessed only by the web pages that set them, tracking across sites is possible due to *cookie syncing*. Cookie syncing is a workaround feature where components from a third-party site are embedded into first-party pages, allowing the exchange of user identification via cookies. Englehardt et al. [22] observed, in an influential experiment monitoring tracking across one million websites, that cookie syncing connected the top 50 third parties with an 85% probability.

Web beacons

Web beacons, also known as pixel tags, work with cookies to enhance tracking capabilities, allowing third parties to identify visitors to a webpage. Web beacons typically consist of small image files that seamlessly blend into the background of a webpage, rendering them entirely transparent. Upon a user's visit to a website, the website automatically downloads the embedded image without the user's knowledge.

Social Integration

Many social media platforms incorporate widgets into websites to enhance user engagement [23]. Widgets are small embedded elements sourced from external platforms that integrate functionality; examples include the Facebook Like button, Google +1, and Twitter tweet, currently X. However, when users interact with these widgets, their data is shared with the respective platform. Consequently, widgets act as agents for third-party tracking.

Another instance of social integration is Single Sign-On (SSO), which enables users to authenticate themselves across platforms managed by identity providers, including Facebook, LinkedIn, Github, or Google [24]. A notable drawback is that the identity provider remains informed of the user's activity whenever they log in using their credentials, enabling

tracking.

4.3 Stateless Tracking

The increasing awareness of privacy concerns and the implementation of regulatory measures such as the General Data Protection Regulation (GDPR) [25] have prompted popular browsers such as Safari and Firefox to integrate built-in protections against third-party cookies. [26]. Google has also announced plans to introduce similar measures starting in 2024.

As measures are taken to prevent stateful tracking, trackers are turning to stateless tracking methods more frequently. Stateless tracking relies on identifying specific attributes of the client's device [27], eliminating the need to store information on the client's computer. As a result, it poses a greater challenge for detection.

Fingerprinting

Browser fingerprinting is a stateless tracking technique that gathers information through a web browser to construct a fingerprint of the device used. This method exploits data exposed by browsers through JavaScript APIs and HTTP headers [28], including device characteristics such as type, operating system, browser version, screen resolution, installed fonts, and browser plugins or extensions.By collecting this information, a tracker can create a browser fingerprint that identifies users and enables tracking. However, unlike deterministic identifiers, a browser fingerprint is statistical [26], meaning its ability to identify a device depends on the uniqueness of the device configuration compared to others.

Early research conducted by Mayer [29] in 2009 highlighted the potential of JavaScript APIs like navigator, screen, and plugins to identify users, leading to the development of browser fingerprinting as an alternative to stateful cookies for web tracking. By hashing the contents of these values, Mayer was able to uniquely identify 96% of client browsers.

In the following year, Eckersley [30] conducted a large-scale experiment through the Panopticlick project on browser fingerprinting that combined additional characteristics of the browser environment, such as timezones and a list of installed fonts to create a user fingerprint. In the experiment, almost half a million browsers were fingerprinted, and about 94% of Flash or Java users had a unique device fingerprint, enabling their identification and tracking.

Despite the decreasing trust in browser plugins and the increasing

popularity of HTML5, browser plugins remain one of the most revealing features, as validated by Laperdrix et al. [31]. Extensions also play an important role, where 9-23% of evaluated extensions were detectable in a study conducted by Nikiforkais [32]. The exact percentage largely depended on the popularity of the extension and thread model.

5 Countermeasures

This section explores countermeasures that mitigate tracking efforts and protect personal data during internet browsing. From private browsing modes to VPNs and the Tor Browser, these solutions offer varying levels of anonymity and security.

Dangers	**Private Browsing**	**VPN**	**Tor Browser**
Local machine traces	✓	×	✓
ISP surveillance	×	✓	✓
Corporate Network	×	✓	✓
Geo-restrictions/censorship	×	✓	✓
DNS leaks	×	✓	✓
Tracking via fingerprinting	×	×	Partial
Tracking via Cookies	×	×	✓

Table 1. Comparison of Countermeasures

5.1 Private browsing and DOT/DOH

Private browsing mode, available in major web browsers, enhances privacy by allowing users to browse the web without storing information on their devices after the session ends. This eliminates any indication of their activities during the browsing session, including browsing history, cookies, and temporary cache files.

While private browsing mode eliminates traces left on the local machine, which a local invader with access to the user's device can target, it does not prevent tracking by other parties, such as ISP and corporate network monitoring or government surveillance.

Private browsing mode effectively reduces third-party stateful tracking via cookies, as cookies do not persist across private sessions and are cleared afterwards. However, it does not effectively address fingerprinting techniques used for stateless tracking [33].

To complement private browsing mode, DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH) can be deployed. These protocols encrypt DNS requests previously sent in the clear, allowing clients to send DNS queries to the resolver over an encrypted transport. However, despite being a significant step for privacy, encrypted DNS queries do not eliminate monitoring or censorship [34]. It is still possible to identify accessed services from the server IP address, TLS handshake, or through traffic analysis. Additionally, DoT and DoH service providers receive a complete log of the DNS requests.

5.2 VPN

Originally developed as a technology for secure data transmission across public networks, VPN has evolved into a privacy-preserving tool with various implementations. A VPN allows clients to connect securely to an intermediate VPN server through an encrypted tunnel [35]. The VPN server then assigns users a different IP address and redirects traffic to the desired destination. This hides personal information and browsing history from third parties such as governments and ISPs. As a result, VPN users can bypass surveillance and access geo-blocked content.

While VPNs provide protection, they are limited in preventing tracking by online services since cookies are stored on the local device. However, VPNs limit what online services can gather about the user since they mask the user's IP address.

Additionally, VPN service providers, particularly free ones, often operate ambiguously without providing sufficient evidence to support their privacy claims [36]. Users can never be sure whether their data undergoes privacy breaches. Trust is transferred from the ISP or corporate network to the VPN service provider when using a VPN.

Some VPN providers have been reported to log user data, sell it to third-party data brokers, and manipulate traffic [36]. Furthermore, traffic leakage can occur unintentionally due to poor security defaults, with DNS leaks and IPv6 leaks being recurring risks [37].

5.3 The TOR Browser

Onion routing, designed to anonymize TCP-based applications like web browsing, operates by constructing a path through a network of TOR routers using layered encryption [38]. As traffic passes through these

nodes, symmetric keys successively unwrap it, resembling layers of an onion. Thus, each node only knows about its predecessor and successor, with only the exit TOR router capable of inspecting the payload and identifying the final destination. This onion-routing network ensures anonymous connections, allowing users to browse public websites without revealing much about their identity to the server.

The TOR Browser, built upon Mozilla Firefox, simplifies connection to the TOR overlay network [39]. It incorporates additional anonymity and privacy features, such as the Tor button, NoScript, HTTPS-Everywhere, and Tor Launcher, with default browsing set to private mode.

The TOR browser prevents user fingerprinting by ensuring uniformity for Tor users, i.e., making one fingerprint for all. Additionally, the Tor browser blocks Canvas and WebGL, removes plugins, and employs a default bundle of fonts [28]. It also masks the OS, consistently reporting the user's machine as Windows. While the TOR Browser offers resistance against user fingerprinting, it still has shortcomings. Nonetheless, it remains superior to previous mechanisms, such as private mode browsing and VPN, which show no resistance to fingerprinting. However, the most prevalent setback about TOR is its speed, where the multiple reroutings make it unsuitable for streaming and gaming. .

6 Discussion

The results of this paper show that user data is a sought-after asset targeted by many entities. Users are far more likely to leave traces behind than to browse the Internet anonymously. For those vigilant about browsing privately, many countermeasures are available, albeit often at the expense of ease and convenience. However, with ongoing research in this field and the frequent proposal of regulations, there is hope that privacy will one day become the default standard in everyday online interactions.

7 Conclusion

This paper investigated and categorized the evidence left of web browsing in different parts of the network. It summarized prominent places with traces linking to user web activity and highlighted the pervasive nature of web browsing tracking. Equipped with this knowledge, users

can proactively protect their privacy and implement the suggested countermeasures outlined in this paper. As the digital landscape continues to evolve, users must remain vigilant and informed, advocating for their right to privacy in an increasingly Internet-reliant world.

References

[1] E. Engström, K. Eriksson, M. Björnstjerna, and P. Strimling, "Global variations in online privacy concerns across 57 countries," *Computers in Human Behavior Reports*, vol. 9, p. 100268, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2451958823000015

[2] E. W. Felten and M. A. Schneider, "Timing attacks on web privacy," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, 2000, pp. 25–32.

[3] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell, "Protecting browser state from web privacy attacks," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 737–744.

[4] M. Reith, C. Carr, and G. Gunsch, "An examination of digital forensic models," *International Journal of digital evidence*, vol. 1, no. 3, pp. 1–12, 2002.

[5] S. Pretorius, A. R. Ikuesan, and H. S. Venter, "Attributing users based on web browser history," in *2017 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE, 2017, pp. 69–74.

[6] J. Oh, S. Lee, and S. Lee, "Advanced evidence collection and analysis of web browser activity," *Digital investigation*, vol. 8, pp. S62–S70, 2011.

[7] M. R. Jadhav and B. B. Meshram, "Web browser forensics for detecting user activities," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 07, pp. 273–279, 2018.

[8] R. Bendrath and M. Mueller, "The end of the net as we know it? deep packet inspection and internet governance," *New Media & Society*, vol. 13, no. 7, pp. 1142–1160, 2011.

[9] P. Ohm, "The rise and fall of invasive isp surveillance," *U. Ill. L. Rev.*, p. 1417, 2009.

[10] T. Callahan, M. Allman, and M. Rabinovich, "On modern dns behavior and properties," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 7–15, 2013.

[11] S. Bortzmeyer, "Dns privacy considerations," Tech. Rep., 2015.

[12] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu, "An end-to-end, large-scale measurement of dns-over-encryption: How far have we come?" in *Proceedings of the Internet Measurement Conference*, 2019, pp. 22–35.

[13] D. P. Bhave, L. H. Teo, and R. S. Dalal, "Privacy at work: A review and a research agenda for a contested terrain," *Journal of Management*, vol. 46, no. 1, pp. 127–164, 2020.

[14] E. Lundin and E. Jonsson, "Anomaly-based intrusion detection: privacy concerns and other problems," *Computer networks*, vol. 34, no. 4, pp. 623–640, 2000.

[15] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of vulnerabilities in internet firewalls," *Computers & Security*, vol. 22, no. 3, pp. 214–232, 2003.

[16] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[17] C. E. Wills and M. Zeljkovic, "A personalized approach to web privacy: awareness, attitudes and actions," *Information Management & Computer Security*, vol. 19, no. 1, pp. 53–73, 2011.

[18] B. Ur, P. G. Leon, L. F. Cranor, R. Shay, and Y. Wang, "Smart, useful, scary, creepy: perceptions of online behavioral advertising," in Proc. Eighth Symposium on Usable Privacy and Security, 2012, pp. 1–15.

[19] T. Libert, "Exposing the invisible web: An analysis of third-party http requests on 1 million websites," *International Journal of Communication*, vol. 9, p. 18, 2015.

[20] M. Ikram, R. Masood, G. Tyson, M. A. Kaafar, N. Loizon, and R. Ensafi, "The chain of implicit trust: An analysis of the web third-party resources loading," in *The World Wide Web Conference*, 2019, pp. 2851–2857.

[21] A. Cahn, S. Alfeld, P. Barford, and S. Muthukrishnan, "An empirical study of web cookies," in Proceedings of the 25th international conference on world wide web, 2016, pp. 891–901.

[22] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, 2016, pp. 1388–1401.

[23] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in IEEE symposium on security and privacy, 2012, pp. 413–427.

[24] D. Fett, R. Küsters, and G. Schmitz, "Spresso: A secure, privacy-respecting single sign-on system for the web," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1358–1369.

[25] M. Degeling, C. Utz, C. Lentzsch, H. Hosseini, F. Schaub, and T. Holz, "We value your privacy... now take some cookies: Measuring the GDPR's impact on web privacy," *Informatik Spektrum*, vol. 42, pp. 345–346, 2019.

[26] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors," in IEEE Symposium on Security and Privacy (SP), 2021, pp. 1143–1161.

[27] T. Urban, M. Degeling, T. Holz, and N. Pohlmann, "Beyond the front page: Measuring third party dynamics in the field," in Proceedings of The Web Conference 2020, 2020, pp. 1275–1286.

[28] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, “Browser fingerprinting: A survey,” *ACM Transactions on the Web (TWEB)*, vol. 14, no. 2, pp. 1–33, 2020.

[29] J. R. Mayer, “" any person... a pamphleteer" internet anonymity in the age of web 2.0,” 2009.

[30] P. Eckersley, “How unique is your web browser?” in Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings 10, 2010, pp. 1–18.

[31] P. Laperdrix, W. Rudametkin, and B. Baudry, “Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints,” in *Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 878–894.

[32] O. Starov and N. Nikiforakis, “Xhound: Quantifying the fingerprintability of browser extensions,” in IEEE Symposium on Security and Privacy (SP). IEEE, 2017, pp. 941–956.

[33] Y. Wu, P. Gupta, M. Wei, Y. Acar, S. Fahl, and B. Ur, “Your secrets are safe: How browsers’ explanations impact misconceptions about private browsing mode,” in Proceedings of the 2018 World Wide Web Conference, 2018, pp. 217–226.

[34] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, “Encrypted dns double right arrow privacy? a traffic analysis perspective,” in 27Th Annual Network And Distributed System Security Symposium . INTERNET SOC, 2020.

[35] J. A. Donenfeld, “Wireguard: Next generation kernel network tunnel.” in *NDSS*, 2017, pp. 1–12.

[36] M. T. Khan, J. DeBlasio, G. M. Voelker, A. C. Snoeren, C. Kanich, and N. Vallina-Rodriguez, “An empirical analysis of the commercial vpn ecosystem,” in Proceedings of the Internet Measurement Conference 2018, 2018, pp. 443–456.

[37] V. C. Perta, M. V. Barbera, G. Tyson, H. Haddadi, and A. Mei, “A glance through the vpn looking glass: Ipv6 leakage and dns hijacking in commercial vpn clients,” *Proceedings on Privacy Enhancing Technologies*, vol. 1, no. 11, pp. 1–15, 2015.

[38] R. Dingledine, N. Mathewson, P. F. Syverson *et al.*, “Tor: The second-generation onion router.” in *USENIX security symposium*, vol. 4, 2004, pp. 303–320.

[39] A. K. Jadoon, W. Iqbal, M. F. Amjad, H. Afzal, and Y. A. Bangash, “Forensic analysis of tor browser: a case study for privacy and anonymity on the web,” *Forensic science international*, vol. 299, pp. 59–73, 2019.

Side Channel and Fault Injection Analysis of Trusted Execution Environments

Sudharsun Lakshmi Narasimhan
sudharsun.lakshminarasimhan@aalto.fi

Tutor: Dr. Lachlan Gunn

Abstract

Trusted Execution Environments (TEEs) are designed to protect sensitive applications; however, they remain susceptible to attacks that leverage legitimate channels. This paper examines various attack vectors, such as fault injection, cache-timing, Spectre and Meltdown, to provide a comprehensive understanding of their security implications. By dissecting these attacks, the paper enables an informed approach to TEE security.

KEYWORDS: Intel SGX, ARM TrustZone, Speculative Execution, Cache-Timing, Rowhammer, Voltage Glitching

1 Introduction

Program execution confinement techniques have significantly improved security measures by isolating and terminating misbehaving programs. By leveraging virtual memory and paging mechanisms, applications are shielded from malicious interference. However, challenges persist, when attackers exploit vulnerabilities to gain elevated privileges and manipulate the underlying operating system. Trusted Execution Environments (TEEs) [17] address these challenges by ensuring the secure execution of sensitive applications despite untrusted operating systems. TEEs, such

as Intel's Software Guard Extensions (SGX) [7] and ARM Limited's TrustZone [14], employ distinct technical approaches, utilizing isolation and confidentiality measures to protect critical operations such as banking and data encryption in untrusted computing environments.

Despite their technical resilience, prominent TEEs such as SGX and TrustZone remain vulnerable to attacks, often due to misuse of communication channels or optimization strategies. This paper explores representative classes of side-channel and fault injection attacks, focusing on SGX and TrustZone. Recent studies [5, 20] indicates that even the latest generation of CPUs remains susceptible to longstanding side-channel attacks, partly due to fragmented information about various vulnerabilities in different TEEs. Therefore, in contrast to papers that focus solely on specific environments [2, 10], this study offers a comprehensive comparison across multiple environments to underscore the security attributes of each TEE. Additionally, such a comparison consolidates information on TEE-related attacks and helps inform the community to prevent recurrent attacks.

2 Background

2.1 Trusted Execution Environments

TrustZone

ARM TrustZone [14] is a hardware-based security feature that establishes two distinct environments within a single processor: a Normal World and a Secure World. In the Normal World, regular operating systems and applications such as Android or iOS run, while the Secure World handles sensitive tasks such as encryption and authentication. TrustZone ensures separation between these worlds, preventing normal world applications from interfering with secure world operations, even if the normal world is compromised. When sensitive tasks are needed, the processor switches to the Secure World, managed by TrustZone, ensuring secure execution. This switching occurs through the Monitor mode overseeing the process by saving and restoring the state of each world during transitions.

An important feature of TrustZone is the NS (Non-secure) bit in the Configuration Register of the processor. When set to 1, it indicates normal world operation; when set to 0, it indicates secure world operation. This bit enables seamless switching between modes and is crucial for other

components within the System on Chip (SoC) to discern between normal and secure world operations. For example, the memory controller can enforce distinct security policies based on the NS bit, ensuring proper handling of memory accesses depending on data sensitivity. In ARM processors equipped with TrustZone, cache memory is adapted to accommodate both secure and normal world requests, with each cache line's tag including the NS bit for security context identification. This adaptation facilitates efficient memory request distinction and eviction, allowing secure and non-secure data to coexist while upholding security.

Intel SGX

Even when facing threats such as untrusted OS or application subversion, SGX [7] ensures sensitive data remains secure. SGX introduces privileged instructions for the creation and management of enclaves, which are hardware-protected areas ensuring confidentiality and integrity. Enclaves and their data structures reside in the Enclave Page Cache (EPC), a portion of the Processor Reserved Memory (PRM). The PRM is a subset of dynamic random access memory (DRAM), which is inaccessible to other software and peripherals. The CPU encrypts enclave data upon leaving the processor, ensuring its confidentiality. Enclaves typically have their own sections for code, data and other resources, similar to non-enclave programs. These sections are encapsulated within the enclave's virtual address space, which is isolated and encrypted by the encryption unit.

3 Attacks on TEEs

3.1 Fault Injection

Fault injection attacks involve inducing faults in devices during operation to analyze their effects, potentially extracting sensitive information. Techniques include power, clock and voltage glitching. Various fault models exist for cryptographic systems, such as single-bit, byte and multiple-byte fault models, each with different precision levels. Injecting precise bit faults is challenging, while inducing faults across bytes is comparatively easier, extracting secret information from them is complex.

Voltage Glitching

The VoltJockey attack [15, 16] targets multi-core systems, where an attacker core manipulates voltage levels to induce hardware faults on victim cores and extracts confidential data through differential fault analysis [9]. Assumptions include the presence of multiple cores with Dynamic Voltage and Frequency Scaling (DVFS) capability and the attacker's permission to adjust voltage and frequency settings. The procedure involves configuring the processor with the appropriate voltage, waiting for target function execution, controlling fault injection points, inducing hardware faults and restoring voltage to avoid detection. In Intel architecture, key parameters for attackers include processor frequency, glitch voltage for inducing faults, voltage for maintaining frequency, waiting time before victim function execution, delay before specific victim code execution and glitch duration. Voltage management involves adjusting the processor's voltage offset via software, while the frequency remains constant.

In Rich Execution Environment (REE), due to fixed operations on plaintext in every round, the waiting time is not considered. To ensure precise duration for fault injection, a custom kernel module is developed, facilitating the execution of the attack procedure. This module can be dynamically loaded during the attack without integration into the kernel image.

In Intel SGX, where physical cores are shared with the normal environment, processor voltage can be altered similarly to REE. AES encryption embedded in an SGX-based program, bound to the victim core, is vulnerable to the attack. Many of SGX's checks are hardware-based and exert minimal influence on the execution time once they are loaded into the EPC. After fault injection, differential fault analysis [9] yielded enclave-protected encryption keys in SGX. The attack exposes gaps in data handling checks during execution, highlighting the importance of safeguarding execution integrity alongside encryption and authentication.

In ARM TrustZone, instead of fixed frequencies, there are two parameters: Fa for the attacker core frequency and Fv for the victim core frequency. Initially, the attacker core operates at a high frequency while the victim core runs at a low frequency. A temporary voltage surge is then applied, sufficient for the attacker core but inadequate for the victim core, inducing errors in the latter. These errors undergo differential fault analysis. Voltage regulation in ARM architecture involves modifying the voltage regulator driver to accept voltage down-tuning only when the frequency is reduced, enabling the attack. TrustZone, similar to SGX,

incorporates hardware-based security mechanisms that minimally affect the timing parameters used between REE and TrustZone use cases.

Rowhammer

Rowhammer is a hardware exploit that targets the DRAM, allowing attackers to manipulate stored bits without direct access. By repeatedly accessing specific rows, electromagnetic interference between adjacent rows induces bit flips, compromising data integrity and posing security risks.

The SGX-Bomb attack [8] exploits the Memory Encryption Engine (MEE) authentication checks in Intel SGX via the Rowhammer attack to lock the processor. While SGX is designed to safeguard against physical attacks on memory integrity by employing a drop-and-lock policy [6], the SGX-Bomb attack allows attackers to utilize these protections to affect the availability of the systems. The attack involves identifying conflicting rows in the EPC region, locating interleaved row addresses within the same bank for the Rowhammer attack and inducing bit flips to trigger processor locking. Specific enclave code executes the Rowhammer attack, causing memory corruption that the MEE interprets as a hardware integrity violation, leading to processor lock-up. This attack poses a significant security threat, allowing unprivileged programs to incapacitate the processor, leading to a system-wide shutdown and loss of control.

In a Rowhammer attack against TrustZone [1], a malicious module operating in the non-secure environment generates faults in specific memory locations using high-frequency memory read operations. These faults extend to adjacent secure memory regions, causing corruption in critical data, such as private RSA keys stored in TrustZone memory. Through inducing these faults and observing resulting changes, attackers can deduce the private key via differential fault analysis [9], compromising the integrity of the secure world.

The above Rowhammer attacks on Intel SGX and ARM TrustZone differ in capabilities. SGX-Bomb aims to compromise system availability by inducing processor lock-up through memory corruption, posing risks, especially in shared cloud environments. In contrast, TrustZone attacks aim to extract sensitive data compromising TrustZone's security.

3.2 Cache-Timing

Cache-timing side-channel attacks exploit cache memory access times to extract sensitive data. The Prime+Probe technique involves saturating

the cache with attacker data and then observing delays caused by victim memory accesses. By analyzing these delays, attackers can infer the victim's accessed memory locations, potentially revealing sensitive information. This underscores the vulnerability of shared cache memory, allowing attackers to monitor and extract secret data from victim processes.

Both TrustZone and SGX cache side-channel attacks discussed in [13, 21] respectively use Prime+Probe for the attack. In TrustZone architectures, cache contention occurs when normal world processes compete for cache space with secure world data, leading to normal world cache lines replacing secure world cache lines. Exploiting this contention, attackers observe cache access timing or patterns to gain insight into secure world activities, potentially revealing sensitive data. This attack utilizes the shared cache mechanism to indirectly access and analyze secure world data through cache contention. High-precision timer ARMv7 PMCCNTR was employed to measure cache access times.

The SGX CacheZoom side-channel attack on Intel processors assumes root access to an OS, allowing kernel module installation and boot property configuration. The attack involves installing a malicious kernel driver for kernel-level control and running the enclave and attacker's code on the same CPU core. Utilizing a high-precision timer with root access, the attacker intermittently interrupts enclave execution to capture fine-grained cache usage information. Through a pointer-chasing technique, controlled data is populated in cache sets, enabling analysis of cache responses to memory accesses. Before each interruption, the L1D cache state is examined and afterward, the cache is primed for probing. Analysis of cache state fluctuations allows inference of enclave-accessed data. This comprehensive approach enhances temporal accuracy for the targeted L1 cache in SGX environments compared to other attacks, due to strategies such as core isolation, interrupt-driven sampling and fine-grained cache control.

3.3 Spectre

Speculative execution, a feature in processors, executes certain instructions before prior instructions complete execution, potentially enhancing program performance. However, if speculation is incorrect, these results are discarded, incurring a performance overhead. Even though the speculatively executed results are discarded, the microarchitectural changes persist, providing an avenue for attackers to deduce sensitive information, for instance, through cache timing. To enhance speculation accu-

racy, processors use branch prediction logic, learning from past branches to predict their outcomes. In one variant of attack, *Branch Target Injection (BTI)*, attackers identify specific regions within a victim's process code that reveal secret data. Attackers craft their program with an identical indirect branch as the victim's and trigger it repeatedly, training the processor's branch predictor to expect the next instruction to be a return to the code that exposes secret data. When the victim's process executes the same branch, the branch predictor speculatively executes instructions based on its learned behavior from the attacker's crafted section, inadvertently loading attacker-controlled data into the processor's cache, which was accessed based on the secret value. Analyzing cache timing discrepancies, attackers deduce the contents of the secret data. The vulnerability lies in the processor's branch predictor, which fails to distinguish between the virtual addresses of different processes.

The SgxPectre [3] attack in Intel SGX enclaves, assumes a prior knowledge of the enclave program's unchanging binary code. The attacker initially manipulates the Branch Target Buffer (BTB) outside the enclave to redirect branch instructions, such as ret instructions, to locations containing secret-leaking instructions instead of their intended targets. Attack preparations include flushing the victim's branch target address and depleting the Return Stack Buffer (RSB), which is a fixed-size buffer that provides predictions for RET instructions, ensuring CPU reliance on the BTB. Once prepared, the attacker sets register values for speculatively executed secret-leaking instructions, enabling them to access enclave memory targeted by the attacker. Cache traces left by these instructions are monitored and a cache timing attack is employed to extract sensitive data effectively. Analysis of cache access patterns allows the attacker to infer the values loaded into registers during speculative execution, compromising the security of the SGX enclave.

Spectre variants have been demonstrated in ARM architectures; however, existing research has not yet revealed any instances of these attacks targeting TrustZone. The absence of documented instances of Spectre attacks on TrustZone presents an area for further investigation and analysis, discussed in Section 5.

3.4 Meltdown

In Meltdown [11], the attacker initiates a user-level program with malicious intent, attempting to access kernel memory, which is typically pro-

tected and inaccessible from user space. Despite the security measures in place, the processor speculatively executes the unauthorized memory access instructions. Although the unauthorized memory access would normally result in an exception and termination of the program, the speculative execution proceeds, potentially allowing the attacker to load sensitive kernel data into CPU caches. The attacker then utilizes timing-based side-channel attacks, such as cache timing analysis, to infer the contents of the accessed kernel memory. By analyzing the discrepancies in memory access times, the attacker can deduce the value of specific memory locations, thereby compromising sensitive kernel data.

Meltdown is only exploitable when the processor switches between different Exception Levels [12] within the same memory translation mechanism. Consequently, it cannot be utilized to access secure memory from the non-secure world, making it inapplicable for TrustZone environments.

One of the key protections provided by SGX is the *abort page semantics*, where attempts to access SGX memory using illegal memory accesses, such as those used in the Meltdown attack, result in all writes being ignored and all reads returning a constant value, typically xFF. However, Foreshadow [19] circumvented these protections by targeting the "present" bit in the page table entry. This bit determines whether a particular memory page is currently loaded into physical memory or has been swapped out to disk. By manipulating the "present" bit to indicate that a page is not present in physical memory, Foreshadow induced translation faults when attempting to access enclave data. These translation faults bypassed SGX's abort page semantics and allowed attackers to perform Meltdown-style attacks, loading and retrieving data from the L1 cache based on the enclave secret value. Even though the L1 cache size is typically limited, usually around 32 kilobytes, by leveraging SGX's mechanisms for moving pages between encrypted memory and regular memory, attackers could bring enclave data into the enclave page cache, effectively loading it into the L1 cache. This allowed attackers to access potentially gigabytes of enclave data from the cache, bypassing size limitations and further compromising the security of SGX enclaves.

4 Defenses

In VoltJockey, restricting software-controlled voltage changes in Intel architecture mitigates attacks against SGX. TrustZone can prevent attacks

by checking regulator driver integrity through storing credentials securely in the boot image and comparing credentials during execution with those in the boot image preventing driver revision as seen in attacks.

Cache prefetching in cryptographic libraries defends against cache timing attacks. It involves loading key-dependent data or instructions into the cache preemptively, reducing the temporal resolution available to attackers. By filling the cache before it is probed, prefetching mitigates the risk of unauthorized access to sensitive cryptographic information.

To mitigate the SGXBomb vulnerability, alternative policies to the drop-and-lock approach, when integrity checks fail, are required. One solution, as outlined in the paper [8], entails implementing drop-and-lock only for SGX. This would mean disabling all subsequent SGX operations if data corruption within an enclave occurs. While this strategy does not entirely eliminate the denial-of-service (DoS) risk, it effectively prevents the entire system from halting or restarting, ensuring the uninterrupted functionality of non-enclave processes. In a Rowhammer attack against TrustZone, Secure memory must be isolated from non-secure memory in DRAM to prevent leakage of confidential information. Also, encryption and authentication mechanisms can be implemented to safeguard data.

Kernel Page Table Isolation (KPTI) divides address space into separate page tables for user and kernel modes. Prevents user programs from accessing sensitive kernel memory, mitigating Meltdown vulnerabilities. One mitigation for Foreshadow involves the implementation of a new instruction to flush the L1 cache during page-in and page-out operations.

The Spectre Branch Target Injection variant is mitigated by deploying distinct branch target buffers (BTBs) for each process or thread, ensuring predictions do not affect other threads' speculative execution.

5 Discussion

The absence of Spectre Branch Target Injection (BTI) variant attacks in TrustZone architecture could be attributed to the facilitated implementation of Branch Target Buffer (BTB) logic. As the BTB operates by predicting the target based on the virtual address of the program counter, the inclusion of an additional Non-Secure (NS) Bit in this prediction process can be implemented with relative ease. Subsequently, by implementing a mechanism to flush the BTB upon detecting any alteration in the NS Bit, mitigation of BTI attacks can be achieved. Therefore, theoretically possi-

ble to execute a Spectre attack within TrustZone, the BTB logic outlined effectively prevents such attempts, indicating that Spectre Branch Target Injection attacks are only feasible under specific conditions.

While measures are taken to mitigate Voltjockey by restricting privilege access and integrity checks, the voltage injection class of attacks remains a persistent threat [18] in TEE environments, necessitating continued research.

Although techniques such as prefetching can make cache timing attacks more challenging to execute, fully mitigating them without compromising performance remains an unresolved challenge [4], given that both trusted and untrusted applications utilize the same cache.

Meltdown in SGX is prevented by abort page semantics, but the absence of L1 cache flushing during page-in and page-out operations or not disabling hyperthreading leaves a Foreshadow [19] vulnerability, technically rendering Meltdown still feasible under certain circumstances.

Table 1. Side Channel Analysis on TEEs and REE

Attack	SGX	TrustZone	REE
Cache Timing	●	●	●
Voltage Injection	●	●	●
Row Hammer	●	●	●
Spectre	●	◑	●
Meltdown	◑	○	●

●: Attack possible. ◑: Attack under conditions. ○: Attack not possible.

6 Conclusion

This paper has examined representative classes of side-channel attacks on TEEs.The landscape of side-channel attacks and Trusted Execution Environments (TEEs) continues to evolve rapidly. As new optimization strategies are introduced, they often bring about novel classes of attacks, necessitating ongoing research and analysis. Understanding the strengths and weaknesses of different TEE environments is crucial in the face of these evolving threats. The comprehensive analysis allows us to discern the efficacy of security measures implemented within each TEE, providing insights into potential vulnerabilities and guiding the development of trusted applications based on their specific security requirements.

References

[1] Pierre Carru. Attack TrustZone with Rowhammer. Available Online: https://grehack.fr/2017/program (accessed on 05 April 2024).

[2] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1416–1432, 2020.

[3] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H. Lai. SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution. In *2019 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 142–157, 2019.

[4] Yun Chen, Ali Hajiabadi, Lingfeng Pei, and Trevor E. Carlson. New Cross-Core Cache-Agnostic and Prefetcher-based Side-Channels and Covert-Channels, 2023.

[5] Lukas Gerlach, Fabian Thomas, Robert Pietsch, and Michael Schwarz. A Rowhammer Reproduction Study Using the Blacksmith Fuzzer. In Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis, editors, *Computer Security – ESORICS 2023*, pages 62–79, Cham, 2024. Springer Nature Switzerland.

[6] Shay Gueron. Memory Encryption for General-Purpose Processors. *IEEE Security and Privacy*, 14(6):54–62, nov 2016. https://doi.org/10.1109/MSP.2016.124.

[7] Matthew Hoekstra, Reshma Lal, Pradeep Pappachan, Vinay Phegade, and Juan Del Cuvillo. Using innovative instructions to create trustworthy software solutions. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, New York, NY, USA, 2013. Association for Computing Machinery. https://doi.org/10.1145/2487726.2488370.

[8] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, SysTEX'17, New York, NY, USA, 2017. Association for Computing Machinery. https://doi.org/10.1145/3152701.3152709.

[9] Marc Joye and Michael Tunstall, editors. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012. https://doi.org/10.1007/978-3-642-29656-7.

[10] Paul Leignac, Olivier Potin, Jean-Baptiste Rigaud, Jean-Max Dutertre, and Simon Pontié. Comparison of side-channel leakage on Rich and Trusted Execution Environments. In *Proceedings of the Sixth Workshop on Cryptography and Security in Computing Systems*, CS2 '19, page 19–22, New York, NY, USA, 2019. Association for Computing Machinery. https://doi.org/10.1145/3304080.3304084.

[11] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval

Yarom, Mike Hamburg, and Raoul Strackx. Meltdown: reading kernel memory from user space. *Commun. ACM*, 63(6):46–56, may 2020. https://doi.org/10.1145/3357033.

[12] ARM Ltd. Learn the Architecture Exception Model. Available Online: https://developer.arm.com/architectures/learn-thearchitecture/exception-model/privilege-and-exception-levels (accessed on 05 April 2024).

[13] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. CacheZoom: How SGX Amplifies the Power of Cache Attacks. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 69–90, Cham, 2017. Springer International Publishing.

[14] Sandro Pinto and Nuno Santos. Demystifying Arm TrustZone: A Comprehensive Survey. *ACM Comput. Surv.*, 51(6), jan 2019. https://doi.org/10.1145/3291047.

[15] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6, 2019.

[16] Pengfei Qui, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. VoltJockey: Abusing the Processor Voltage to Break Arm TrustZone. *GetMobile: Mobile Comp. and Comm.*, 24(2):30–33, sep 2020. https://doi.org/10.1145/3427384.3427394.

[17] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted Execution Environment: What It is, and What It is Not. In *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01*, TRUSTCOM '15, page 57–64, USA, 2015. IEEE Computer Society. https://doi.org/10.1109/Trustcom.2015.357.

[18] Xhani Marvin Saß, Richard Mitev, and Ahmad-Reza Sadeghi. Oops..! i glitched it again! how to multi-glitch the glitching-protections on ARM TrustZone-M. USA, 2023. USENIX Association.

[19] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: extracting the keys to the intel SGX kingdom with transient out-of-order execution. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, page 991–1008, USA, 2018. USENIX Association.

[20] Daniel Weber, Fabian Thomas, Lukas Gerlach, Ruiyi Zhang, and Michael Schwarz. Reviving Meltdown 3a. In *Computer Security – ESORICS 2023: 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25–29, 2023, Proceedings, Part III*, page 80–99, Berlin, Heidelberg, 2024. Springer-Verlag. https://doi.org/10.1007/978-3-031-51479-1_5.

[21] Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y. Thomas Hou. TruSense: Information Leakage from TrustZone. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, page 1097–1105. IEEE Press, 2018. https://doi.org/10.1109/INFOCOM.2018.8486293.

Different approaches of inspecting encrypted packages in a content delivery network

Tino Korpelainen
tino.korpelainen@aalto.fi

Tutor: Jose Luis Martin Navarro

Abstract

Content delivery networks (CDNs) are integral to modern web traffic, serving the majority of all requests on the Internet. As the majority of internet traffic shifts to HTTPS, CDNs face challenges in inspecting encrypted packets for mitigating attacks, particularly denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. This paper examines various methods for inspecting encrypted traffic within CDNs, focusing on their effectiveness, false positive rates, privacy preservation, and performance.

Three inspection approaches are analyzed: Man-in-the-middle, knowledge-based, and machine learning-based. The Man-in-the-middle approach involves decrypting, inspecting, and re-encrypting traffic, which can achieve high detection and false positive rates but raises legal, security, and ethical concerns. The knowledge-based approach relies on domain experts creating filters or rules based on patterns in malicious traffic, which can be effective and privacy-preserving but requires regular updates and expert input. Machine learning-based approaches involve using machine learning algorithms to identify patterns in encrypted traffic based on various features, which can achieve high success rates in categorizing legitimate and malicious traffic but face challenges in dataset selection.

In the end, a good solution will be a combination of different approaches depending on their tested effectivity in the same environment. The solution

needs to balance the need for effectiveness with security, privacy, and performance concerns.

KEYWORDS: web application firewall, deep packet inspection, CDN, CDN attack, encryption

1 Introduction

Content delivery networks (CDNs) serve an important role in modern web traffic. They are the middleman to a large percentage of internet requests. According to Cisco Visual Networking Index: Forecast and Trends 2017-2022, the amount of internet traffic that flowed through CDNs was 56% in 2018 with an expected rise to 72% by the year 2022 [2].

The core idea of a CDN is to cache content as close to the user as possible to reduce the amount of latency to the user and reduce the load on the origin server [9]. CDNs also provide many different security capabilities, the largest of them being the mitigation of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks [1] [9].

DoS and DDoS attacks consist of an attacker sending large amounts of traffic to one host in an attempt to shut down a host. A distributed denial-of-service attack differs from a denial-of-service attack by using multiple different hosts to send illegitimate traffic. One of the ways that CDNS can mitigate DDoS attacks is by deploying web application firewalls (WAFs) [4]. These firewalls can inspect end-user requests and responses in a process called deep packet inspection (DPI) to detect illegitimate traffic [3].

But with most of the internet traffic being encrypted and sent over HTTPS [6], WAFs can no longer perform deep packet inspections on request payloads. This paper aims to give an overview of different ways of inspecting encrypted traffic to mitigate DDoS/DoS attacks on the part of a web application firewall. This paper also aims to analyze the different approaches and their strengths and weaknesses.

2 Overview of the problem

A content delivery network consists of 4 main components: end users, origin servers, edge servers, and request routing. End users are the consumers end-usernt. Origin servers are the original hosts of the content.

Edge servers sit between the origin servers and end users at points of presence (PoPs). Request routing is the component of a CDN that distributes end user requests to different edge servers. The main goal of request routing is to choose edge servers that will decrease latency for end users and evenly distribute the load [1].

All end user requests go to the edge server which will then either serve the content if it has it or it will fetch it from the origin server. Due to this edge servers are an ideal place to add security features. Thanks to request routing edge servers are naturally great at absorbing DDoS attacks but additional security can also be deployed such as the previously mentioned WAFs. Before HTTPS became popular WAFs could do simple inspections of the request payload to detect malicious actors. One example of this would be the User-Agent header; a WAF can detect simple DDoS attacks where the headers are not well-known web browsers. Now the amount of information that can be extracted from requests has been greatly reduced and other ways of detecting malicious requests are needed.

3 Different inspection approaches

There exist multiple different ways of inspecting encrypted traffic. This paper analyzes 3 different approaches: the Man-in-the-middle approach, the knowledge-based approach, and the machine learning-based approach. The specifics of these different approaches will be explained further but they all have similar requirements that they want to fulfill.

The main thing that they need to achieve is to be effective, meaning that they have some non-negligent detection rate. The other main thing they need is low a false positive rate, meaning that regular end users are not treated as malicious actors by the CDN. Other things that benefit the approach are privacy-preservation and performance. We want to preserve the security and privacy that HTTPS offers. The approach also needs to be performant. CDNs need to handle large amounts of requests in a short time, so inspections can't take a large amount of time. The simplest way of solving this problem is to negate it entirely. This can be done by decrypting the traffic at the CDN.

3.1 Man-in-the-middle approach

One way of inspecting encrypted data is by decrypting it at the CDN, doing the inspection, and then re-encrypting it before sending it to the final destination. This is called the Man-in-the-Middle approach and it can have many different problems, including legal, ethical, and security problems.

Interception of encrypted packages may increase the legal exposure of a company using it according to Jarmoc [8]. This is because one is usually inspecting communication that is expected to be private between two parties. This can be especially true when inspecting packages inside a company network as some countries or unions may have regulations that restrict the eavesdropping of employee communication. Even if the company is legally allowed to inspect their employee's packages it can still be unethical if employees are not cognisant of the inspection. An employee might, for example, see the green lock icon in their browser and assume that it is end-to-end encrypted like normally it would be. In this case, even though they might have signed an employment contract that allows for the inspection of their encrypted packages their browsers don't make that transparent to them.

Dormann identifies many different security issues with the man-in-the-middle approach [5]. One of the main problems is that the user cannot verify the encryption and correct usage of TLS/SSL between the CDN and their final destination. Dormann also analyzed different software that performs the MiTM and found several different mistakes that they make in their process, including not correctly validating upstream certificate validity, not correctly communicating back to the client all certificate information, and different warnings that may have propped up. Some applications even send the client's request to the server before communicating a warning about a certificate error, potentially giving an attacker sensitive data [5].

The man-in-the-middle approach solves multiple of the previously set requirements. It achieves the same detection and false positive rate as regular DPI solutions for non-encrypted traffic. This approach also adds a slight delay to all requests since it needs to perform an additional step of decryption and encryption. The biggest requirement that it fails is privacy-preservation.

3.2 Knowledge-based approach

Knowledge-based approach requires that a domain expert analyzes patterns in malicious traffic, creates filters or rules based on those patterns and then those are used by the CDN to categorize incoming traffic. Filters and rules about the specific domain are called the knowledge base As an example of this Husák, Martin et al. [7] used only the SSL/TLS cipher suite lists that were offered in the SSL/TLS handshake. By mapping cipher suite lists to their HTTP User-Agent values they were able to create a dictionary that can be used on unknown traffic to identify client types. They were able to achieve a 95.4% detection rate of different client types inside of a campus network.

However, there are multiple problems with the knowledge-based approach. First, you need to have an expert in the field that finds and analyzes the differences. Second, the knowledge base needs to be updated regularly, for example in Husák, Martin et al. [7] a popular browser could update their offered cipher suites and the mapping would need to be redone.

The knowledge-based approach can also have multiple upsides. If one can find rules with a high effectiveness, they can often be easily calculated. The knowledge-based approach can also be privacy-preserving when only using data from non-encrypted parts of the packet.

3.3 Machine learning-based approach

One of the most important parts of machine learning is choosing what features to use. Shen, Meng et al. [11] divide features in analyzing encrypted traffic into three different categories packet-based features, statistical features, and raw traffic representation. Packet-based features include f, for example, source and destination IP address, source and destination port, and Time-To-Live of the IP header. Statistical features are calculated from a group of packets. For example an average TCP header size over some amount of packages. Raw traffic presentation means that we don't extract any features manually but instead present the raw traffic in some form to a deep learning algorithm. These forms can be sequences, graphs, or images [11].

One of the used machine learning algorithms for identifying network traffic is called k-means clustering. K-means clustering aims to partition data points into defined clusters. In this approach, a certain amount of

clusters is predefined and the algorithm tries to fit the data points into that amount of clusters. If the algorithm finds appropriate clusters then new data can be fit into a specific cluster. In the domain of DDoS prevention, an ideal outcome would be two clusters with one being legitimate traffic and the other being malicious traffic. One example of using a machine learning algorithm k-means clustering is a paper by Qin et al.[10]. In the paper, they used 11 features consisting of a session consisting of encrypted packets. The features included such fields as source IP address, destination IP address, number of packets of different sizes, and the total duration of the flow. Using just these features they were able to achieve a successful categorization rate of over 95%, when the used dataset size exceeded 4000 data points.

A big problem with machine learning approaches is the selection of the dataset on which to train the model. The dataset must be representative of realistic traffic. This means that it must encompass a variety of different web drivers such as Firefox, Chrome, and curl, across different operating systems including Android, IOS, Windows, and Linux, as well as having different modes of internet access such as 2G, 3G, 4G, Wi-Fi and Ethernet. The creation of a useful and representative dataset can be difficult.

The best way to create a comprehensive dataset would be to monitor real-life traffic, which includes legitimate and malicious traffic. The problem with this approach is legal and ethical. Monitoring and capturing public internet traffic by individuals is illegal in most jurisdictions without explicit permission from a user. Large corporations such as internet service providers or CDNs usually have already expansive consent from users to monitor and capture their traffic without limits [12]. But large corporations have an incentive to keep the datasets to themselves, as it is a valuable asset to have. This results in a landscape where solutions are created on proprietary datasets that cannot be widely used or validated.

4 Conclusion

To fulfill their roles CDNs WAFs need to be able to inspect encrypted traffic in some form. They need to achieve a high detection rate, low false positive rate, performance and privacy-preservation. This paper brought forth 3 different approaches including man-in-the-middle approach, knowledge-based approach, and machine learning-based approach. The best practi-

cal solution will be a combination of these different approaches tailored to their environment. The solution needs to balance the need for effectivity with security, privacy and perfor- mance concerns as each of these approaches has their own strengths and weaknesses.

Man-in-the-middle approach involves decrypting the encrypted data at the CDN before inspection can occurs. This approach can achieve as high detection and false positive rates as regular WAFs on unencrypted traffic. But what suffers is end user security and privacy. Also ethical concerns can be raised as not all end users might be aware of the decryption of their data.

Knowledge-based approach relies on domain experts analyzing patterns in malicious traffic and creating filters or rules based on those patterns. It can be effective and privacy-preserving when using data from non-encrypted parts of the packet. However, it requires regular updates to the knowledge base and the expertise of a domain expert.

Machine learning-based approach involves using machine learning algorithms to identify patterns in encrypted traffic based on various features. It can achieve high success rates in categorizing legitimate and malicious traffic. However, the selection of a representative dataset for training the model can be challenging due to legal and ethical considerations.

The research and development of different approaches is highly dependant of representative datasets that encompass the complexity of the Internet. Achieving this kind of a dataset is extremely hard due to legal concerns and because companies don't want to give up their valuable assets for free. Due to this, it is hard to evaluate and compare different approaches. Especially in the machine learning approach the model can be fitted to one dataset with a high efficacy but when generalized to the Internet or another dataset its efficacy might not be enough or its false positive rate becomes unbearable. In the future of encrypted packet inspection research, there should be a concerted effort to create unified representative datasets or for companies to give researchers outside of their companies more access to their datasets.

References

[1] Akamai. What Is a CDN (Content Delivery Network)?, 2024. https://www.akamai.com/glossary/what-is-a-cdn.

[2] Thomas Barnett, Shruti Jain, Usha Andra, and Taru Khurana. Cisco visual networking index (vni) complete forecast update, 2017–2022. *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pages 1–30, 2018.

[3] Cloudflare. What is a next-generation firewall (NGFW)?, 2024. https://www.cloudflare.com/learning/security/what-is-next-generation-firewall-ngfw/.

[4] Cloudflare. What is a Web Application Firewall (WAF)?, 2024. https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/.

[5] William Dormann. The risks of ssl inspection. Carnegie Mellon University, Software Engineering Institute's Insights (blog), Mar 2015. Accessed: 2024-Mar-12.

[6] Google. Transparency report, 2024. Accessed: 2024-Mar-12. https://transparencyreport.google.com/https/overview.

[7] Martin Husák, Milan Cermák, Tomás Jirsík, and Pavel eleda. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016:1–14, 2016.

[8] Jeff Jarmoc. Ssl/tls interception proxies and transitive trust. In *Black Hat Europe*, 2012.

[9] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, aug 2010.

[10] Xi Qin, Tongge Xu, and Chao Wang. Ddos attack detection using flow entropy and clustering technique. In *2015 11th International Conference on Computational Intelligence and Security (CIS)*, pages 412–415, 2015.

[11] Meng Shen, Ke Ye, Xingtong Liu, Liehuang Zhu, Jiawen Kang, Shui Yu, Qi Li, and Ke Xu. Machine learning-powered encrypted network traffic analysis: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 25(1):791–824, 2023.

[12] Douglas Sicker, Paul Ohm, and Dirk Grunwald. Legal issues surrounding monitoring during network research. pages 141–148, 10 2007.

Overview of Utility-First CSS

Tommi Pakarinen
tommi.pakarinen@aalto.fi

Tutor: Juho Vepsäläinen

Abstract

This paper provides an overview of Utility-First CSS and its impact on web development practices. We analyze and compare the technical aspects of Semantic CSS and Utility-First CSS. The paper examines popular implementations, such as Tailwind CSS, Tachyons, and Open Props, discussing their features and impact on web development. Our findings contribute to a deeper understanding of the evolving landscape of CSS methodologies.

KEYWORDS: CSS, Web, Utility-First, Tailwind

1 Introduction

Traditionally, a website is created using multiple technologies with different roles. Hypertext Markup Language (HTML) is the standard language for creating the skeleton of a site. This website is then styled using Cascading Style Sheets (CSS) to give the page more aesthetic elements. However, there is no consensus on how style sheets should be organized for easier scalability and less redundancy.

The most common method of organizing style sheets is to use generic classes, such as a ".button" class for an HTML element that acts as a clickable button. However, this style of CSS has drawbacks that can make

stylesheets harder to iterate and modify. This paper examines Utility-First CSS, which attempts to address these problems by using utility classes instead of creating unique classes for every kind of element.

The remaining sections are organized as follows: Section 2 presents technical details about the CSS language and explains the differences between the traditional and Utility-First paradigms. Section 3 lists the most popular implementations of Utility-First CSS. Section 4 discusses the impact of Utility-First CSS on web development. Finally, Section 5 contains concluding remarks.

2 CSS Paradigms

Cascading Style Sheets are based on style files consisting of blocks of rules. These rule blocks consist of two components: the selector and the declaration block. The selector specifies which elements in the HTML should be selected, and the declaration block is essentially a list of key-value pairs dictating how the properties of the HTML element should be modified. These blocks cascade, meaning that an element can be the target of multiple declaration blocks, and it will apply all the rules as long as there are no conflicts. These features are the basis of CSS and have been part of the language since 1996 in the first W3C recommendation [19].

CSS file

```css
/*This is the selector*/
.button {
    /*Start of the declaration block*/
    background-color: blue;
    margin: 4px;
    /*End of the declaration block*/
}
```

HTML file

```html
<!-- The styles are applied to this html element -->
<button class="button"></button>
```

Figure 1. Example of a CSS and HTML file.

These declaration blocks, containing multiple rules, can be reused multiple times using classes. Classes are versatile and arguably the cornerstone of CSS. Due to this versatility, it is not clear what a class should represent. In modern web development, classes commonly represent a generic idea: a button, a sidebar, or even as small as an icon. This in-

tuitive approach can be supplemented with utility classes such as white text class or background color class. This idea of designing classes for an abstract idea is commonly referred to as Semantic CSS [20]. On the other hand, a recent idea known as Utility-First CSS has been on the rise. In this paradigm, all the classes are utility classes, and HTML elements are modified by giving them a list of utility classes dictating their appearance [12].

The following sections list and compare the main differences between current mainstream CSS frameworks, such as Bootstrap, and emerging utility-First CSS frameworks, such as Tailwind CSS.

2.1 Semantic CSS

Semantic CSS is the traditional strategy for writing CSS stylesheets. This strategy is officially endorsed by W3C in the working draft for HTML5 [18]. In this paradigm, each class is semantically tied to an abstract concept. For instance, a class such as ".ord-list" might signify an ordered list.

Many Semantic CSS frameworks operate similarly but offer different styling for components. These frameworks encourage developers to utilize the provided classes. For example, frameworks like Bootstrap [1] and MaterialUI [2] offer classes defining elements like the "select field", with the styling contained within these classes. Utilizing these frameworks involves adding HTML elements and applying the relevant classes to achieve the desired appearance, thus facilitating ease of use and rapid iteration. This model, characterized by pre-styled components, is referred to as the Semantic CSS in this paper.

This kind of organizational model is easy to expand on and reuse, assuming that there is no need to alter any pre-existing classes. However, if this is the case, the CSS might become hard to modify and debug. Several methods exist for altering existing classes, but many of these methods are considered bad practice. These include using "!important", inline styles, or increasing specificity to force precedence, which may result in convoluted stylesheets that are difficult to read and debug. Consequently, modifying such changes may necessitate the continued addition of priority-increasing keywords.

Another issue with Semantic CSS is redundant declarations, where classes may contain repeated declarations, leading to bloated stylesheets. A solution to this redundancy is to extract repeated declarations into a utility class that can be applied to classes without redundancy. However,

modifying the style of an element with a utility class containing multiple rules may require the use of priority-increasing strategies. Utility-First CSS is based on this idea of utility classes.

2.2 Utility-First CSS

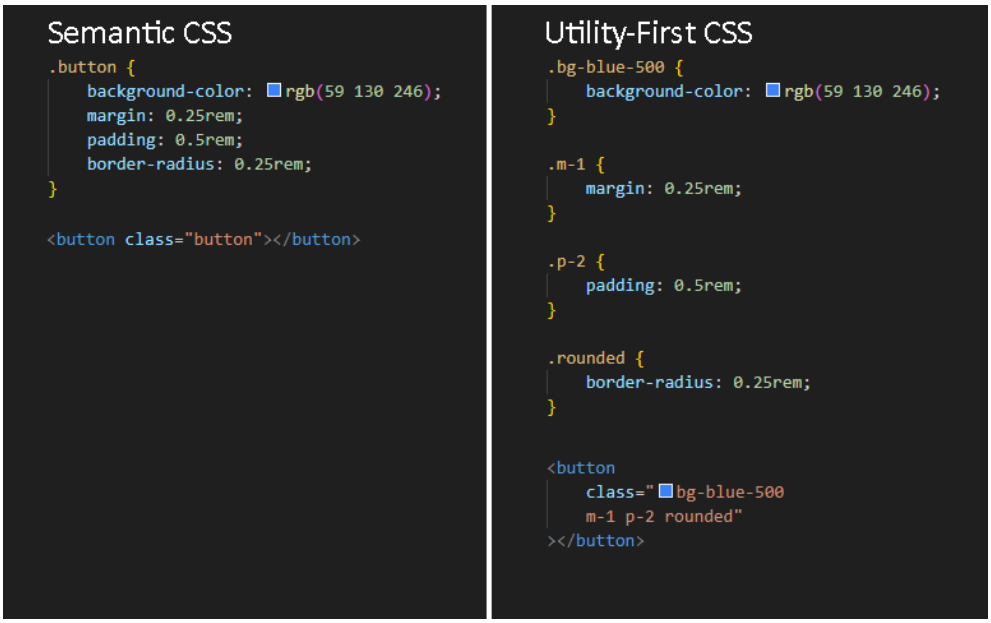


Figure 2. Comparison between Semantic CSS (left) and Utility-First CSS (right)

In Semantic CSS, developers need to define and name semantic classes that correspond to abstract concepts, such as a button. Utility-First CSS turns this concept around and aims to streamline styling by eliminating the need to create semantic classes. This is achieved by introducing utility classes consisting of only a few CSS rules declarations at most [12]. For instance, there could be a utility class named ".bg-blue" whose sole purpose is to apply the CSS declaration "background-color: blue". Therefore, styling an element involves combining these utility classes to achieve the desired appearance. The difference between these two CSS paradigms is illustrated in Figure 2.

Iterating on design with Utility-First CSS is fast since there is no need to design CSS classes, resulting in a codebase that is easier to maintain [21]. However, these lower-level utility classes may lead to lengthy and repetitive class lists that are challenging to parse by a human [15]. Some frameworks, like Tailwind CSS, address these issues by introducing methods to bundle utility classes into semantic classes [13]. Additionally, using component-based frameworks such as React provides a solution for reusing components without the necessity to write redundant utility

classes.

3 Implementations of Utility-First CSS

Semantic CSS and Utility-First CSS are paradigms that differ significantly, but they are not mutually exclusive. A Semantic CSS framework usually includes utility classes for more abstract features, such as breakpoints for responsive design. Bootstrap 5 introduces a new feature named the Utility API, enabling developers to generate utility classes for the project [5].

Currently, the most popular Utility-First CSS libraries are Tailwind CSS [7], Open Props [3], and Tachyons [4] according to the State of CSS 2023 survey [17]. The rise of Utility-First CSS is a recent trend, with Tailwind CSS particularly standing out due to its high satisfaction ratio among developers [17]. This section provides examples of CSS frameworks that utilize the Utility-First Fundamentals.

3.1 Tailwind CSS

Tailwind CSS [7] is currently the largest Utility-First CSS framework and the second most used CSS framework behind Bootstrap [17]. It is utilized by companies such as Netflix [10]. The framework offers a vast library of utility CSS classes that can be utilized out of the box. Additionally, it can dynamically generate new utility classes for different modifiers, such as "dark" or "hover". Despite its extensive library, Tailwind CSS optimizes performance by bundling classes into a single small build stylesheet containing only utility classes used in the project [8]. Figure 3 shows an example of Tailwind CSS.

```
<div class="bg-gray-50 p-8">
  <a class="rounded-full py-2
    px-4 text-sm font-semibold
    bg-slate-900 text-white
    hover:bg-slate-700
    hover:text-slate-100"
    href="/link">Button</a>
</div>
```

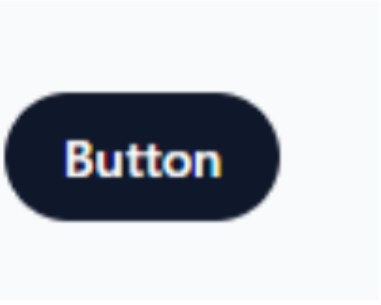


Figure 3. Example of Tailwind CSS. Left is CSS and right is the render. Each class applies a small set of CSS rules. For example the class ".py-2" is equal to the following: "padding-top: 0.5rem; padding-bottom: 0.5rem;"

Tailwind CSS also supports Semantic CSS through the use of the "@ap-

ply" directive. This directive enables developers to apply multiple classes under another class label, essentially creating a reusable shorthand for a group of utility classes. For instance, one can semantically define a class ".button" that applies the following classes: ".font-medium .rounded-lg" [13].

In modern web development, web pages are accessed on various devices with different aspect ratios, such as phones and tablets. Responsive web design is a crucial design approach that ensures smooth viewing across different devices [16]. Tailwind CSS facilitates responsive web design through breakpoint prefixes, which can be added to any utility class to create conditional classes. These classes are enabled only at specific screen resolutions, allowing developers to define resolution breakpoints according to their needs [11].

Tailwind CSS offers other built-in features as well. Modifier utility classes can emulate native CSS pseudo-classes such as ":focus" and ":hover". Additionally, dark mode classes are supported with the "dark" modifier. Similar to breakpoint prefixes, these modifiers can be attached to any utility class, and the classes are efficiently generated and bundled by the Tailwind CSS bundler [9].

3.2 Tachyons

Tachyons [4] is another Utility-First CSS library that follows a similar workflow to Tailwind CSS. Although it has only a tenth of the usage rate that Tailwind CSS has, it stands as the second most popular Utility-First CSS framework [17].

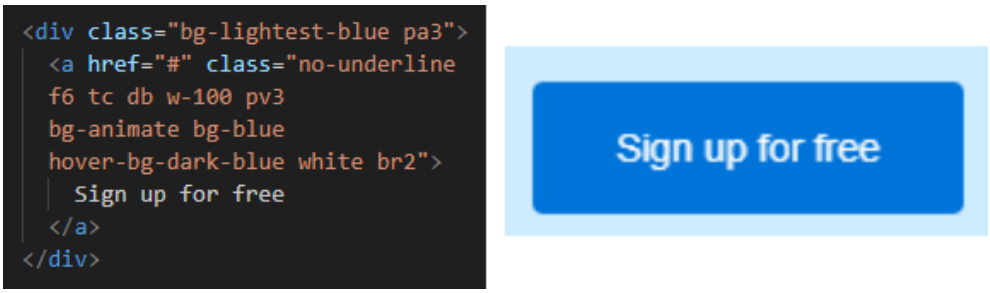


Figure 4. Example of Tachyons. Left is CSS and right is the render. The naming scheme is different but the workflow is similar to Tailwind CSS. For example the class ".db" is equal to the following: "display: block;"

Tachyons is an extensive library of utility classes that are readily available for use. In comparison to Tailwind CSS, Tachyons is an extremely lightweight library that occupies considerably less space [14].

However, the framework is a pure Utility-First CSS library and therefore does not offer any additional features beyond a rich collection of utility functions. Unlike Tailwind CSS, it is not possible to group utility classes under a single label using Tachyons [4].

3.3 Open Props

Open Props [3] is a recent lightweight CSS framework that incorporates many Utility-First CSS fundamentals and applies them to Semantic CSS. It shares a similar usage rate to Tachyons, being the 10th most used CSS framework according to the State of CSS 2023 survey [17].

Open Props works on the built-in CSS variable feature, that allows developers to set arbitrary values to variables. This CSS framework includes an extensive library of utility variables, such as fixed text sizes or colors, that can be used in CSS declarations. Unlike Tailwind CSS, however, it is not possible to directly apply these utility variables to an HTML element. Instead, developers must create semantic classes that serve as a collection of these utility variables. The framework operates similarly to Tailwind CSS's "@apply" directive. Additionally, Open Props supports responsive web design through media query props, enabling developers to set conditional styles [3].

Figure 5. Example of Open Props with CSS (right side), HTML (top left), and the render (bottom left). Semantic CSS classes are assigned to elements. These classes consist of variables provided by Open Props. For example "padding: var(–size-3)" evaluates to "padding: 1rem".

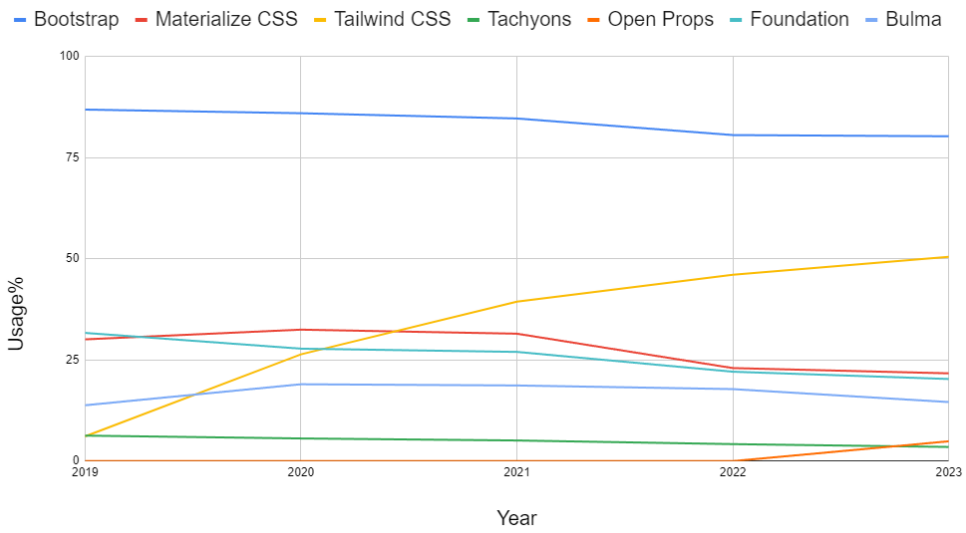


Figure 6. State of CSS 2023 survey [17].

4 Impact on Web Development

Utility-First CSS is a new paradigm that only started being adopted in the 2020s. According to the State of CSS 2023 survey [17] (Figure 6), Tailwind CSS initially had a modest usage rate of only 6.1% among web developers. However, its adoption has seen consistent growth, with Tailwind CSS reaching a substantial usage rate of 50.5% by 2023, positioning it as the second most utilized CSS framework after Bootstrap (80.3%). Currently, Tailwind CSS stands as the de facto Utility-first CSS framework.

Despite the rapid rise of the Utility-first paradigm, it remains relatively young. When examining the usage of CSS frameworks across the internet, it becomes evident that the market share of Utility-First CSS frameworks remains relatively small. W3Techs is an institution analyzing technologies used in websites through data scraping [6]. As of April 2024, Tailwind CSS only holds a market share of 0.6% among every website employing a CSS framework. In contrast, Bootstrap, having been in use for a longer duration, occupies a market share of 77.9% among all websites. Nonetheless, Tailwind CSS still ranks as the fifth most prevalent CSS framework on the internet. [6]

Utility-First CSS is still evolving, and its impact on web development is currently relatively small. However, the adoption rate of libraries such as Tailwind CSS has been increasing steadily year by year [17], which could potentially inspire more developers to switch to Utility-First CSS. Due to the recency of this paradigm, there is a clear lack of academic literature on Utility-First CSS and its impact. Nevertheless, Utility-First CSS

remains a relatively new paradigm compared to Semantic CSS frameworks like Bootstrap.

5 Conclusion

Utility-First CSS offers a departure from traditional Semantic CSS methodologies by prioritizing utility classes over semantic classes. Instead of defining unique classes for each semantic element, developers utilize utility classes that provide specific styling properties. This approach aims to streamline development processes, enhance maintainability, and reduce redundancy in stylesheets.

Among the popular implementations of Utility-First CSS frameworks, Tailwind CSS emerges as a prominent example. Tailwind CSS offers a comprehensive library of utility classes, allowing developers to rapidly iterate on designs and create responsive layouts. The framework's support for Semantic CSS through features like the "@apply" directive enables developers to combine utility classes into semantic classes for better organization and reusability.

While Utility-First CSS is still in its early stages compared to Semantic CSS frameworks like Bootstrap, its rapid adoption among developers signifies its potential to reshape the landscape of web development. As developers continue using Utility-First CSS, its impact on web development will become more pronounced. Many frameworks such as Bootstrap already include many features that follow Utility-first ideas.

References

[1] Bootstrap. https://getbootstrap.com/docs (Accessed 29.01.2024).

[2] MUI. https://mui.com (Accessed 29.01.2024).

[3] OpenProps. https://open-props.style (Accessed 04.04.2024).

[4] Tachyons. https://tachyons.io (Accessed 29.02.2024).

[5] Utility API. https://getbootstrap.com/docs/5.3/utilities/api (Accessed 03.04.2024).

[6] Usage statistics and market shares of CSS frameworks. April 2024. https://w3techs.com/technologies/overview/css_framework (Accessed 04.04.2024).

[7] A. Wathan. Tailwind CSS. https://tailwindcss.com (Accessed 29.02.2024).

[8] A. Wathan. TailwindCSS. https://tailwindcss.com (Accessed 03.04.2024).

[9] A. Wathan. Handling Hover, Focus, and Other States, 2020. https://tailwindcss.com/docs/hover-focus-and-other-states (Accessed 04.03.2024).

[10] A. Wathan. Optimizing for Production, 2020. https://tailwindcss.com/docs/optimizing-for-production (Accessed 04.03.2024).

[11] A. Wathan. Responsive Design, 2020. https://tailwindcss.com/docs/responsive-design (Accessed 04.03.2024).

[12] A. Wathan. Utility-First Fundamentals, 2020. https://tailwindcss.com/docs/utility-first (Accessed 28.01.2024).

[13] A. Wathan. Utility-First Fundamentals, 2020. https://tailwindcss.com/docs/utility-first (Accessed 29.02.2024).

[14] Anjolaoluwa Adebayo-Oyetoro. Tailwind CSS vs. Tachyons. May 2020. https://blog.logrocket.com/tailwindcss-vs-tachyons/ (Accessed 04.04.2024).

[15] Rehan Alam. Utility-first CSS framework? - Tailwind CSS, 2020. https://xenox.dev/tailwind-css-utility-first-css-framework (Accessed 29.02.2024).

[16] Fernando Almeida and José Monteiro. The role of responsive design in web development. *Webology*, 14(2), 2017.

[17] Sacha Greif. State of CSS 2023. https://2023.stateofcss.com/en-US/css-frameworks (Accessed 29.02.2024).

[18] Ian Hickson. HTML5, A vocabulary and associated APIs for HTML and XHTML. April 2011. https://www.w3.org/TR/2011/WD-html5-20110405 (Accessed 02.04.2024).

[19] Håkon Wium Lie and Bert Bos. Cascading Style Sheets, level 1. December 1996. https://www.w3.org/TR/REC-CSS1-961217 (Accessed 28.01.2024).

[20] Tero Piirainen. NUE, 2023. https://nuejs.org/blog/tailwind-vs-semantic-css (Accessed 29.02.2024).

[21] Inc. ThoughtWorks. Technology radar vol. 22. 2020. https://assets.thoughtworks.com/assets/technology-radar-vol-22-en.pdf (Accessed 29.02.2024).

State orchestration in web with finite-state machines

Tuomas Salminen
tuomas.t.salminen@aalto.fi

Tutor: Juho Vepsäläinen

Abstract

State management has become increasingly important in web development, as demand for feature-rich web applications complicates development. Finite-state machines have been widely used in various fields of software development to model the various states the system can have, as well as the transitions between these states. This paper reviews the use of finite-state machines in web application state management. This approach is also referred to as state orchestration. Two major libraries for state orchestration, XState and Robot, are examined in this paper, as well as the benefits and drawbacks of such approaches. These libraries are based on the statecharts model, which is an extension of the finite-state machine model.

This paper argues that by using state machines, and statecharts in particular, the system can be made more robust, with fewer possibilities for ending up in unwanted states and thus causing errors. However, the learning curve is steep, especially because state machines or statecharts are rarely used in web development. Therefore, using libraries such as XState and Robot is beneficial in systems which can be effectively modeled as state machines, and which benefit from robust state management.

KEYWORDS: finite-state machine, statecharts, state management, state orchestration

1 Introduction

Businesses are increasingly deploying their services as web applications, that are expected to be feature-rich and reactive. Therefore, web development has become highly complicated. In particular, state management in web applications has become problematic, as the applications grow larger with high amounts of possible events and outcomes. Modern web applications often use a component framework, such as React, Vue, Angular, or Svelte. These frameworks allow developers to create web applications using components that can manage their own state, and the state of their child components by passing down the state to them. Large web applications usually use a state management library, such as Redux, MobX, or Zustand, to handle the application's global state.

These state management libraries act as a data store for information needed in multiple parts of the application. However, this approach leaves much to be desired when web applications have increasingly complex state transitions with multiple conditions for moving from one state to another. In the data store approach, the developer has to manually implement the logic for state transitions, for example, by using if statements. This can result in unwanted states and bugs in the code.

This paper reviews an alternative approach for state management in web applications with the use of finite-state machines (FSM). FSMs can enable the state management logic in the application to specify which transitions are allowed between the states. This approach can also be considered state orchestration instead of state management since FSMs control how the data can change in the application, instead of only storing it. Currently, XState is the most widely used solution for implementing state management via FSMs in web development.

This paper tries to answer the following questions: what is the current state of the art for state orchestration using FSMs in web development and what are the benefits and disadvantages of such approaches? XState is examined in particular, and a small demo of an XState state machine will be presented to understand the library better. Another similar library, Robot, will also be discussed in this paper.

The paper is organized as follows. Section 2 introduces FSMs, statecharts, and the actor model. Section 3 examines the XState and Robot state orchestration systems. Section 4 presents an implementation of a network request flow using XState. Section 5 discusses the benefits and

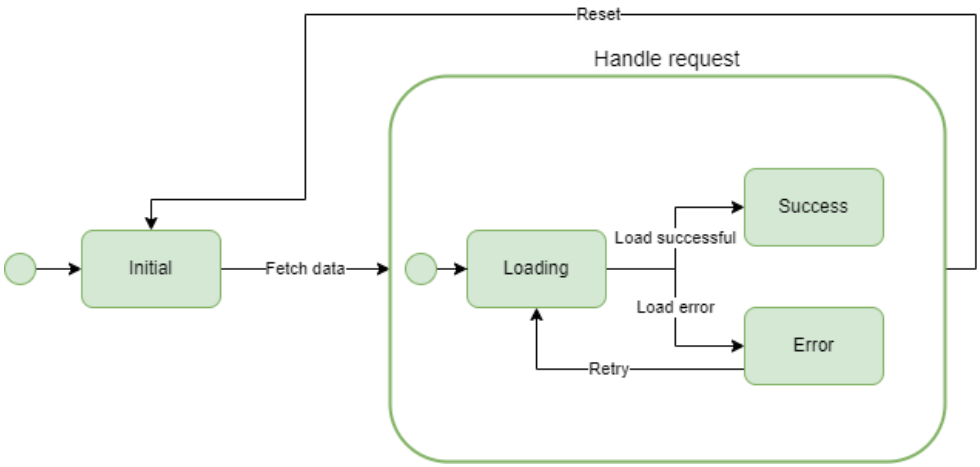


Figure 1. Statechart diagram for a network request flow.

drawbacks of the presented approaches to state management. Section 6 presents the conclusion.

2 Finite-state machines, statecharts, and the actor model

This section covers the FSM, statechart, and actor model. The modern FSM state orchestration solutions use statecharts in their design, which extend the conventional FSMs with powerful features, such as hierarchy, concurrency, and communication [10]. The newest version of XState allows the use of the actor model [5], in which components of the system are actors that will send and receive messages to and from other actors [12].

2.1 Finite-state machine

The FSM is a model of computation that has a finite amount of states, a start state, inputs, and transitions [6]. Transitions determine how to move from one state to another given some input [6]. FSMs can be specified using state diagrams, which also help to visualize the system. FSMs have been studied and used in various fields and applications, including game development [14, 8], space launch systems [11], and event reconstruction in digital investigations [9]. However, the basic FSM model is too simple to be used in complex systems effectively, as the state diagram quickly grows to an unmanageable size when the number of states and transitions increases [10].

2.2 Statechart

Harel [10] extends the FSM model by proposing the statechart model, which is a highly descriptive language for concisely expressing complex systems. Statecharts can model hierarchy, conditions, and concurrency between states. Due to hierarchy, the states can be nested inside other states, allowing parts of the system to be decoupled from each other. Conditions, also known as guards, allow for preconditions for transitions, meaning transitions can be blocked if some condition is not met. With concurrent states, the system can be split into multiple parts, with the states in these parts not affecting each other but instead working in parallel.

Figure 1 illustrates a simple network request flow with the statechart model. The handle request state is a parent state with three child states. The initial state does not have to know how the request-handling logic works, therefore it will only interact with the parent state. In a later section, I will present how this logic can be implemented using XState.

2.3 Actor model

The actor model uses an actor as the central component of a system. The actor has only one behavior, which is sending messages to other actors [12]. Hewitt et al. [12] present many benefits of this model, including privacy, synchronization, and intentions. Actors will work only for actors which have the authority to use them, actors can work concurrently, and actors can be evaluated based on whether they satisfy their intention, which can help with testing [12]. In XState, the actor model is used to represent each FSM as an actor.

3 Current state orchestration solutions

The two major solutions for FSM state orchestration in web applications are XState and Robot. Both use the statechart model but also have many differences. In this section, the technologies are reviewed and compared.

3.1 XState

XState is a JavaScript and TypeScript library for creating FSMs and using them for state orchestration in both frontend and backend systems.

XState uses the statechart and actor model to manage the state of increasingly complex web applications. It is framework-agnostic and can be used with vanilla JavaScript or TypeScript if wanted. As of writing this paper, it has 1.6 million weekly downloads [4]. Although the framework has been researched little, it has been effectively used in some research, for example by Chen [7], where a clinical guidance system is implemented to prevent medical errors. With the use of statecharts in XState, the guidance system was built to be lightweight, safety-prone, and configurable.

XState uses event-driven architecture together with the actor model. In XState, a FSM is an actor, which can receive and send events, as well as change its behavior when it receives an event [1]. The actors can also invoke or spawn other actors to do some designated workflow. The difference between invoking and spawning is that invoked actors represent a single state-based task that is stopped when the invoking state is exited, and spawned actors represent multiple entities and can be started and stopped at any time [1].

3.2 Robot

Similar to XState, Robot is used for state orchestration via FSMs in JavaScript and TypeScript applications. It also uses statecharts in its design. However, there are some key differences between them [2]. Robot improves on XState by intentionally keeping the bundle size of the library small at 1 kB, whereas XState bundle size is 13 kB. In Robot, the FSMs are composable, meaning they are constructed from smaller reusable parts, unlike in XState where the FSM is defined with one JavaScript options object. XState supports the statechart feature of parallel states for a single FSM while Robot does not. In Robot, this feature can still be implemented by defining separate machines. Robot does not support the actor model as XState does and it does not have a visualizer tool that draws diagrams of the defined FSMs as in XState.

4 XState example

Figure 2 shows how the network request diagram in figure 1 looks as an XState FSM. XState allows the use of statecharts features such as hierarchy and guards, as seen in the diagram with the handle request parent state providing hierarchy to the machine, and the if-else condition for

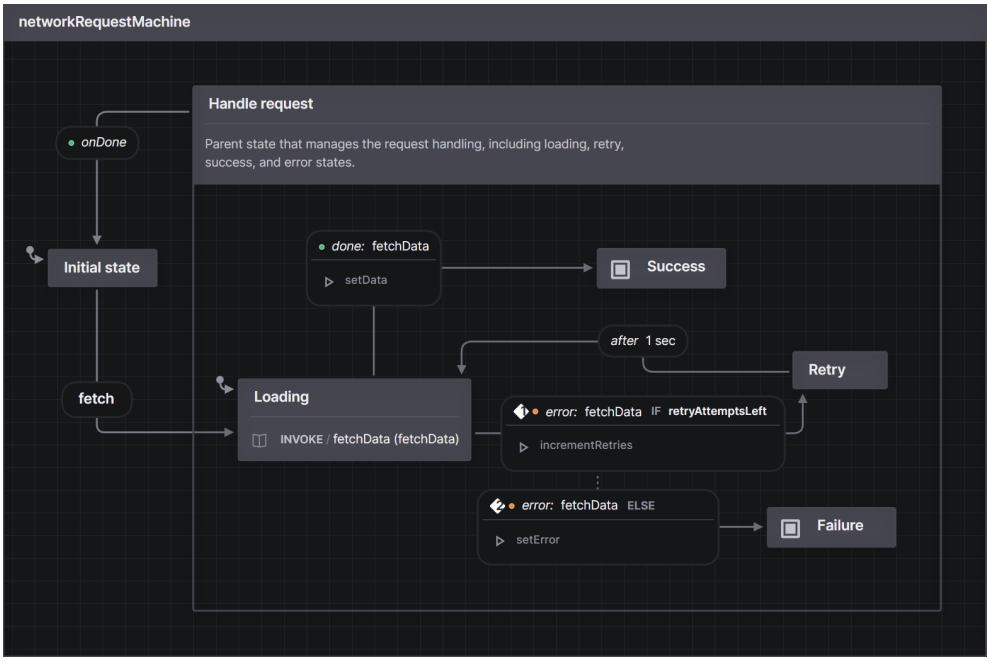


Figure 2. XState state machine diagram for a network request flow. Created with the XState visualizer.

transitioning to the retry or failure states guarding unwanted transitions from occuring. The success and failure states are the final states inside the handle request parent state and will transition back to the initial state via the onDone event.

The actor model in XState allows invoking actors to do a designated workflow. The loading state invokes a workflow that tries to load data from the server and will either complete it successfully or cause an error. The loading state can react to this result and transition to the correct state. Certain transitions also manipulate the state machine context, such as when the machine transitions from loading to success state. Here the setData function will be called, which sets the loaded data into the context. The context information can be used in the machine, or retrieved and used outside the machine.

The machine state will transition from the loading state to the retry state if there are retry attempts left. The function retryAttemptsLeft is defined in the code and will return true if the amount of retries is below five. At the same time, the amount of retries will be incremented by one each time the transition from loading to retry state occurs. A delayed transition will then take place from the retry state back to the loading state.

5 Discussion

The previous sections reviewed the current state orchestration solutions and gave background on the theory behind them. This section discusses the benefits and drawbacks of these approaches.

5.1 Learning curve

Finite-state machines, and statecharts in particular, offer an effective option to state orchestration in web applications, as discussed in [15]. However, the paper also points out that some prerequisite competence in FSMs is needed to benefit from this approach. This makes the learning curve for XState and Robot and similar solutions steep. Therefore, there should be careful consideration before such tools are used. Horrocks [13] states that statecharts are not widely used in user interface development because their benefits are not obvious. Currently, there are many tools available for frontend development, and often FSMs or statecharts are not needed. In the worst scenario, they complicate the development process more with little benefit.

To make the learning curve easier, XState has extensive and clear documentation and a visualizer tool, as well as AI tools to help with the FSM creation. Robot lacks the visualizer and AI tools, and the documentation is not as extensive, but the amount of features to learn is also fewer in Robot. In particular, the visualizer can be useful in understanding the logic of the FSM. The XState visualizer allows for stepping through the states one by one and exporting the FSM diagram as code. There is also an extension for Visual Studio Code that generates a diagram directly from the given code, which can improve the development experience.

5.2 Code robustness

Horrocks [13] found that bug count was lower when using statecharts. The bug count in statechart-based programs was only between 10 and 20 percent of the number of bugs in the programs built using other approaches. This can be due to FSMs and statecharts making it harder to have unwanted states in the program. Furthermore, modeling a system using statecharts can make testing easier when the possible events and outcomes can be observed from the state diagram. The premium version of XState studio also allows for automatic test case creation [3] based on

the provided code, making it simple to have high testing coverage for the state machines.

6 Conclusion

The objective of this paper was to research the current state of the art for state orchestration using FSMs in web development and analyze the benefits and disadvantages of such approaches. This paper examined the two major FSM state orchestration solutions, XState and Robot. Both allow for building robust statecharts-based software, but XState offers a wider range of features, such as the actor model. Compared to other, non-FSM state management solutions, they allow for more robust state handling, but can also make the code more complicated.

Future research on this topic would be valuable, given the rising demand for feature-rich, large web applications. This paper only reviewed the technologies superficially, therefore a more detailed study into them would be beneficial. Particularly, studying the use of statecharts-based state management tools in larger systems could yield useful results. Additionally, studying the developer experience of such tools could be valuable.

References

[1] Actors. https://stately.ai/docs/actors. Accessed: 2024-03-02.

[2] Comparison with xstate. https://thisrobot.life/guides/comparison-with-xstate.html. Accessed: 2024-03-02.

[3] Generate test paths. https://stately.ai/docs/generate-test-paths. Accessed: 2024-04-03.

[4] npm xstate. https://www.npmjs.com/package/xstate. Accessed: 2024-03-02.

[5] Xstate concepts. https://xstate.js.org/docs/about/concepts.html. Accessed: 2024-02-01.

[6] Paul E. Black. finite state machine. Technical report, National Institute of Standards and Technology, 2021. Accessed: 2024-03-02.

[7] Luting Chen. *Modern lightweight approach for design and implementation of workflow-based clinical guidance system*. PhD thesis, University of Illinois at Urbana-Champaign, 2021.

[8] Christopher Dragert, Jorg Kienzle, and Clark Verbrugge. Statechart-based ai in practice. *Proceedings of the AAAI Conference on Artificial Intelligence*

and Interactive Digital Entertainment, 8(1):136–141, Jun. 2021.

[9] Pavel Gladyshev and Ahmed Patel. Finite state machine approach to digital event reconstruction. *Digital Investigation*, 1(2):130–149, 2004.

[10] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[11] Joshua A Harris and Ann Patterson-Hine. State machine modeling of the space launch system solid rocket boosters. Technical report, 2013.

[12] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, page 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[13] Ian Horrocks. *Constructing the User Interface with Statecharts*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1999.

[14] Devang Jagdale. Finite state machine in game development. *algorithms*, 10(1), 2021.

[15] Thanh Nguyen. Statecharts for modern web application state management. 2020.

Power and energy monitoring for sustainable large-scale computing

Tyler Eck
tyler.eck@aalto.fi

Tutor: Vesa Hirvisalo

Abstract

The emergence of edge computing has presented numerous challenges in determining the environmental impact and long-term sustainability of centralized and decentralized computing systems. The environmental impact of decentralized computational systems, by nature, is difficult to ascertain. To this end, this paper aims to contribute to the existing literature by comprehensively reviewing the environmental considerations associated with edge computing. By examining the effect of a common energy monitoring technique, cooling optimization, across distributed computing nodes in a large-scale edge computing system, this paper seeks to determine its effectiveness in managing power consumption in decentralized computing architectures.

KEYWORDS: Edge Computing, Large Scale Systems, Cooling Optimization, Cloud Computing, Power Monitoring, Energy Management Systems, Sustainability

1 Introduction

In recent years, cloud computing services have become increasingly popular solutions to the rapid increase in demand for accessing computing

resources and data from any location at any time. Cloud computing data centers, while centralizing computational services, have come at a significant financial and environmental cost. With energy-intensive data centers accounting for 1% of worldwide electricity [2] and 2% of global greenhouse gas emissions [4], there is a significant need for data center operators to conceptualize and adopt strategies focused on managing their power and energy consumption to meet changing international and domestic standards.

To tackle the challenges of reducing the carbon footprint and operational costs of cloud computing data centers, industry leaders such as Google, Amazon, and Microsoft have employed advanced power monitoring techniques to increase energy efficiency. These intelligent approaches involve utilizing Power Usage Effectiveness (PUE) for optimizing energy management systems. While effective in enhancing energy efficiency inside the data centers, monitoring energy consumption has become more complicated as computing has expanded outside these centers with the emergence of edge computing. Edge computing extends the centralized data centers to form modern large-scale systems closer to the user. Although this distribution of resources has significant advantages, this extension introduces challenges in accurately measuring power and energy consumption due to its complex structure.

In both cloud and edge computing, effective power management dictates the sustainability and cost-effectiveness of a data center system. Of particular concern is the optimization of cooling mechanisms, which play a pivotal role in maintaining power efficiency within data centers. Traditional cooling approaches are inefficient and environmentally unsustainable on large scales, which pose substantial energy costs and environmental risks. Consequently, data center operators have found the need to seek alternative cooling strategies, including geographical positioning and manipulating the layout of equipment placement within their facilities. These innovative approaches offer the dual benefit of reducing energy consumption while enhancing cooling efficiency, thereby addressing the need for power management in modern computing environments. That said, the challenge remains in monitoring the efficacy of cooling optimization techniques and determining the most practical one.

This paper is to provide a comprehensive review of the power and energy monitoring techniques within data centers and analyze the effectiveness and challenges of cooling optimization in large-scale computing

systems. Following this introduction, the paper will begin with a brief discussion of background information related to the evolution of cloud computing into edge computing. Then, the third section will examine the strategies and techniques used to monitor power and energy usage, management, and sustainability inside data centers. The fourth section will then elaborate on the challenges in power and energy monitoring in edge computing systems. Finally, the fifth and sixth sections are the discussion and conclusion.

2 Background

This section will first discuss the evolution of cloud computing and data centers, then outline the emergence and importance of edge computing by illustrating how it complements the cloud to form large-scale systems. Finally, the concept of power and energy monitoring is outlined, focusing on the relationship between power monitoring and energy management.

2.1 The Evolution of Cloud Computing and Data Centers

Cloud computing provides access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal effort or service provider interaction [19]. Widely adopted cloud services and applications, like Amazon's EC2/S3 and Google's Google Mail and Google Drive, promote cloud computing as the most significant role in the IT sector [13]. The widespread adoption of cloud computing across multiple sectors has resulted in the construction of data centers to minimize latency and increase service availability. As of mid-2020, there were a total of 541 public hyper-scale data centers globally and, with a 45% increase in global cloud traffic, this number is predicted to double [21] and suggests a rise in market value from USD 559.2 billion in 2023 to USD 2297.37 billion by 2032 [20].

While cloud computing has brought down costs and market entry requirements for companies seeking to host their services online, this advancement has come at a cost. When considering the effect of Internet of Things (IoT) and Moore's law, it is predicted that the energy consumption of data centers will rise from 286 TWh in 2016 to 1287 TWh in 2030 [14]. This 447% increase in power consumption will skyrocket the financial and environmental effects of cloud computing and bring into question the sus-

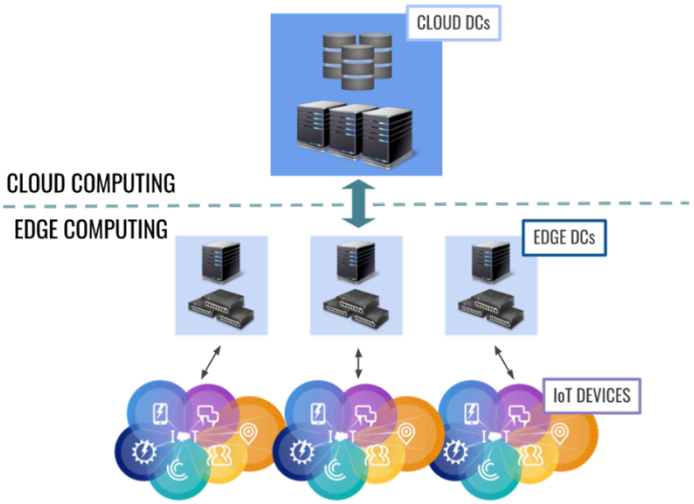


Figure 1. Cloud Computing and Edge Computing forming Large-Scale Systems [11]

tainability of the cloud. To this end, data center hosts have faced the need to employ techniques to monitor power utilization to increase energy efficiency, reduce operational costs and limit the environmental impacts of these centres [4].

2.2 Transition to Edge Computing

While traditional cloud computing has been designed for processing information in a centralized data center, edge computing represents a shift in computing closer to the data source or the "edge". Edge computing architecture involves placing smaller data centers located closer to edge devices to minimize the volume of data sent to the cloud. By providing short-period data analysis closer to the device side, edge computing provides advantages over simple cloud computing including improved system performance, protected data security and privacy, and reduced operational costs [16]. This relatively new approach has become increasingly popular as demand for IoT devices, like smartphones and PCs, are more prominent in consumers' lives. For instance, the number of smartphones and PCs has raised the number of devices connected to the internet to 12.5 billion in 2010 and 50 billion in 2020 [9], underscoring the need for greater utilization of edge computing in daily life.

2.3 Large-scale Systems

Edge computing extends centralized cloud data centers to form large-scale systems. Figure 1 illustrates a high-level view of this system, showing the distinction between the centralized cloud data centers, edge data centers, and IoT devices [11]. It should be mentioned that, while the term "large-scale" systems can be used to encompass multiple types of systems, the approach this paper takes represents a fusion of the centralized cloud and decentralized edge computing connected through telecommunication networks.

2.4 Power and Energy Monitoring

The growth in complexity of large-scale systems has escalated the need for power and energy monitoring techniques. This is reflected in the relationship between power monitoring and energy management. Although the two approaches are used interchangeably, they focus on different aspects of the "power and energy monitoring" processes outlined in this paper.

Power monitoring serves as the foundational level monitoring technique, providing the necessary data needed to optimize energy management systems. Through the continuous collection of power-related data, which is further complicated by the different levels of the power architectures in these systems, power monitoring helps to identify inefficiencies in power quality to save cost. On the other hand, energy management encompasses the planning and operation of energy consumption in these systems. Usually, these intelligent management systems utilize power information collected from power monitoring tools to monitor, control, and conserve energy. As a result, energy management systems lower the operational cost and environmental impacts of these systems, making them more sustainable.

This paper will analyze Power Usage Effectiveness (PUE) as the power monitoring metric used to optimize power utilization in large-scale systems. The effectiveness of this monitoring can be seen from the analysis of intelligent energy management systems used to increase energy efficiency.

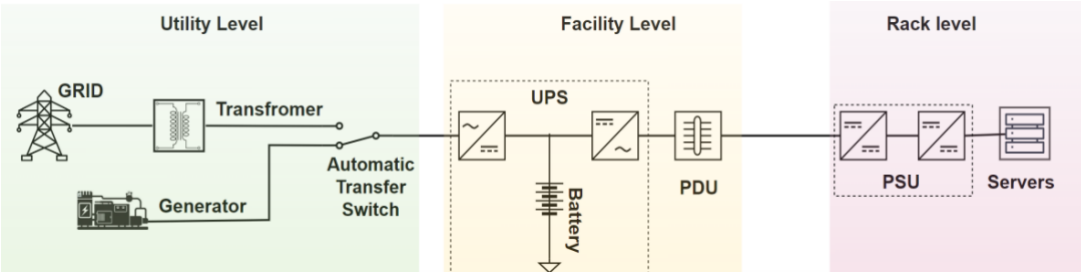


Figure 2. Data Center Power Distribution Architecture [1]

3 Power and Energy Monitoring Inside Data Centers

In this section, the current methods for power and energy monitoring within the data centers are discussed. However, it is important to first examine how power is distributed within the data centers as it can give insight into the PUE levels of measurement.

3.1 Data Center Power Distribution Architecture

Power distribution is a critical architectural consideration that can greatly affect a data center's performance, efficiency, and cost. It is designed to deliver electricity to multiple levels of the data center, starting from the utility source, transferring through the physical facility, and arriving at the individual servers and networking equipment. This architecture is structured around three main levels: Utility Level, Facility Level, and Rack Level. These levels each contain a unique set of components including transformers, switchgear, backup generators, power distribution units (PDU), power supply units (PSU), servers, and cooling systems. Figure 2 illustrates a simplified representation of a typical data center power distribution architecture.

At the utility level, data centers receive power from the grid, renewable energy sources, or generators. Transformers step down the voltage for utilization at the facility level. While this is an abstraction of a highly complex process, essentially the power distributed at this level is the source of power for the entire data center, and increasing efficiency at this level can affect the entire facility.

At the facility level, the power received from the utility level is distributed to various loads within the facility, including IT equipment, cooling systems, networking equipment, and lighting. The distribution is physically operated by switchgear, busbars, and panel boards. Uninterruptible power supply (UPS) systems are also at this level and offer backup

power in case of utility failures including power surges, spikes, and outages. Improving efficiency at this level can include methods for utilizing DC systems instead of the common AC systems for less power loss by conversions. For example, Europe utilizes lower input voltages, thus bypassing the need for PDUs and conversions which increases power efficiency [1].

At the rack level, data centers distribute power received from the facility-level UPS systems to individual racks of servers and cooling systems. Each rack of servers, or Power Domains (PD), can consist of a few thousand machines, an individual PDU and PSU, and rack-level UPS systems. Usually, these PDs are grouped into clusters on the data center floor and belong to a single job scheduling domain. The PDUs provide multiple outlets for connecting IT equipment and offer features for metering, switching, and most importantly, power monitoring.

3.2 Power Usage Effectiveness

Power Usage Effectiveness (PUE) is the most widely adopted metric that is used to measure power usage and analyze energy efficiency in data centers. PUE is defined by the ratio between the total power consumed by a data center and the power used by the servers [3], i.e.,

$$PUE = \frac{TotalFacilityEnergy}{ITEquipmentEnergy} \tag{1}$$

This power usage metric is important for understanding how much of the energy supplied to a data center is used for computing versus non-computing functions. A PUE value of 1.0 indicates that the data center has perfect efficiency as all the energy is used for servers. Research conducted by the U.S. Environmental Protection Agency (EPA) determined that the average PUE of modern data centers is 1.92 [5].

To accurately measure PUE, hardware components at multiple levels of the power distribution architecture work together in monitoring the power consumption at their designated position. The Green Grid provides a framework that outlines three distinct levels of metering for PUE measurement [18].

1. Basic Metering – Level 1 PUE measurement captures the IT load at the output of UPS equipment of the facility level. This basic metering approach represents the manual readings through a single meter on the

UPS output bus. Losses from transformers and electrical distribution are not considered as only one reading is counted toward this basic PUE measurement.

2. Intermediate Metering – Level 2 PUE measurement is described as a "hybrid metering solution" as it typically involves both automatic and manual recordings at the utility and facility levels. Measurements from the utility, switchgear, and cooling systems meters are all considered for this PUE measurement. However, this level still doesn't provide the level of granularity needed for accurately measuring PUE as losses from PDUs are counted as part of the infrastructure and not IT.

3. Advanced Metering – Level 3 PUE measurement describes the advanced approach utilized by modern data centers. At this level, the metering of all levels is done automatically. This approach includes measurements from all major power consumption hardware including IT, cooling systems, PDUs, and UPSs. This approach also measures values of meters from subsystems of these major components including chillers, pumps, fans, and lighting.

3.3 Energy Management Techniques for Cooling Optimization

Power analytic data like PUE plays a crucial role in optimizing energy management systems. These systems aim to increase energy efficiency in areas of the data center where power utilization is high. Figure 3 [1] illustrates the power consumption breakdown of an average data center. Although the largest share of power is allocated to IT equipment, cooling loads account for 38% of the power consumption.

These cooling systems are an essential part of the facility's infrastructure designed to remove heat generated by servers and other computing hardware. In general, Temperature control inside data centers is crucial for preventing overheating and maintaining optimal performance as these electronic components are designed to operate efficiently within a specific temperature range. The traditional approaches used by modern data centers include implementing computing room air conditioners (CRACs), rearranging server racks to separate rows into "cold aisles" and "hot aisles", and using chilled water systems [6].

While these methods optimize airflow and cooling fluid characteris-

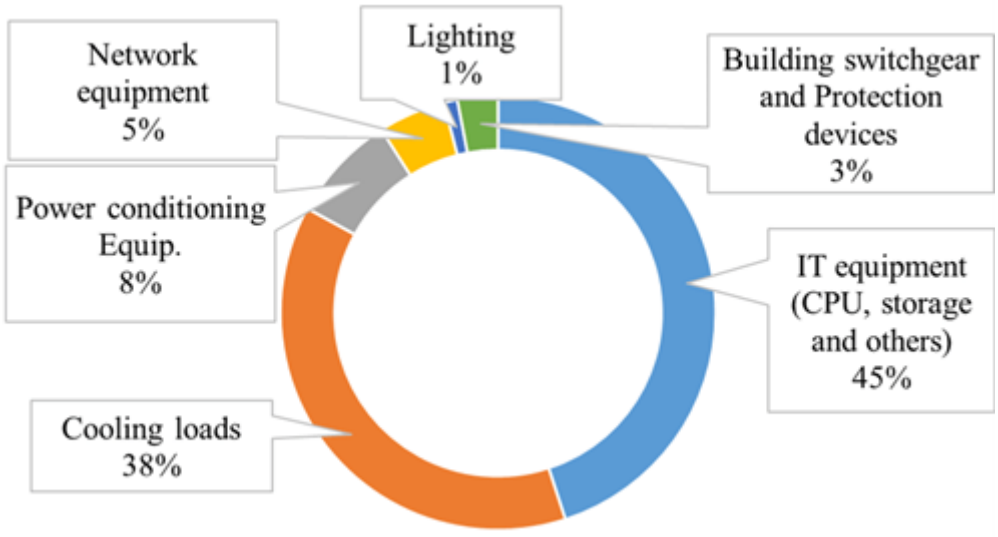


Figure 3. Analysis of power consumption in data centers [1]

tics to handle heat efficiently, they operate with constant, predetermined settings. To combat this, advanced adaptive cooling methods have been proposed to handle dynamic real-time data center needs. Li et al. developed a model that can predict temperatures near servers using continuous collected temperature and airflow data. As a result, "Thermocast" forecasts temperature better than a machine learning approach, predicting the thermal limits of servers 4.2 minutes before overheating [15]. On a larger scope, Google's DeepMind AI machine learning is utilized in its data centers to reduce energy used for cooling by 40%, which equated to a 15% reduction in overall PUE overhead [10].

3.4 Sustainability Techniques for Cooling Optimization

While energy management systems optimize cooling through adaptive algorithms using PUE and other power analytic data, sustainability techniques have also proven to be effective in limiting the power for cooling. The use of renewable energy sources over traditional "brown" energy can significantly improve the overall energy efficiency and sustainability in data centers. Utilizing solar, wind, and hydro energy does not only lower operational costs when compared to fossil fuels, but it also reduces its carbon emissions.

The largest cloud providers like Google, Microsoft, and Amazon have all demonstrated how the integration of renewable energy sources can enhance data center efficiency. From 2019 to 2020, Google's data centers increased their reliance on carbon-free energy (CFE) from 61% to 67%,

vowing to operate all their data centers with CFE by 2030. Microsoft has also worked towards operating its data centers on 100% of renewable energy by 2025. Amazon Web Services has a similar target to Microsoft, but they've achieved the best progress, currently relying on 85% renewable energy across its businesses [12].

This switch to carbon-free energy has led to even more cooling optimization techniques. One method involves the strategic placement of these data centers in geographically cooler areas. For example, Finland's cold climate enables data centers to utilize the natural cooling provided by the 5 degree Celsius seawater. Harnessing this natural energy can reduce the energy required by artificial cooling, offering a sustainable and cost-effective solution for energy efficiency. As a result, there has been a significant increase in data centers located in the Baltic region [17]. A more advanced technique that can further optimize cooling is utilized in Google's Carbon-Intelligent Compute Management System. This system focuses on reducing grid carbon emissions from its data center electricity. It accomplishes this goal by allowing flexible workloads to tolerate delays, pushing them to run at a less carbon-intensive time of day [8].

4 Challenges in Power and Energy Monitoring in Edge Computing

Monitoring power and energy consumption in edge environments poses unique challenges when compared to traditional data center settings. These challenges are primarily due to the distributed nature of edge computing.

When focusing on measuring PUE through the metering techniques in data centers, all the metering equipment for each system is located in one facility. However, as we expand into edge computing, these power meters are distributed geographically, making it difficult to physically measure. By extending power monitoring to include the estimated 30.9 billion edge devices, the wide variety of different power, cooling, and operational requirements for each device makes it even harder [8]. Although it is logistically complex to manage every edge device's power utilization, Data Center Infrastructure Management (DCIM) software can be utilized for monitoring and device management in small edge data centers. Although this software needs stable electrical power and sufficient ventilation, DCIM offers a method for remotely monitoring power analytic data. More importantly, DCIM software provides remote visibility and early warning in conjunction with device and environmental sensors and cameras [7].

Challenges also arise in the energy efficiency and sustainability of edge data centers. Through the construction of more edge nodes, energy efficiency of these networks significantly decreases, with a rising PUE value of 1.1 to 1.55 [2]. Due to its geographical distribution, edge nodes cannot utilize all the benefits that are received from switching to carbon-free energy. Arroba et. al states that this increase in PUE value is mainly due to efficiency problems in the cooling systems of these edge nodes and proposes a novel two-phase immersion cooling strategy for these edge deployments. After testing thew prototype of this approach, energy consumption and operating cost both decreased by 20% and 30% respectively.

5 Discussion

In evaluating the effectiveness of cooling optimization techniques, such as strategic geographical placement and the equipment layout within data centers, it becomes evident that these approaches offer tangible benefits in energy management across both centralized and decentralized computing architectures. However, the practical application of these strategies faces inherent limitations.

Geographical location is an important consideration when constructing cloud computing centers or edge computing hubs. Factors such as elevation and climate can greatly influence the cooling that nature can provide. Ideally, good geographical positioning can optimize cooling and reduce overhead costs related to artificial methods of dissipating heat. Yet, while geographic position can be advantageous in leveraging natural cooling resources, it may not always be feasible due to land availability, infrastructure costs, and proximity to other users, especially in the context of edge computing. Additionally, factors such as climate change and extreme weather events can introduce unpredictability to the effectiveness of natural cooling methods, creating the additional need for backup systems and contingency plans. Overall, while geographical location can lend a hand in dissipating heat in a computing system, it is not a sustainably reliable method of optimizing cooling within a large-scale system. On the other hand, equipment layout optimization within facilities can offer immediate cooling efficiency gains but requires continuous adaptation to technological advancements and changes in computing demand. The formation of “hot aisles” and “cold aisles” to direct heat away from the system can greatly increase cooling efficiency. Furthermore, combining equip-

ment layout with external vents and/or artificial cooling mechanisms can compound the benefits of a well-structured data center.

The transition towards more sustainable cooling solutions necessitates a deep understanding of these methods' potential and constraints, ensuring they are effectively integrated within the broader framework of large-scale computing systems' power and energy monitoring strategies. While natural and logistical optimization routes exist, a holistic approach to cooling optimization is necessary to fully address the scalability challenges in supporting the ever-growing demands of the global computing infrastructure. Data center operators must evaluate how best to utilize multiple cooling techniques in tandem with each other to ensure that their centers operate with maximum (sustainable) efficiency.

6 Conclusion

The transition of cloud to edge computing and the formation of decentralized large-scale systems has brought numerous challenges in managing the financial and environmental burdens of data centers and edge computing architectures. Data center operators are obligated to meet these challenges by integrating power and energy monitoring techniques to determine the best way to limit the environmental impact of their data centers and increase their long-term sustainability. Cooling optimization within a data center constitutes one of the most energy-demanding systems in a data center. PUE adaptive algorithms and other power analysis data have allowed data center operators to adjust cooling levels and find the optimal balance between cost and overall sustainability. Yet, challenges arise in determining the environmental impact of edge computing systems, given their far-reaching and decentralized nature. To mitigate this, DCIM software can be used to track energy usage over a decentralized network and provide the necessary data to maintain operations within acceptable environmental sustainability limits. This information can help data center operators to calibrate their cooling optimization strategies in the short and long-term, so as to meet carbon footprint limitations, reduce overhead costs, and create a more sustainable and eco-friendly computing system.

References

[1] Adeel Arif. Examining future data center power supply infrastructures. *Master's thesis in Technology*, 2024.

[2] Patricia Arroba, Rajkumar Buyya, Román Cárdenas, José L.Risco-Martín, and José Manuel Moya. Sustainable edge computing: Challenges and future directions. Master's thesis, University of Melbourne, 2023.

[3] C. Belady and C. Malone. Efficiency metric called pue. *Green Grid*, 2006.

[4] Zhiwei Cao, Xin Zhou, Han Hu, Zhi Wang, and Yonggang Wen. Toward a systematic survey for carbon neutral data centers. *IEEE Communications Surveys and Tutorials*, 2022.

[5] J. Cho and Y. Kim. Improving energy efficiency of dedicated cooling system and its contribution towards meeting an energy-optimized data center. *Applied Energy*, 165:967–982, 2016.

[6] datacenter.com. Data center cooling: Future of cooling systems, methods and technologies, 2023. https://www.datacenters.com/news/data-center-cooling-future-of-cooling-systems-methods-and-technologies.

[7] Patrick Donovan. How modern dcim addresses cio management challenges within distributed, hybrid it environments. *White Paper*, 2022.

[8] Borko Drljaca. Edge computing challenges and how to solve them, 2022.

[9] Dave Evans. The internet of things: How the net evolution of the internet is changing everything. Technical report, Cisco, 2011.

[10] Rich Evans and Jim Gao. Deepmind ai reduces energy used for cooling google data centers by 40*The Keyword*, 2016.

[11] Artecs Research Group. Energy efficiency and large scale computing, 2021.

[12] Jane Harkness. Tracking the transition to renewable energy across data centers, 2023.

[13] Avita Katal, Susheela Dahiya, and Tanupriya Choudhury. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster compute 26*, 26:1845–1875, 2022.

[14] Martijn Koot and Fons Wijnhoven. Usage impact on data center electricity needs: A system dynamic forecasting model. *Applied Energy*, 291, 2021.

[15] L. Li, C.-J. M. Liang, J. Liu, S. Nath, A. Terzis, and C. Faloutsos. Thermocast: a cyber-physical forecasting model for datacenters. *ACM*, 2011.

[16] Dongqi Liu, Haolan Liang, Xiangjun Zeng, Qiong Zhang, Zidong Zhang, and Mintong Li. Edge computing application, architecture, and challenges in ubiquitous power internet of things. *Frontiers in Energy Research*, 2022.

[17] Motiva. Energy-efficient data centre. Technical report, Motiva, 2011.

[18] U.S. Department of Energy. Data center metering and resource guide. *Better Buildings*, 2017.

[19] PeterMell and Timothy Grance. The nist definition of cloud computing. Technical report, National Institute of Standards and Technology, September 2011. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf.

[20] Precedence Research. Cloud computing market: Forecast 2023-2032. Technical report, Precedence Research, 2023.

[21] Duncan Stewart, Nobuo Okubo, Patrick Jehu, and Michael Liu. The cloud migration forecast: Cloudy with a chance of clouds. *Technology, Media, and Telecommunications Predictions 2021*, 2020.

Exploring the Effectiveness and Challenges of Self-Help Applications for Mental Health Issues Raised by COVID-19

Ulaş Sedat Aydın
ulas.aydin@aalto.fi

Tutor: Sanna Suoranta

Abstract

This paper examines the advantages and challenges of self-help applications for mental health. Individualized treatment approaches for mental health problems were often utilized before COVID-19. However, the prevalence and scope of mental health problems have expanded due to the pandemic. Self-help applications have emerged as a significant tool, potentially aligning well with the needs and habits of individuals affected by these changes. This paper highlights the advantages and challenges of self-help applications, considering their functions and features. By exploring these aspects, this paper aims to provide a comprehensive overview of the current state of self-help applications and what methods can be utilized to deliver improved usability, accessibility, and acceptability. Finally, this paper aims to predict the future of self-help applications using analysis and technological developments.

KEYWORDS: Cognitive-behavioral therapy, Self-help applications, Depression, COVID-19

1 Introduction

After 2020, the COVID-19 pandemic had a profound impact on widespread mental disorders, such as depression, loss of focus, anxiety, and ADHD, particularly those with pre-existing mental illnesses. For instance, Lewis et al. (2022) found that 60% of participants with a history of mental health issues reported a degeneration in their mental state during the pandemic. This degradation was correlated with factors such as younger age, difficulty accessing mental health services, and socioeconomic challenges. The study underlines the role of the pandemic in intensifying mental health problems, indicating the need for targeted interventions to mitigate the effects of the epidemic on vulnerable groups [7].

In the United States, approximately a quarter of the population experiences serious mental health disorders annually, affecting half of the population over their lifetime, and a majority of them remains untreated [5]. This gap in the treatment of mental disorders shows that only individuals who are aware of their mental health and have both the financial resources and time can access psychological therapy to solve their problems.

In this context, self-help applications accessible through mobile devices offer a potential solution for mental health disorders. This paper discusses the effectiveness and challenges of utilizing self-help applications for mental health. additionally highlighting their impact on specific demographic groups such as adolescents and older populations.

Section 2 examines the usage of self-help applications, including their functions and how they are used to enhance mental health. Section 3 and Section 4 assess the advantages and challenges of these applications. It is followed by Section 5, which explores ways to enhance their usability, accessibility, and acceptability. Finally, Section 6 discusses future directions for self-help applications in mental health.

2 Usage of Self-Help Applications

Self-help applications are digital mobile tools primarily designed to aid people in improving their mental well-being and awareness [6]. Users engage with these applications in various ways, depending on their specific needs and the features provided by the application.

The COVID-19 pandemic has highlighted the importance of self-help

applications, mainly due to rising anxiety and depression levels. With the shift towards remote working and the comfort of online tasks, people have increasingly turned to these applications for mental health support. These tools offer immediate, accessible, cost-effective assistance, eliminating physical space limitations. Therefore, the pandemic has highlighted the significance of self-help applications in managing mental health challenges and facilitated their enhanced utility through increased technology adoption.

In order to address the mental health problems that have increased due to the pandemic, self-help applications utilize diverse therapeutic methods and physical approaches. For instance, some applications monitor user activities, such as sleep patterns, physical activity, mood fluctuations, and stress levels, to provide personalized feedback and recommendations. In addition to monitoring user activities, they integrate game-like elements and guidance, such as progress tracking, achievement badges, and interactive challenges, to motivate users and make their experience more interactive. By integrating these features, self-help applications offer personalized meditation sessions and regular exercise programs based on the users' health data, ensuring that users remain active and committed to their mental health journey.

3 Advantages and Effectiveness of Self-Help Applications

3.1 Eliminating Physical Visit Requirements

Self-help applications assist individuals' psychological treatment and mental well-being by eliminating physical visits. This aspect significantly reduces the time that users lose on their journey to improve their mental health. These applications can enable users to manage their mental health more flexibly by providing features such as meditation, online psychotherapy sessions, and physical activity solutions to improve well-being. The time saved by eliminating travel and waiting times for traditional face-to-face sessions can be very practical, especially for people with tight schedules or limited access to traditional services.

3.2 Cost Efficiency

Self-help applications can potentially make mental health solutions more affordable. Warmerdam et al. showed that internet-based therapy may be a cost-effective alternative to clinical treatment for treating depression in adults [14]. For instance, recognizing early signs of burnout allows people to be more mindful, adjust their pace, and mitigate the risk of experiencing burnout due to the awareness enhanced by self-help applications. In contrast, traditional mental treatment may be expensive, making it challenging to realize burnout before it is too late. Self-help applications also provide a bridge to professional treatment, easing the transition and maintaining well-being during recovery [8].

4 Challenges in Developing Self-Help Applications

4.1 Privacy and Data Security

Privacy is essential in self-help applications due to the sensitive nature of the information involved. Users of self-help applications share personal health information and expect that information to remain private. Surani et al. indicate that users are more likely to share sensitive mental health information in digital environments when they trust that their data will be kept confidential and secure [11]. However, privacy breaches can lead to severe consequences, including loss of trust in self-help healthcare applications and even degenerating mental health symptoms due to increased anxiety and stress [12].

In 2020, Vastaamo, a private psychotherapy company in Finland, experienced a severe security breach, leaking confidential and sensitive therapy session notes belonging to thousands of patients. This breach not only compromised the privacy of individuals seeking therapy but also caused profound psychological consequences for patients after the incident due to extreme distress, anxiety, and fear of exposure [4]. This breach highlights the importance of strict data protection rules to protect user information from unauthorized access and data leaks.

4.2 Adopting Adults and Elderly

Eisner et al. investigated the prevalence of smartphone ownership and interest in mental health applications among surveyed individuals, reporting that a significant majority (90%) owned smartphones, and nearly as many (88%) expressed their willingness to explore mental health applications [2]. Moreover, adolescents who are accustomed to daily phone use are expected to find it easier to integrate personal development and self-help applications into their routines. However, adopting self-help applications may be difficult for adult and elderly mobile users because they mostly use mobile devices only for functional purposes, leading to difficulty starting and getting used to these applications.

In order to improve the usability of self-help applications for older users, various design choices can be taken into account for their specific needs. For instance, simplifying the interface with larger icons, buttons, and text can enhance visibility and ease of use. Additionally, intuitive application navigation between features with a clear flow can prevent confusion. Incorporating voice commands and audio assistance can provide alternative interaction methods for those who struggle with touchscreens or small text. An interactive onboarding process can guide users through the features and functionalities, promoting better understanding and engagement. By integrating these design considerations, self-help applications can become more user-friendly for older adults, potentially increasing their acceptance and regular use of such digital mental health solutions.

5 Enhancing Usability, Accessibility, and Acceptability of Self-Help Applications

In order to address challenges in self-help applications and to enhance the usability, accessibility, and acceptability of such tools, it is crucial to consider various design choices, especially considering the wide range of mental health issues raised by COVID-19. Before the pandemic, mental health issues were relatively less common, allowing for more personalized treatment approaches in specialized environments. However, due to COVID-19, mental health problems have become more prevalent in different demographic groups, and their scope has been expanded [7]. Hence, it is essential to consider usability, accessibility, and acceptability while de-

veloping self-help applications so people with different backgrounds and abilities can use them.

5.1 Enhancing Usability

Developers can focus on intuitive design that accommodates users with different levels of technical proficiency to increase the usability of self-help applications. Usability is crucial for self-help applications designed to support users suffering from mental issues due to the significant effects of mental disorders on people's cognitive abilities, motivation, and attention span. For instance, users going through intense depression or anxiety may experience symptoms such as shortened attention span and memory loss. It can be challenging for these people to remember long instructions or get used to the application's complex features. Furthermore, it may be extra challenging for them to find motivation to use self-help applications.

Streamlining navigation within the application, providing straightforward instructions, and interactive tutorials help users understand and use the application effectively. Nielsen states that user-centered design principles are crucial in creating intuitive interfaces [10]. Considering the effects and difficulties that mental disorders have on people's cognitive abilities and emotional states, developers must put more effort into creating more user-friendly applications that support users in their mental health journey.

5.2 Enhancing Accessibility

In order to enhance the accessibility of self-help applications, it is essential to consider implementing responsive designs that can adjust the interface depending on the device and the screen size. Furthermore, providing text-to-speech functionality in those applications can significantly enhance accessibility for people with visual impairments. For instance, the application can read the mindfulness meditation instructions and allow visually impaired users to engage better with the application. Providing subtitles or sign language options for audio content in the application can also assist people with hearing problems.

On the other hand, utilizing accessibility standards can significantly enhance the accessibility of self-help applications and ensure compatibility with assistive technologies, such as screen readers. For instance, the Web Content Accessibility Guidelines (WCAG) is an accessibility standard

that provides extensive recommendations to improve web content accessibility, which would help to ensure that people with disabilities have better access to self-help applications [13].

5.3 Enhancing Acceptability

In order to enhance the acceptability of self-help applications among different user groups, it is crucial to appease concerns related to privacy and data security. Building trust can be achieved through transparent communication with users about the data handling process and by ensuring compliance with high data protection standards. This highlights the importance of not only being transparent but also ensuring that the information provided is accessible and understandable to all users by avoiding the use of overly complex language that may inhibit user participation. However, it is vital to recognize that transparency mechanisms alone may not be entirely effective. Acquisti et al. show that a significant majority of Internet users do not read privacy policies, and of those who do, almost half find the policies written in language too complex to understand [1].

On the other hand, providing personalization features in line with users' needs and preferences can foster engagement and make these applications more appealing to a broader range of users. For instance, developers can achieve a more personalized experience by integrating a mood-tracking feature into self-help applications, especially in cases of mental disorders such as depression and anxiety, which have dramatically increased after the pandemic. The mood tracking feature encourages users to record their emotional states during the day through the application. Depending on these analytics, the application can create personalized mindfulness meditation programs or develop a method more appropriate to the person's current emotional state in cognitive behavioral therapy (CBT). Moreover, thanks to the mood tracking feature, in situations such as perceived depression or high stress, the application can help users get through this process more effortlessly and smoothly with more motivating quotes or meditation sessions instead of physical activity recommendations.

6 Future Directions for Self-Help Applications

As self-help applications continue to evolve, emerging technologies such as artificial intelligence (AI) and blockchain have the potential to innovate mental health care. AI can increase efficacy through machine learning algorithms by personalizing treatment plans based on users' previous data. It also helps predict the success of the interventions. For instance, Meinlschmidt et al. (2020) demonstrates the potential of random forest algorithms to predict the efficacy of psychotherapeutic micro-interventions based on daily mood assessments. This approach highlights the importance of AI's ability to tailor interventions to suit individual mood dynamics, potentially increasing the effectiveness of mental health self-help applications [9].

On the other hand, integrating Blockchain technology into self-help applications can provide improved privacy and data security in the future. Thanks to being decentralized, blockchain securely and immutably records users' sensitive health data. According to Fekih and Lahami (2020), blockchain offers a promising solution to the privacy and security challenges in digital health care. Therefore, integrating blockchain into self-help applications can build trust among users by ensuring that their data is handled securely [3].

7 Conclusion

In conclusion, self-help applications for mental health hold significant promise in solving common mental disorders, especially those that have increased after COVID-19, by offering a convenient, accessible, and affordable way. Although there are challenges in developing and adopting these applications, they can be an alternative to traditional mental health treatment methods when design actions are taken to increase usability, accessibility, and acceptability.

References

[1] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Privacy and human behavior in the age of information. *Science (New York, N.Y.)*, 347:509–14, 01 2015. 10.1126/science.aaa1465.

[2] Emily Eisner, Natalie Berry, and Sandra Bucci. Digital tools to support mental health: a survey study in psychosis. *BMC Psychiatry*, 23(1):726, 10 2023. 10.1186/s12888-023-05114-y.

[3] Rim Fekih and Mariam Lahami. *Application of Blockchain Technology in Healthcare: A Comprehensive Study*, pages 268–276. 06 2020. 10.1007/978-3-030-51517-1_23.

[4] Eija Heikkilä and Jaana Hevonoja. Useat tahot tutkivat psykoterapiakeskus vastaamon tietomurtoa ja kiristystä – kyberturvallisuuskeskus pitää tapausta poikkeuksellisena, 2020. https://yle.fi/a/3-11605223.

[5] Alan E. Kazdin. Addressing the treatment gap: A key challenge for extending evidence-based psychosocial interventions. *Behaviour Research and Therapy*, 88:7–18, 2017. 10.1016/j.brat.2016.06.004.

[6] Rachel Kenny, Barbara Dooley, and Amanda Fitzgerald. Developing mental health mobile apps: Exploring adolescents' perspectives. *Health Informatics Journal*, 22(2):265–275, 2016. 10.1177/1460458214555041.

[7] Katie J. S. Lewis, Catrin Lewis, Alice Roberts, Natalie A. Richards, Claudia Evison, Holly A. Pearce, Keith Lloyd, Alan Meudell, Bethan M. Edwards, Catherine A. Robinson, and et al. The effect of the covid-19 pandemic on mental health in individuals with pre-existing mental illness. *BJPsych Open*, 8(2):e59, 2022. 10.1192/bjo.2022.25.

[8] Thies Lüdtke, Lilian Klara Pult, Johanna Schröder, Steffen Moritz, and Lara Bücker. A randomized controlled trial on a smartphone self-help application (be good to yourself) to reduce depressive symptoms. *Psychiatry Research*, 269:753–762, 2018. 10.1016/j.psychres.2018.08.113.

[9] Gunther Meinlschmidt, Marion Tegethoff, Angelo Belardi, Esther Stalujanis, Minkyung Oh, Eun Kyung Jung, Hyun-Chul Kim, Seung-Schik Yoo, and Jong-Hwan Lee. Personalized prediction of smartphone-based psychotherapeutic micro-intervention success using machine learning. *Journal of Affective Disorders*, 264:430–437, 2020. 10.1016/j.jad.2019.11.071.

[10] Jakob Nielsen. 10 usability heuristics for user interface design, 2024. https://www.nngroup.com/articles/ten-usability-heuristics/.

[11] Aishwarya Surani, Amani Bawaked, Matthew Wheeler, Braden Kelsey, Nikki Roberts, David Vincent, and Sanchari Das. Security and privacy of digital mental health: An analysis of web services and mobile applications. In Vijayalakshmi Atluri and Anna Lisa Ferrara, editors, *Data and Applications Security and Privacy XXXVII*, pages 319–338, Cham, 2023. Springer Nature Switzerland. 10.1007/978-3-031-37586-6_19.

[12] John Torous, Gerhard Andersson, Andrew Bertagnoli, Helen Christensen, Pim Cuijpers, Joseph Firth, Adam Haim, Honor Hsin, Chris Hollis, Shôn Lewis, David C. Mohr, Abhishek Pratap, Spencer Roux, Joel Sherrill, and

Patricia A. Arean. Towards a consensus around standards for smartphone apps and digital mental health. *World Psychiatry*, 18(1):97–98, 2019. 10.1002/wps.20592.

[13] W3C World Wide Web Consortium. Web Content Accessibility Guidelines (WCAG) 2.1. Technical report, W3C, 2023. https://www.w3.org/TR/WCAG21/.

[14] Lisanne Warmerdam, Filip Smit, Annemieke van Straten, Heleen Riper, and Pim Cuijpers. Cost-utility and cost-effectiveness of internet-based treatment for adults with depressive symptoms: Randomized trial. *J Med Internet Res*, 12(5):e53, Dec 2010. 10.2196/jmir.1436.

Robustness Assessment in ML Systems

Viktoriia Kovalenko
viktoriia.kovalenko@aalto.fi

Tutor: Samuel Marchal

Abstract

As machine learning systems become increasingly prevalent, ensuring their robustness against adversarial attacks is crucial. This paper explores recent developments in robustness assessment in machine learning systems, focusing on both adversarial and certified approaches. We explore various types of attacks, including evasion, poisoning, and backdoor attacks, and examine the metrics used for robustness assessment across different domains such as natural language processing and computer vision.

Comparing robustness assessment frameworks sheds light on their usability and contribution potential. Despite the availability of open-source tools, several challenges arise due to limited documentation and varying attack types and metrics. By highlighting these challenges and identifying areas for future research, this paper aims to facilitate the development of more effective and user-friendly frameworks to protect machine learning systems against adversarial attacks.

KEYWORDS: robustness assessment, robustness assessment evaluation metrics, adversarial attacks, robustness assessment frameworks

1 Introduction

In recent years, machine learning has been widely adopted in numerous industrial and research projects. The advent of deep learning models has revolutionized various tasks that were performed exclusively by humans, namely computer vision and natural language processing (NLP). Deep learning models process extensive amounts of data to generate complex patterns and predict an output, thereby frequently making them uninterpretable. In contrast to software development testing, machine learning systems cannot be easily tested in production due to their uncertainty, which makes them vulnerable to adversarial attacks.

Testing machine learning systems becomes increasingly critical for mission critical software systems. While traditional metrics on validation and test data can be beneficial for robustness assessment, they may neglect hidden weaknesses in deep learning systems, hence more advanced evaluation techniques need to be defined. To address the flaws, an approach of robustness assessment in machine learning systems has currently gained great popularity.

Robustness may be assessed with empirical and certified approaches, which differ in providing a guarantee for safety for machine learning models. Empirical (also known as adversarial) evaluation process is a subset problem which estimates the robustness of a model under adversarial attacks, while certified robustness assessment warrants safety by releasing a robustness certificate.

Typically, adversarial robustness assessment is performed by executing an attack and applying evaluation metrics either manually or with dedicated software frameworks. However, due to their diverse applications, limited information exists regarding the generalization and comprehensive comparison of these tools. Consequently, researchers may need a considerable amount of time to explore various resources and find the appropriate framework for testing their model.

This paper overviews recent advances in adversarial and certified assessment methods. It compares open-source adversarial robustness frameworks due to the limited availability of certified robustness tools, which are currently provided by only a few existing frameworks [17], [42], [9], [39]. Additionally, this paper aims to identify the potential for adversarial robustness assessment in areas that have been insufficiently covered.

This paper is organized as follows. Section 2 presents a classification

of adversarial attacks, a summary of robustness assessment metrics and a recap of robustness certificate methods. Section 3 provides state-of-the-art adversarial robustness assessment toolkits. Section 4 compares robust assessment frameworks according to their application domains, range of metrics, and usability. Finally, Section 5 concludes the current approaches for robustness assessment.

2 Adversarial and certified assessment attacks and metrics

Adversarial attacks play a crucial role in assessing adversarial robustness since they evaluate the behavior of the machine learning model under abnormal conditions. On the contrary, certified robustness assessment evaluates the guaranteed machine learning model's lower bound of robust accuracy against both known and unknown attacks under specific conditions [17]. This section describes the adversarial attacks on machine learning models along with the common and advanced robustness assessment metrics and summarizes the current state of research on robustness certification.

2.1 Classification of Adversarial Attacks

The primary attack types for machine learning models include poisoning, extraction, evasion, inversion, inference, backdoor and model injection attacks [3]. This research mostly focuses on the three types of attacks, which are evasion, poisoning and backdoor. Attacks can also be categorized into three main categories based on the knowledge of the model: black-box, grey-box, and white-box attacks. In grey-box and white-box attacks an attacker is partially or completely aware of model parameters, training data and other features, while in black-box attacks an attacker has no information about the model [5].

NIST AI report [27] classifies evasion attacks into black-box (optimisation-based, universal evasion, physical evasion) and white-box attacks (score-based, decision-based). Optimization-based, score-based and decision-based evasion attacks are of particular interest within our research. Optimization-based attack [11], [20], [2], [16] calculates the gradients of the model's loss function, aiming to generate adversarial examples that are close to the original testing samples. Score-based and decision-based evasion attacks aim to obtain adversarial examples by querying the model classifi-

cation or confidence scores.

Poisoning attack [18], [34] constructs a poisoned training dataset for a model that will predict a wrong prediction label while having the same training loss as a model trained on an original dataset. Neural backdoor attack [12], [4], [14] occurs when an attacker modifies a subtle set of training data causing a model to misbehave on the test time.

2.2 Robustness Assessment Metrics

Natural language processing metrics can be summarized and grouped into two semantic categories, namely performance evaluation metrics (accuracy rate, success rate, error rate, IBP accuracy) and fairness evaluation metrics (fairness, sensitivity) [26]. Performance evaluation metrics are fundamental for assessing the overall performance of models and evaluating the effectiveness of evasion attacks. They are calculated by finding the percentage of successful/failed classification of an input text. On the other hand, IBP accuracy is particularly useful for verifying model robustness against adversarial attacks. The advantage of the metrics is the ease of calculation, however, they may not be fully descriptive due to the lack of quality properties of the adversarial examples that commit to the metric rate. Fairness and sensitivity metrics ensure that models are not biased against specific ethnic groups or genders. These metrics are essential for detecting NLP biases, nonetheless, they should not be considered as evidence of bias absence.

Computer vision robustness metrics categories include but not limited to performance evaluation metrics(classification accuracy, precision at Top-1) and robustness evaluation metrics(corruption error, robustness score) [8]. Classification accuracy and precision at Top-1 are valuable for evaluating the effectiveness of a model in classification tasks, as they offer ease of interpretation and comparability. The downside of the metrics is they may not fully demonstrate the potential to be robust against various perturbations or corruptions in the input data. Alternatively, corruption error and robustness scores are designed to evaluate the ratio of the corruption error or accuracy of the model to its clean error or accuracy.

In the context of *graph neural networks* certified accuracy metric [37] is utilized for measuring robustness in structural perturbations. J. Xu et al. [41] introduced robustness evaluation metrics for graphs, such as classification margin, adversarial risk and adversarial gap. These metrics offer a more comprehensive assessment of the robustness since they

examine both discrete and continuous adversarial examples.

2.3 Advanced Robustness Assessment Metrics

Brendel et al. [1] introduced boundary attack with L2-distance metric across all samples with adversarial perturbation on a specific example. Similarly, Weng et al. [40] proposed the CLEVER metric for the attack-agnostic and feasible measurement of large neural networks. Additional metrics for adversarial attacks on object detection models were suggested in [43], namely, False Positive Attack, False Negative Attack, Fractional Precision, and Fractional Recall. The robustness evaluation framework presented in [13] contains 23 data-oriented and model-oriented metrics for conducting evaluations of a machine learning model. CERScore defined in [35] presents robustness measurement of a black-box model. FOL metric proposed by Wang et al. [38] along with the RobOT framework provides quantification of the loss in each test case for improving robustness. Three additional sets of metrics, namely, out-of-distribution generalization, stability, and uncertainty were also proposed to evaluate the robustness of classification models [7]. An average robustness was introduced in [23] which indicates the robustness to adversarial perturbations of a classifier.

2.4 Robustness certification

Robustness certification aims to provide formal proof and guarantee of resistance against adversarial attacks. The process consists of two methodologies defined in [17], specifically, robustness verification and robust training. The main objective of robustness verification is to provide a lower bound of robust accuracy score against any attacks under certain perturbation conditions, while robust training can enhance certified robustness by training deep neural networks without theoretical limitations.

SoK [17] is the first open-source framework for robustness verification and certification methodologies. It provides a benchmarking tool for over 20 verification and training methods. GCERT [42] presents complete robustness certification under various input modifications. Ferrari et al. [9] developed branch-and-bound based complete verification for medium-size machine learning models. Mirman et al. [22] implemented the GenProve framework which computes bounds for the robustness of a generative model.

3 Robustment Assessment Frameworks

The research area has a significant number of developed frameworks with empirical attacks, however, we choose to focus on well-documented and actively maintained toolkits. Additionally, we prioritize attacks that provide quantifiable results, namely robustness assessment metrics, allowing more objective comparisons. We also analyze the domain of each library, the types of data and models it can handle and the user-oriented design of each library.

3.1 Cleverhans

Cleverhans [30] library introduced by Papernot et al. is a robustness assessment software library that implements 11 evasion attacks. It explores attacks exclusively on deep learning models with image data and employs test accuracy metric for model benchmarking. The provided Github documentation contains 2 tutorials with attacks applied to CIFAR-10 [15] and MNIST [6] datasets. Cleverhans supports 3 machine learning frameworks, hence, authors mostly focus on PyTorch attack implementation.

3.2 Foolbox

Foolbox [33], [32] is a Python package for generating adversarial attacks, namely 9 gradient-based evasion attacks, 3 score-based evasion attacks and 8 decision-based evasion attacks. A robust accuracy metric is used for evaluating the robustness of the classifier over the magnitude of perturbation. The Foolbox framework was designed for generating image data attacks for computer vision models, consequently, it is incompatible with models in other areas of machine learning. Foolbox works natively with models in 3 machine learning frameworks and has a documentation webpage with one tutorial published in Github.

3.3 SecML

Cleverhans and Foolbox attack implementations were integrated into a SecML [21] software library along with the extended implementation of evasion and poisoning attacks. SecML implements 11 evasion attacks against deep neural networks and 3 poisoning attacks against *scikit-learn* classifiers with 19 conventional accuracy metrics to measure the robustness. The advantage of the library is the ability to monitor loss function

values and the intermediary stages of the model through attack iterations and to provide quantitative analytics of function and gradient evaluations. Additionally, its interface enables visualization of attack performance and logging of the execution time. The framework supports both image and tabular data and contains documentation along with the GitHub page with 15 tutorials.

3.4 DeepSec

DeepSec [19] is a platform for evaluating attacks with defenses to measure the affect of attacks. The framework supports the PyTorch library, which consists of 10 robustness assessment metrics along with 16 evasion attacks and 13 defense methods. Similarly to Cleverhans [30] and Foolbox [33], it works with computer vision models and image datasets. The framework offers brief guidance on generating adversarial examples, executing defense, and evaluating the robustness of each defense.

3.5 Adversarial Robustness Toolbox (ART)

ART [25] implements 5 different adversarial attacks, including 46 evasion attacks and 8 poisoning attacks and 3 backdoor attacks along with 3 robustness assessment metrics. The framework, which supports 9 machine and deep learning libraries, accommodates diverse types of source data, including images, tables, audio and video. It supports a variety of machine learning and deep learning models and provides robustness certification with DeepZ [36]. The GitHub page contains a vast amount of tutorials for the different libraries. This makes the ART framework a preferred tool for assessing machine learning models for both researchers and practitioners.

3.6 TextAttack

A toolkit of 16 evasion NLP attacks for benchmarking NLP models was first implemented in TextAttack [24] library. According to the authors, it aims to address similar tasks as Cleverhans [30] in the NLP domain, though it implements attacks with a set of common components. TextAttack provides a summary of attack robustness with 5 conventional accuracy metrics and has a documentation website with a basic tutorial for attacking a large language model.

3.7 Robustness Gym

Robustness Gym [10] was implemented for robustness assessment of NLP models under evasion attacks with 6 conventional accuracy metrics. These attacks are performed on generated data slices, which are subsequently aggregated into a testbench to generate a descriptive report. Robustness Gym utilizes TextAttack [24] framework adversarial attacks while supporting three other evaluation idioms and it has a documentation page with a basic tutorial.

3.8 Counterfit

Counterfit [31] is a software tool that enables ingesting multiple data types with corresponding attacks from different frameworks. It leverages TextAttack [24], ART [25] and AugLy [29] adversarial attacks for text, image, and tabular attacks. Counterfit enables measuring conventional accuracy metrics for images and tabular data and presents identical metrics to TextAttack [24] for text adversarial attacks. The overall number of attacks account for 62 evasion attacks, 8 poisong attacks and 3 backdoor attacks along with 8 robustness assessment metrics. The platform has a GitHub documentation page with 6 tutorials for both image and text attacks.

3.9 TrojanZoo

TrojanZoo [28] pioneered the development of a framework for evaluating backdoor attacks in deep neural networks in the computer vision domain. The toolkit contains 8 backdoor representative attacks with 6 robustness assessment metrics. Example files demonstrating the basic usage of the framework are available on the GitHub page of the project.

4 Comparison of Robustment Assessment Frameworks

In the previous section, we discussed the functionalities of each robustness assessment framework, outlining their capabilities. However, in addition to functionality, user accessibility factors should be examined for selecting the right framework for assessing model robustness. This section analyzes the strengths and weaknesses of each framework considering factors such as usability, potential for user contributions, and the ongoing

maintenance of the framework. Consequently, based on these considerations, we suggest the most suitable frameworks for different tasks.

4.1 Framework Usability Assessment

Evaluation of open-source robustness assessment frameworks highlights a potential framework adoption challenges since several frameworks, namely Cleverhans [30], DeepSec [19], Counterfit [31] do not provide an API documentation page. Moreover, Cleverhans [30], SecML [21], DeepSec [19] and TrojanZoo [28] lack adequate documentation regarding the metrics employed in robustness assessment, which requires users to investigate them in the source code. Additionally, only a few frameworks (SecML [21], ART [25] and Counterfit [31]) offer a sufficient number of tutorials to enhance understanding of the framework API.

4.2 Contribution and Maintenance

Active contribution to open-source frameworks and diligent maintenance are crucial for ensuring they remain valuable and effective in addressing the shifting demands of users. An analysis of the reviewed frameworks indicates recent contributions within the past year, with four frameworks (Foolbox [33], ART [25], TextAttack [24], and TrojanZoo [28]) having contributions within the last three months. Moreover, ART [25] and TextAttack [24] frameworks offer the most informative contribution guides and maintain active Slack channels for their contributors. This essential characteristic facilitates the prompt addressing of any developmental errors or feature suggestions within the framework.

4.3 Comparative Analysis of Attack Types and Robustness Assessment Metrics

Finally, we compare the functionality of each framework based on the attacks and metrics in a concise manner. Table 1 presents a comparative analysis of attack types and robustness assessment metrics across different robustness assessment frameworks. As illustrated in the Table 1, Counterfit [31] stands out for having the most evasion and poisoning attacks, while SecML [21] and DeepSec [19] offers the widest range of robustness assessment metrics. TrojanZoo [28] excels in backdoor attacks, providing a valuable resource for assessing various backdoor attack scenarios.

	Evasion Attacks	Poisoning Attacks	Backdoor Attacks	Robustness Metrics
Cleverhans [30]	11	0	0	1
Foolbox [33], [32]	20	0	0	1
SecML [21]	1	3	0	19
DeepSec [19]	16	0	0	10
ART [25]	46	8	3	3
TextAttack [24]	16	0	0	5
Robustness Gym [10]	16	0	0	6
Counterfit [31]	62	8	3	8
TrojanZoo [28]	0	0	8	6

Table 1. Comparison of attacks and robustness assessment metrics across various robustness assessment frameworks. The table presents the number of evasion attacks, poisoning attacks, backdoor attacks and the robustness assessment metrics evaluated by each framework.

5 Conclusion

This paper overviews recent developments in robustness assessment regarding both adversarial and certified approaches. It outlines common attacks and domain-specific metrics utilized for evaluating the effectiveness of attacks and assessing adversarial robustness. Additionally, the study explores predominant open-source frameworks for empirical robustness assessment and discusses their adversarial attacks with corresponding robustness assessment metrics. The last section compares the highlights and challenges of each toolkit and suggests potential areas for researchers to explore in order to develop an enhanced framework for robustness assessment.

References

[1] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[3] Pin-Yu Chen and Sijia Liu. Holistic adversarial robustness of deep learning models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15411–15420, 2023.

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[5] Tapadhir Das. Vulnerabilities of machine learning algorithms to adversarial attacks for cyber-physical power systems.

[6] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[7] Josip Djolonga, Frances Hubis, Matthias Minderer, Zachary Nado, Jeremy Nixon, Rob Romijnders, Dustin Tran, and Mario Lucic. Robustness Metrics. `https://github.com/google-research/robustness_metrics`, 2020.

[8] Nathan Drenkow, Numair Sani, Ilya Shpitser, and Mathias Unberath. A systematic review of robustness in deep learning for computer vision: Mind the gap? *arXiv preprint arXiv:2112.00639*, 2021.

[9] Claudio Ferrari, Mark Niklas Muller, Nikola Jovanovic, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*, 2022.

[10] Karan Goel, Nazneen Rajani, Jesse Vig, Samson Tan, Jason Wu, Stephan Zheng, Caiming Xiong, Mohit Bansal, and Christopher Ré. Robustness gym: Unifying the nlp evaluation landscape. *arXiv preprint arXiv:2101.04840*, 2021.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[13] Jun Guo, Wei Bao, Jiakai Wang, Yuqing Ma, Xinghai Gao, Gang Xiao, Aishan Liu, Jian Dong, Xianglong Liu, and Wenjun Wu. A comprehensive evaluation framework for deep model robustness. *Pattern Recognition*, 137:109308, 2023.

[14] Yujie Ji, Xinyang Zhang, and Ting Wang. Backdoor attacks against learning systems. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2017.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *Advances in Neural Information Processing Systems 25*, 2012.

[16] 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.

[17] Linyi Li, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. In *2023 IEEE symposium on security and privacy (SP)*, pages 1289–1310. IEEE, 2023.

[18] Jing Lin, Long Dang, Mohamed Rahouti, and Kaiqi Xiong. Ml attack models: Adversarial attacks and data poisoning attacks. *arXiv preprint arXiv:2112.02797*, 2021.

[19] Xiang Ling, Shouling Ji, Jiaxu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. Deepsec: A uniform platform for security analysis of deep learning model. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 673–690. IEEE, 2019.

[20] 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.

[21] Marco Melis, Ambra Demontis, Maura Pintor, Angelo Sotgiu, and Battista Biggio. secml: A python library for secure and explainable machine learning. *arXiv preprint arXiv:1912.10013*, 2019.

[22] Matthew Mirman, Alexander Hägele, Pavol Bielik, Timon Gehr, and Martin Vechev. Robustness certification with generative models. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 1141–1154, 2021.

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[24] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020.

[25] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

[26] Marwan Omar, Soohyeon Choi, DaeHun Nyang, and David Mohaisen. Robust natural language processing: Recent advances, challenges, and future directions. *IEEE Access*, 2022.

[27] Alina Oprea and Apostol Vassilev. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations. Technical report, National Institute of Standards and Technology, 2023.

[28] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, Xiapu Luo, and Ting Wang. Trojanzoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 684–702. IEEE, 2022.

[29] Zoe Papakipos and Joanna Bitton. Augly: Data augmentations for robustness, 2022.

[30] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[31] Will Pearce and Ram Shankar Siva Kumar. Ai security risk assessment using counterfit. *Microsoft Security Blog*, 2021.

[32] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017.

[33] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020.

[34] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

[35] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *arXiv preprint arXiv:1905.07857*, 2019.

[36] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.

[37] Binghui Wang, Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of graph neural networks against adversarial structural perturbation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1645–1653, 2021.

[38] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. Robot: Robustness-oriented testing for deep learning systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 300–311. IEEE, 2021.

[39] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

[40] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

[41] Jiarong Xu, Junru Chen, Siqi You, Zhiqing Xiao, Yang Yang, and Jiangang Lu. Robustness of deep learning models on graphs: A survey. *AI Open*, 2:69–78, 2021.

[42] Yuanyuan Yuan, Shuai Wang, and Zhendong Su. Precise and generalized robustness certification for neural networks. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4769–4786, 2023.

[43] Mohammad Zarei, Chris Ward, Josh Harguess, and Marshal Aiken. Adversarial barrel! an evaluation of 3d physical adversarial attacks. In *2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–6. IEEE, 2022.

Trade-offs for Securing ML Systems

Yahya Al-Eryani
yahya.al-eryani@aalto.fi

Tutor: Samuel Marchal

Abstract

The growing integration of machine learning (ML) technology across various vital industry sectors emphasizes the importance of securing ML systems from emerging threats. This survey explores the security vulnerabilities of machine learning systems, including adversarial attacks, data poisoning, and model inversion, that pose a threat to their integrity and reliability. It highlights the advancement of defense mechanisms, such as adversarial training and differential privacy, to mitigate these threats. However, the process of enhancing security often involves compromises that could potentially impact the accuracy, privacy, and fairness of the systems. This paper investigates the trade-offs involved underscores the need for adopting a comprehensive security approach that takes into account the interplay between robustness and other critical system attributes.

KEYWORDS: Machine Learning Security, Adversarial Attacks, Data Poisoning, Model Inversion, Defense Mechanisms, Trade-offs, Adversarial Training, Differential Privacy, System Robustness, Privacy, Fairness, Accuracy

1 Introduction

ML systems have become integral to modern computing, revolutionizing various sectors and industries from healthcare to finance, and from autonomous vehicles to cybersecurity. However, with the growing integration of ML technologies into vital systems and applications, the potential impact of its security vulnerabilities becomes increasingly significant, and the importance of securing ML systems from malicious attacks becomes apparent.

Security in ML systems is a critical area of focus that includes a range of practices and technologies designed to protect ML models from various threats. A prominent example of such threats is adversarial attacks[15]. These attacks involve creating inputs specifically designed to confuse or deceive the model, causing it to make incorrect predictions or classifications. This type of threat can undermine the integrity and reliability of ML systems. Another significant threat arises from data poisoning attacks[5], where attackers inject malicious data into the training set, causing the model to learn incorrect patterns and behaviors. These attacks highlight the importance of robust data handling and processing mechanisms to prevent unauthorized access and manipulation of sensitive information. Likewise, model inversion attacks introduce another layer of vulnerability, as they exploit ML models to extract sensitive information from the training data, thereby compromising the data confidentiality[2].

In response to these security threats, a wide range of defense mechanisms and strategies has been developed, each targeting particular vulnerabilities. For instance, adversarial training is employed to enhance the robustness of ML models by training them with a variety of malicious inputs[16]. This approach trains models to recognize and accurately respond to deceptive attempts, thereby strengthening their security. Another defense mechanism is the use of differential privacy techniques[21], which train models in a manner that prevents the disclosure of sensitive information from their training data, thus protecting data privacy. Furthermore, robustness testing and formal verification are utilized to rigorously evaluate the reliability and security of ML systems against various threats[20]. Ultimately, the goal of these defensive technologies and approaches is to maintain the integrity, availability, and confidentiality of data and models[2], ensuring they operate as intended and deliver reli-

able and trustworthy outcomes.

Despite the development of advanced technologies to secure ML systems, research indicates that achieving security often involves trade-offs, compromising essential elements, such as accuracy, privacy, and fairness[13]. Building on this premise, this paper is structured as follows: Section 2 discusses ML threats and vulnerabilities, including the adversarial threat model, attacks on ML systems, and defense techniques and methodologies. Section 3 analyzes the trade-offs in securing ML systems, focusing on security versus privacy, security versus fairness, and security versus accuracy. Section 4 summarizes the main findings, analyzes the methodologies and their impact, and outlines future research directions. Section 5 presents the conclusion, bringing together the insights and implications of the paper.

2 ML Threats And Vulnerabilities

2.1 Adversarial Threat Model

The study of adversarial threats against ML systems has evolved to include a comprehensive framework that extends beyond the attacker's knowledge to involve their goals, capabilities, strategies, and the expected impact of their attacks. This approach, based on previous works in the field [18, 4, 3], provides an in-depth understanding of the strategies used by adversaries against ML systems.

A well-defined adversarial threat model has four components: the adversary's goal, knowledge, capability, and attacking strategy[18]. The goals range from launching indiscriminate attacks that disrupt model integrity with high false positive and false negative rates to targeted attacks for privacy breaches or data extraction.

Adversarial activities can be categorized into three main knowledge-based scenarios: *black-box, gray-box, and white-box* attacks[2]. In black-box attacks, adversaries lack knowledge of the model's internals but can still observe its outputs to craft deceptive inputs. Gray-box attacks represent a middle ground, where the adversary is aware of the model's architecture but lacks knowledge of its specific parameters. The most informed, white-box attackers, possess complete knowledge of the model, including its parameters, allowing for the most sophisticated and direct

attack strategies.

The adversary's capability reflects their control over training and testing data and is evaluated across several factors: the causal or exploratory nature of the attack, the percentage of data under their control, and their knowledge of model features and parameters[18]. Finally, the attacking strategy refers to the methods used to manipulate data, modify category labels, and tamper with features to achieve desired outcomes[18].

2.2 Attacks on Machine Learning Systems

The vulnerabilities of ML systems are most prominent during their training and testing phases. These stages are particularly susceptible to attacks that can diminish the performance and reliability of the systems. identifying and addressing vulnerabilities in ML systems is crucial given their vital significance in various applications. This section focuses on the potential threats posed to machine learning (ML) systems throughout the processes of training, testing, and inference, highlighting the need to implement robust security measures.

Training Phase Attacks

- **Poisoning Attacks**: These attacks involve the manipulation of the training data to diminish the efficacy of the machine learning model performance [5]. Such attacks are able to target a range of models, such as neural networks (NN), Support Vector Machines (SVM), and clustering algorithms by inserting adversarial samples (SVM)[11]. Adversarial samples are generated to create model misclassification or decline in performance which can significantly affect the reliability and accuracy of ML systems [5]. Hence, the insertion of these samples into the training dataset enables the poisoning attack without the alteration of the model itself [17]. Consequently, attackers have been utilizing advanced technologies such as Generative Adversarial Networks (GAN) to generate high-quality adversarial samples [27] to make the defense against these attacks more challenging [18].

- **Backdoor and Trojan Attacks**: These threats embed hidden malicious features into machine learning models during the training process, which are triggered by particular inputs during the inference phase. The utilization of this concealed strategy does not compromise the model's

efficacy when applied to usual inputs but rather guarantees intentional errors when exposed to manipulated inputs imposing notable security vulnerabilities [8, 26].

Testing and Inferring Phase Attacks

- **Adversarial Example Attacks**: The attacks were initially recognized by Szegedy et al.[23] and they involve the exploitation of vulnerabilities in the model by adding alterations to the input data, resulting in inaccurate model prediction. Deep Neural Networks (DNNs) are particularly vulnerable to malicious attacks in an array of applications, including image recognition and autonomous driving [11]. The vulnerability is increased by the transferability of adversarial examples across several models, which allows attackers to compromise systems even in the absence of accurate model knowledge[26, 23].

- **Evasion and Impersonation**: Attackers can avoid detection or impersonating legitimate users by creating inputs that are misclassified by the ML model. [18] has demonstrated evidence of these attacks in several domains, such as spam detection, malware identification, and facial recognition systems.

- **Inversion and Extraction Attacks**: These involve utilizing the model outputs to deduce sensitive information about the training set or to rebuild the model itself [11]. Inversion attacks impose substantial privacy risks, while extraction attacks could cause intellectual property theft and unauthorized replication of ML services [26].

2.3 Defense Techniques and Methodologies

The advancement in ML security has led to the development of resilient defensive mechanisms specifically designed to safeguard ML systems from a wide range of adversarial attacks. This section presents an overview of defensive strategies employed across the whole machine learning system lifetime. It outlines essential strategies for addressing potential risks, improving the protection of data, and preserving privacy, thereby strengthening the robustness of machine learning systems against various forms

of malicious attacks.

Security Assessment, Proactive and Reactive Mechanisms

Machine learning security involves the implementation of both proactive and reactive defense techniques. According to [18], Proactive defenses refer to identifying potential vulnerabilities through penetration testing before an attack occurs, whereas reactive defenses include evaluating the impact of attacks and updating the models accordingly. It indicates that the evaluation of the system security largely depends on these methods, which leverage the non-stationary data distribution induced by adversarial samples to predict and mitigate attacks. The methods employed include quantitative security analysis, evaluation within adversarial settings, and a particular focus on the difference in distribution between training and testing data when subjected to an adversarial attack.

Training Phase Countermeasures

In the training phase, it is crucial to prioritize the maintenance of data purity and the enhancement of learning algorithms' robustness. Effective countermeasures include techniques, such as data sanitization, which involves the separation and removal of malicious samples from training data, as well as strengthening algorithmic robustness through approaches, such as the Bootstrap Aggregating and Random Subspace Method (RSM)[**?**]. To protect ML models against poisoning and backdoor attacks, advanced techniques, such as secure SVM (Sec-SVM), are used to protect against feature manipulation and evasion attacks [9].

Testing and Inference Phase Countermeasures

During the testing or inference phase, the robustness of learning algorithms can be improved by applying game theory principles[6]. [18] states that this approach simulates the dynamic interactions between potential attackers and defenders, employing techniques such as Stackelberg Games and Nash-SVM algorithms, which are based on the notion of Nash equilibrium. Furthermore, it reports that the implementation of defensive distillation techniques improves the performance of deep neural networks (DNNs) in detecting adversarial samples, reducing false alarm rates. It further conveys that active defensive solutions, which involve retraining learning models using adversarial samples to meet testing data distribu-

tions, also play a crucial role in resisting testing phase attacks.

Data Security and Privacy

Securing and protecting data, particularly in the pre-training and training stages, involves the utilization of sophisticated cryptographic methods. In the context of training machine learning (ML) models, encryption techniques such as differential privacy (DP)[10] and homomorphic encryption (HE)[1] are employed to secure data both before and during its processing. This encryption ensures that the data remains usable for training purposes without any compromise. By including these methods at the early stages of the machine learning lifecycle, particularly before the commencement of training and throughout it, [18] confirms that we guarantee that potential attackers are unable to extract private user data from computational outcomes. It conveys that the use of this proactive strategy offers a strong defense mechanism against future privacy breaches and data recovery attacks.

Comprehensive Defense Frameworks

Wu et al.[24] emphasize that the defensive mechanisms span a wide range of strategies that can potentially be used at different stages of the model lifecycle. They imply that these strategies include algorithm-specific methods along with broader approaches that improve the overall resilience and security of ML models. During the training phase, methods, such as ANTIDOTE and Bagging Classifiers are used to identify outliers and address poisoning threats, and adversarial training to improve the model's robustness to adversarial examples [18]. Subsequently, in the post-training phase, Xue et al.[26] state that activation clustering and pruning-fine-tuning techniques are employed to enhance the model's ability to mitigate backdoor attacks. Furthermore, they indicate that secure aggregation protocols, such as Federated Learning, are essential for maintaining privacy in machine learning applications. These protocols are applied during the data aggregation phase, particularly in situations when model training is distributed. These examples emphasize the adaptability and wide range of security measures across the machine learning lifecycle [26].

3 Analyzing The Trade-offs in Securing ML Systems

Ensuring the security of machine learning systems is a complex task that requires a careful balance between robustness and other metrics of performance. In the process of enhancing defensive measures to safeguard these systems from adversarial attacks, it is important to take into account the potential impact on other essential attributes of the system. Security enhancements can inadvertently impact the transparency of the model's decision-making processes and the privacy of the data it handles. Furthermore, it is crucial to carefully consider the impact of countermeasures on the fairness and accuracy of the machine learning system. This section will explore the trade-offs associated with enhancing the security and resilience of ML systems, particularly on privacy, fairness, and accuracy.

3.1 Security Versus Privacy

Song et al.[22] assert that enhancing the security of machine learning models against adversarial examples tends to increase their vulnerability to privacy risks, particularly membership inference attacks. They indicate that the paradoxical effect emerges due to the inherent trade-off between the need to maintain model robustness and the urge to protect the privacy of training data. They justify by stating that in the case of adversarially robust models, the protection against adversarial inputs inadvertently amplifies the model's sensitivity to its training data. The amplified sensitivity results in an increased susceptibility to membership inference attacks, whereby an adversary can determine with certainty if an input was included in the model's training dataset.

[22] convey that robust models, which are specifically built to withstand attacks from adversaries, have a substantial gap in their ability to generalize. They point out that this gap is illustrated by a notable difference between their performance on training data, known as adversarial train accuracy, and their performance on unseen data, referred to as adversarial test accuracy. For instance, Robust CIFAR10 classifiers examined in the study exhibit significant divergence in their performance during training and testing when subjected to adversarial conditions. Specifically, the adversarial train accuracy is at 96%, while the adversarial test accuracy is only 47%. The presence of this gap suggests that the models have become more responsive to changes in the training data. This is

due to robust training techniques designed to ensure that prediction accuracy remains within a predetermined range, particularly for the training instances, as highlighted by [22].

[22] further indicates that the membership inference attacks leverage this sensitivity by exploiting the confidence with which models predict labels for inputs. it establishes that Robust models, which are trained to ensure consistent predictions for altered versions of training examples, inadvertently reveal more information about these examples compared to their non-robust counterparts. This is evident from the success of membership inference attacks, where the accuracy of such attacks significantly surpasses the baseline accuracy achievable through random guessing. This conveys that the attacks exploit both benign and adversarially altered inputs, with the latter providing a stronger basis for inference due to the models' attempts to maintain prediction confidence under disturbance.

Addressing the privacy risks introduced by robustness enhancements involves adopting countermeasures that mitigate the models' sensitivity to training data without compromising their defensive capabilities, as established by Song et al. They suggest that techniques, such as temperature scaling and regularization, aimed at improving robustness generalization offer potential pathways to balancing the security-privacy trade-off. Temperature scaling, for instance, reduces the prediction confidence disparity between training and test data, thereby obscuring the signals used by membership inference attacks. Regularization techniques focus on enhancing the model's generalization across adversarial conditions, thus narrowing the performance gap that facilitates membership inference [22].

3.2 security Versus Fairness

The challenging interplay between security and fairness in machine learning models, especially those trained on adversarial samples, underscores a complex trade-off. [25] highlights that adversarial training strategies, while intended to enhance the resilience of models, inadvertently aggravate the differences in accuracy and resilience across various classes or groups in the data. it states The concept of "robust fairness" highlights a fundamental conflict between the improvement of model security and ensuring equitable treatment across diverse data groups.

According to [25], Adversarial training methods, such as PGD (Pro-

jected Gradient Descent) and TRADES, despite their efficacy in improving model robustness, exhibit significant variances in performance across different classes. They illustrate that models trained on datasets, such as CIFAR10 and SVHN, exhibit substantial differences in both standard accuracy and robustness, with certain classes ("automobile" in CIFAR10) faring significantly better than others ("cat" in CIFAR10). This variance is not present in models trained without adversarial examples, indicating a unique challenge introduced by adversarial training techniques.

To mitigate this "robust fairness" issue, the introduction of the Fair-Robust-Learning (FRL) framework marks a pivotal step towards balancing the scales of security and fairness [7]. The FRL framework employs strategies such as "Reweight," "Remargin," and a combination of both, aiming to dynamically adjust training weights and adversarial perturbation margins for underperforming classes [25]. This approach seeks to minimize the disparity in performance, ensuring that models remain robust without compromising on fairness.

3.3 Security Versus Accuracy

Achieving an optimal balance between the security of machine learning models and their accuracy is a critical yet challenging aspect of designing ML systems. Recent research conducted by Zhuo et al.[28], has extensively examined and confirmed robustness against adversarial threats often coming at the cost of diminished model performance on legitimate inputs. They indicate that trade-offs are evident in several aspects, particularly in the model's reaction to adversarial examples, data manipulation attacks, and the overall integrity of the training process.

Adversarial examples exemplify the critical security challenges facing contemporary ML systems. Efforts to enhance model resilience to such attacks, as demonstrated through adversarial training techniques, can inadvertently lead to a degradation in accuracy on unperturbed data [11]. The phenomenon, as analyzed in the security-accuracy trade-off, highlights the careful balance required in model design—prioritizing adversarial robustness may result in models that are overly cautious or inaccurate in normal conditions.

The integrity of training data plays a crucial role in the model's eventual performance and security. Data poisoning attacks, wherein malicious data is introduced into the training set, present a direct threat to the model's accuracy by corrupting its learning process [5]. Defense

mechanisms against such attacks, including rigorous data validation and anomaly detection techniques, are essential but can also complicate the data processing pipeline and potentially exclude valuable, albeit anomalous, training examples [11].

The configuration of model hyperparameters emerges as a critical factor in navigating the security-accuracy trade-off. As detailed by [28], hyperparameter optimization (HPO) strategies, such as multi-objective optimization, offer a promising avenue to systematically explore the trade-off landscape. By simultaneously considering adversarial robustness and accuracy during the HPO process, it is possible to identify configurations that strike an optimal balance, tailored to the specific requirements of the application domain.

4 Discussion

The interconnections between security, privacy, fairness, and accuracy are not independent problems, but rather intricately linked, exerting substantial influence on each other. For instance, enhancing the robustness of models against adversarial threats may inadvertently risk privacy. This was shown in [12], where it was found that robust models became more sensitive to the training data, hence increasing the vulnerability to membership inference attacks. Similarly, efforts to secure privacy through differential privacy mechanisms can introduce disparities in model fairness and accuracy, as emphasized in [14]. The complexities mentioned highlight the need for a comprehensive strategy in designing machine learning systems that simultaneously considers these trade-offs.

Achieving a state of balance among these opposing objectives requires the implementation of sophisticated technologies and frameworks that enable precise management of the influence of security measures on privacy, fairness, and accuracy. One example that may be cited is federated learning, which is explored in [14]. Federated learning presents a potential solution for improving privacy while maintaining model fairness and accuracy [19]. This is achieved by sharing the learning process across multiple clients. Furthermore, the differential privacy mechanism described in [14] introduces a systematic approach to explicitly manage privacy trade-offs, providing a framework for effectively managing the balance between protecting privacy and ensuring fairness in the model.

To comprehensively address these trade-offs, it is essential to use frame-

works and techniques that explicitly include fairness and privacy issues in the process of machine learning development. The technique introduced in [14] offers a measurable method for assessing the fairness and privacy implications of machine learning models. This enables informed decision-making that effectively balances these factors. Moreover, the incorporation of adversarial training and differential privacy, as proposed in [12] offers a practical approach to improve the security of models while taking into account the implications on privacy and fairness.

Although there have been improvements in dealing with the compromises between security, privacy, fairness, and accuracy in machine learning systems, there is still a notable lack of research in creating complete frameworks that can thoroughly assess and alleviate these compromises in real-world scenarios. Contemporary approaches often address these concerns in a fragmented manner, resulting in solutions that may prioritize one attribute while neglecting others. Further studies should focus on the advancement of comprehensive frameworks that include these factors from their initial development, possibly employing multi-objective optimization methodologies to identify optimal balance.

Furthermore, it is essential to conduct further empirical research to get a comprehensive understanding of the real-world implications associated with these trade-offs, particularly in critical fields, such as healthcare, finance, and law enforcement. Exploring the impact of various defensive mechanisms in diverse circumstances will provide a more profound understanding of their effectiveness and constraints.

5 Conclusion

The mission of ensuring the security of machine learning systems reveals a complex set of compromises between security, privacy, fairness, and accuracy. While advancements in ML security technologies have significantly enhanced the resilience of these systems against malicious attacks, they have, consequently, revealed a new set of vulnerabilities and challenges. Achieving an optimal balance between these conflicting demands requires a sophisticated methodology that acknowledges the interdependence of these challenges. Further research should focus on the development of comprehensive frameworks that include security, privacy, fairness, and accuracy factors from the first stages. This will facilitate the building of ML systems that are not only secure and resilient but also

equitable, accurate, and privacy-preserving.

References

[1] Nawal Almutairi, Frans Coenen, and Keith Dures. K-means clustering using homomorphic encryption and an updatable distance matrix: Secure third party data clustering with limited data owner interaction. pages 274–285, 08 2017.

[2] Afnan Alotaibi and Murad A. Rassam. Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense. *Future Internet*, 15(2), 2023.

[3] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, May 2010.

[4] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):984–996, April 2014.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines, 2013.

[6] Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. New York, NY, USA, 2011. Association for Computing Machinery.

[7] Hongyan Chang and Reza Shokri. On the privacy risks of algorithmic fairness, 2021. doi: 10.48550/arXiv.2011.03731.

[8] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017.

[9] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! a case study on android malware detection, 2017.

[10] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[11] European Union Agency for Cybersecurity (ENISA). Securing machine learning algorithms. Technical report, European Union Agency for Cybersecurity (ENISA), December 2021. https://www.enisa.europa.eu/publications/securing-machine-learning-algorithms.

[12] Alex Gittens, Bülent Yener, and Moti Yung. An adversarial perspective on accuracy, robustness, fairness, and privacy: Multilateral-tradeoffs in trustworthy ml. *IEEE Access*, 10:120850–120865, 2022.

[13] Xiuting Gu, Zhu Tianqing, Jie Li, Tao Zhang, Wei Ren, and Kim-Kwang Raymond Choo. Privacy, accuracy, and model fairness trade-offs in federated learning. *Computers Security*, 122:102907, 2022.

[14] Xiuting Gu, Zhu Tianqing, Jie Li, Tao Zhang, Wei Ren, and Kim-Kwang Raymond Choo. Privacy, accuracy, and model fairness trade-offs in federated learning. *Computers Security*, 122:102907, 2022.

[15] Muhammad Irfan, Sheraz Ali, Irfan Yaqoob, and Numan Zafar. Towards deep learning: A review on adversarial attacks. pages 91–96, 04 2021.

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale, 2017.

[17] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering, 2016.

[18] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor C. M. Leung. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE Access*, 6:12103–12117, 2018.

[19] Priyanka Mary Mammen. Federated learning: Opportunities and challenges, 2021.

[20] Mark Huasong Meng, Guangdong Bai, Sin Gee Teo, Zhe Hou, Yan Xiao, Yun Lin, and Jin Song Dong. Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2022.

[21] Valentin Mulder and Mathias Humbert. *Differential Privacy*, pages 157–161. Springer Nature Switzerland, Cham, 2023.

[22] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. CCS '19, page 241–257, New York, NY, USA, 2019. Association for Computing Machinery. doi: 10.1145/3319535.3354211.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[24] Baoyuan Wu, Shaokui Wei, Mingli Zhu, Meixi Zheng, Zihao Zhu, Mingda Zhang, Hongrui Chen, Danni Yuan, Li Liu, and Qingshan Liu. Defenses in adversarial machine learning: A survey, 2023.

[25] Han Xu, Xiaorui Liu, Yaxin Li, Anil K. Jain, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training, 2021. doi: 10.48550/arXiv.2010.06121.

[26] Mingfu Xue, Chengxiang Yuan, Heyi Wu, Yushu Zhang, and Weiqiang Liu. Machine learning security: Threats, countermeasures, and evaluations. *IEEE Access*, 8:74720–74742, 2020.

[27] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. Generative poisoning attack method against neural networks, 2017.

[28] Yue Zhuo, Zhihuan Song, and Zhiqiang Ge. Security versus accuracy: Trade-off data modeling to safe fault classification systems. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2023.

Ad-hoc Cloud: A User-Provided Cloud Infrastructure at Network Edge

Huang Yaojun
yaojun.huang@aalto.fi

Tutor: Sara Ranjbaran

Abstract

As demands for processing complex tasks continue to rise and device capabilities continue to advance, resource sharing and collaboration among user devices become increasingly critical. In this scenario, traditional computing paradigms face challenges in effectively utilizing resources. Ad-hoc Cloud is a novel user-provided infrastructure concept that leverages existing and non-exclusive edge devices to achieve resource sharing. This paper provides an overview of Ad-hoc Cloud Computing, exploring various implementations, and highlighting their contributions to optimizing resource utilization and enhancing user experience. Additionally, the paper concludes several challenges and future research directions in Ad-hoc Cloud Computing, including user incentives, task delegation, and security. Despite being in its early stages, Ad-hoc Cloud Computing shows promise in supplementing existing computing paradigms.

KEYWORDS: User-Provided Cloud Infrastructure, Ad-hoc Cloud Computing, Edge Computing, Task offloading

1 Introduction

Although hardware technology continues to advance, individual machines still cannot keep up with the increasing demands of processing complex tasks. Cloud Computing (CC), a novel computing model [1], emerges as a solution to this challenge by revolutionizing the delivery of applications over the Internet. It offers virtually unlimited computing resources that can be provisioned on-demand, empowering users with unprecedented flexibility [2]. Mobile Cloud Computing (MCC) is a paradigm that combines Mobile Computing and Cloud Computing [3]. By outsourcing computing tasks to data centers, thin devices can deliver interactive experiences that exceed local capabilities. However, this model, while powerful, comes with a drawback: high latency.

Edge Computing and Fog Computing are models proposed to maximize the potential of Cloud Computing. These models aim to decentralize computing power and bring it closer to the user to reduce data transfer latency and enhance service responsiveness. Multi-Access Edge Computing (MEC) [4], seeks to improve application performance and user experience by processing relevant tasks in a place closer to users. However, compared to the centralized managed data centers, the decentralized management at the edge exacerbates the deployment challenges of edge servers. One of the challenges that prevent the feasibility of such a model is who will deploy such infrastructure.

The concept of User-Provided Infrastructure (UPI) has been developed to optimize system efficiency and utilize the potential of network edges. UPI makes use of idle heterogeneous edge devices that have increasing computing power to supplement edge computing capabilities. User devices cooperate to perform complex tasks beyond the capabilities of a single machine. Ad hoc Cloud Computing [5]. is one way to achieve UPI. It is a local cloud that utilizes computing resources from existing, non-exclusive, and unreliable infrastructures.

To evaluate the practicality of Ad-hoc Cloud Computing, this paper discusses the characteristics of Ad-hoc Cloud and reviews various Ad-hoc Cloud frameworks, attempting to summarize the challenges and directions in this area.

The paper is structured as follows. Section 2 provides an overview of the computing paradigms. Section 3 explores various research related to Ad hoc Cloud architecture. Section 4 discusses the approaches outlined in

this paper. Finally, Section 6 concludes the paper.

2 Computing paradigm

Cloud computing is a model that offers access to a shared pool of computing and storage resources on-demand, ensuring scalability and accessibility from any location [2]. However, it faces significant challenges due to the geographical centralization of computing resources in distant data centers. This structure introduces transmission delays across the network, resulting in bottlenecks that are particularly challenging for latency-sensitive applications such as Augmented Reality and Cloud Gaming.

This section introduces Fog Computing and Edge Computing, two main computing paradigms that try to alleviate the problems of Cloud Computing and task offloading, the main approach to utilize the computing paradigm.

2.1 Fog Computing

Fog Computing (FC) was introduced by Cisco in 2012 as a response to the limitations of Cloud Computing [6]. It acts as an intermediary platform between conventional data centers and end devices, providing computing and storage services. Essentially, it represents a "descending" cloud that is closer to end-users.

Fog computing has several advantages over Cloud Computing, such as reduced bandwidth consumption, location awareness, low latency, and mobility support. The architecture of fog computing is typically geographically distributed and consists of a diverse array of heterogeneous devices with varying capabilities [7]. These unique properties enable the construction of large-scale automated systems, making fog computing a highly efficient solution for Internet of Things (IoT) implementations. FC has found widespread applications in areas such as smart cities.

2.2 Edge Computing

Edge Computing (EC) is another concept to extend the Cloud Computing paradigm. It is a distributed computing framework that leverages edge devices for data processing instead of offloading tasks solely to the cloud. By processing and analyzing data in place only one jump to its

source, Edge Computing significantly reduces latency that occurs when data is transmitted to distant cloud centers, thus achieving shorter response times and enhanced user experience [8]. Edge Computing is not intended to replace cloud computing but rather acts as a supplement. By bringing computational capabilities to the network edge, it eliminates bottlenecks inherent in centralized architectures, thereby supporting the cloud computing experience. Multi-Access Edge Computing (MEC) [4] is an architecture defined by the European Telecommunications Standards Institute (ETSI). MEC provides cloud computing capabilities and IT services at the network edge, offering features such as on-premises deployment, lower latency, and location awareness [8]. The advantages of Edge Computing make it an essential solution for a variety of latency-sensitive applications, establishing its critical role in the fields of multimedia entertainment and smart homes.

2.3 Task offloading

Compared to data centers, end devices typically have limited computing power and may not be able to perform resource-intensive computing tasks locally, which prompts the adoption of task-offloading strategies. Task offloading involves delegating computing tasks to other nodes, aiming to reduce processing time, but potentially increase transmission time. There are three types of task offloading: local execution, partial offloading to other capable devices, and full remote computing [9]. The specific task offloading strategy chosen depends on various factors, such as the nature of the tasks, the characteristics of the processing devices, and optimization objectives like energy efficiency, latency, cost, security, and Quality of Experience [10].

Mobile Cloud Computing architectures involve processing tasks generated by end devices either locally or by sending them to cloud centers for processing. With advancements in hardware and the advent of 5G technology, end devices now have increased computing capabilities and reduced communication latency which allows tasks to be offloaded to the cloud or neighboring end devices for direct collaboration, known as Device-to-Device (D2D) Task Offloading. This approach increases flexibility and leads to a more diverse architecture. However, the joining of numerous heterogeneous devices distributed across different layers of the architecture makes the optimization problem intricate. Various methods have been proposed for optimized task scheduling, which can be catego-

rized into three types: (a) Mathematical Optimization algorithms; (b) Machine learning algorithms; and (c) Control Theory-based approaches [11].

In the scenario of the Internet of Things (IoT), edge devices often exhibit different capabilities, so collaborative efforts are essential to efficiently execute tasks. For example, a camera may be tasked with image collection while a phone can be responsible for processing. By offloading tasks to other devices, end devices can work together to complete more complex tasks with shorter execution and transmission times.

3 Ad-hoc Cloud Computing

Ad hoc Cloud is an emerging research direction. It refers to a computational model that leverages existing, non-exclusive, and unreliable infrastructures to acquire computing resources [5]. Originating from the concepts of volunteer and grid computing, Ad hoc Cloud has the key following difference. It operates as a local cloud consisting of volunteer resources, eliminating the need for trust between users and infrastructure. Moreover, it ensures job continuity and minimizes interference with the host, adept at handling diverse workloads.

Ad hoc Cloud typically comprises a group of nearby mobile devices willing to share resources, alleviating the computational bottleneck at the edge server. Additionally, it provides the capability to execute compute-intensive applications locally when remote network connections encounter issues [12]. Ad hoc Cloud facilitates a cloud-style paradigm within the local network, effectively utilizing idle computing resources in the local network environment, thereby enhancing device utilization and reducing energy consumption [13].

This section presents a variety of implementations and applications of Ad hoc Cloud.

3.1 Dynamic Mobile Cloud Computing

Dynamic Mobile Cloud Computing [14] is a local cloud framework designed to optimize resource utilization in mobile computing. It includes three key components: resource handler, job handler, and cost handler. The resource handler is responsible for tasks such as resource discovery, monitoring, and metadata exchange. The job handler is tasked with managing jobs and scheduling while the cost handler estimates costs and han-

dles transactions between client devices and the master device. In this framework, the master device initiates device discovery to identify potential clients. Subsequently, considering factors such as user priorities, requirements, and constraints, the framework calculates costs to select the optimal node for task delegation. Before or after job delegation, the master device pays the client device for its resources. The experiments use Bluetooth as the transmission solution and show that the offloading strategy can always provide better performance.

3.2 Transient Clouds

Transient Clouds [15] is a temporary network that forms dynamically among nearby devices, allowing them to act as a cloud platform and share their resources. A transient cloud is established when devices come together and dissipates when they leave. Resources shared in the Transient Cloud include general computing capabilities such as CPU and storage, and heterogeneous capabilities such as sensing and localization. Devices within the network submit tasks to the Transient Cloud, which then distributes the tasks based on the device's capabilities. The transient cloud employs a modified version of the Hungarian method, which dynamically adjusts the cost based on previous assignments to achieve load balancing. In the experimental simulation results using Wifi Direct and Android, Transient Clouds outperforms standalone execution when the ratio of computation to transmission size is high. This highlights the potential for collaborative resource sharing among users.

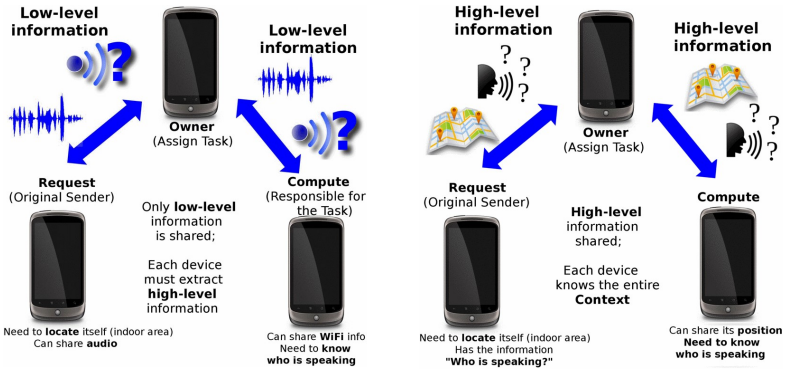


Figure 1. Comparison between traditional (left) and Context-Awareness (right) Transient Clouds [16]

Sciarrone et al. enhanced Transient Clouds by integrating the high-level information: context and introduced the concept of the Transient Context-Aware Cloud (TCAC) [16]. Within TCAC, devices share high-level

information instead of low-level capabilities. This approach can significantly reduce the workload on the devices. For instance, if a device possesses knowledge of its position, it can share this information with other devices, thus obviating the continuous same task delegation to devices equipped with GPS functionality.

3.3 Ad-hoc Cloud Implementation

The paper [5] presents a comprehensive, integrated, and end-to-end Ad hoc Cloud solution, with an implementation based on BOINC. The overall architecture of the Ad-hoc Cloud is shown in Figure 2. It consists of cloudlets and ad hoc guests that utilize existing idle resources from host users to allocate them to tasks submitted by cloud users. An ad hoc client installed on the host manages the resource load and provides feedback on the host information to the server. This implementation follows the concept of Platform as a Service (PaaS), which ensures that the platform details are transparent to cloud users. The framework employs reliability as a metric for task scheduling, which is calculated by the previously assigned and completed jobs, host and guest failures, and resource load. To enhance system robustness, snapshots are periodically synchronized to potential backup machines, and in the event of a fault, one of the nodes is instructed to restore the snapshot for fault recovery.

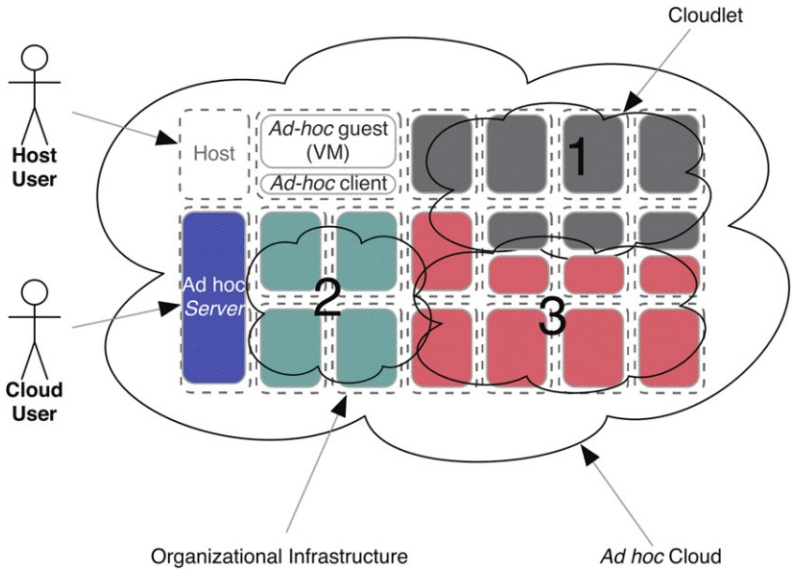


Figure 2. The Ad-hoc Cloud architecture [5]

Chi and Wang [17] extend the concept of Ad hoc Clouds into the domain of cloud gaming, introducing a progressive game resource download mechanism. In this framework, each end client possesses a copy of the

initial game state and gradually downloads additional content from the cloud or nearby peers as users progress through the game. In addition, a collaborative task mechanism is employed to dynamically distribute game components across different devices. The ad-hoc cloudlet is leveraged for task execution initially, but when its capabilities are exceeded, tasks are transferred to the cloud. This approach optimizes resource utilization and effectively reduces energy consumption.

3.4 Crowdsourced Edge Computing

Crowdsourced Edge Computing (CEC) [18] extends the concept of Ad-hoc Cloud Computing to a larger scale at the network edge. CEC operates in an ad hoc manner, acting as a distributed computing framework where all peers are considered equal. Roles within the network are divided into three categories: Task handlers, Workers, and Message brokers, and each node takes on one or more of these roles. Compared to the prevailing commercial driving force of mainstream edge computing paradigms, the CEC model places a stronger emphasis on community and voluntary participation. In CEC, devices contribute computing resources based on the desires of individuals or communities. This vendor-neutral service fosters the creation of more sustainable communities and can lead to entirely new forms of digital economies. Such democratized resource sharing facilitates decentralization and holds significant potential for application in the IoT area. However, this appealing computing paradigm is still in its nascent stage of development and faces significant challenges in areas such as Quality of Service, data consistency and security, and resource management.

4 Discussion

Ad-hoc Cloud emerges as a promising frontier aimed at optimizing resource utilization and mitigating deployment challenges inherent in edge and fog computing. Despite various explorations in this domain, the paradigm is still in its infancy and awaits further investigations to reveal its full potential. The following are some research trends and directions:

1. User Incentives Mechanism: While conventional architectures rely on monetary incentives to drive users to join the network, recent research

has started to explore the integration of social computing. These novel approaches aim to encourage users to voluntarily contribute their devices, fostering a more sustainable ecosystem.

2. Task Delegation and Load Balancing: Task scheduling approaches have evolved from simple indiscriminate task offloading to account for the characteristics of heterogeneous devices, and further to a comprehensive task distribution that takes into account both device capabilities and load balancing. This evolution is crucial for the extension of Ad-hoc Cloud to the context of IoT environments, where collaboration among diverse devices is essential for efficient operation.

3. Scenario Analysis and Data Sharing: By analyzing tasks and data flow in practical applications, devices can be instructed to share contexts and individual information. This enables the Ad-hoc Cloud to make more optimal decisions at macroscopic scales, reducing redundant computations and data transmissions, which further optimizes resource utilization and reduces bandwidth consumption.

However, despite advances in areas such as network formation and task scheduling, Ad-hoc Cloud still faces significant challenges in terms of security and privacy. Ad-hoc Cloud does not assume the trustworthiness of devices within the network, allowing any node to join or leave the network at any time. This architectural flexibility makes it highly vulnerable to malicious attacks, including network contamination and data leakage. For Ad-hoc Cloud to be deployed effectively in the IoT domains such as smart cities, it is imperative to ensure the protection of participants' data. Therefore, further research in this direction is necessary to achieve a reliable and secure implementation of Ad-hoc Cloud.

5 Conclusion

This paper provides an overview of different implementations of the Ad-hoc cloud computing paradigm. Ad-hoc Cloud is a new model that uses idle resources to redefine edge computing infrastructures. The paper introduces two early-stage frameworks: Dynamic Mobile Cloud Computing and Transient Clouds, and presents a detailed implementation of Ad-hoc Cloud. It also discusses the concept of Crowdsourced Edge Computing.

In addition, the paper evaluates the current achievements of these approaches and identifies several future research trends, including the user incentive mechanism, task delegation, data sharing and security. Addressing these challenges has the potential to facilitate the utilization of Ad-hoc Cloud and transform the deployment paradigm of edge computing.

References

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. Technical report, 2009.

[2] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011. doi: 10.6028/NIST.SP.800-145.

[3] Atta ur Rehman Khan, Mazliza Othman, Sajjad Ahmad Madani, and Samee Ullah Khan. A survey of mobile cloud computing application models. *IEEE Communications Surveys Tutorials*, 16(1):393–413, 2014. doi: 10.1109/SURV.2013.062613.00160.

[4] Sami Kekki, Walter Featherstone, Yonggang Fang, Pekka Kuure, Alice Li, Anurag Ranjan, Debashish Purkayastha, Feng Jiangping, Danny Frydman, Gianluca Verin, et al. Mec in 5g networks. *ETSI white paper*, 28(2018):1–28, 2018.

[5] Gary A. McGilvary, Adam Barker, and Malcolm Atkinson. Ad hoc cloud computing. In *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, June 2015. doi: 10.1109/CLOUD.2015.153.

[6] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery. doi: 10.1145/2342509.2342513.

[7] 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. doi: 10.1109/SMC.2016.7844455.

[8] Vasco Pereira Jorge Bernardino Gonçalo Carvalho, Bruno Cabral. Edge computing: current trends, research challenges and future directions. *Computing*, 103:993–1023, 2021. doi: 10.1007/s00607-020-00896-5.

[9] Junna Zhang and Xiaoyan Zhao. An overview of user-oriented computation offloading in mobile edge computing. In *2020 IEEE World Congress on Services (SERVICES)*, pages 75–76, 2020. doi: 10.1109/SERVICES48979.2020.00029.

[10] S. Taheri-abed, A.M. Eftekhari Moghadam, and M.H. Rezvani. Machine learning-based computation offloading in edge and fog: a systematic review. *Cluster Computing*, 26:3113–3144, 2023. doi: 10.1007/s10586-023-04100-z.

[11] Firdose Saeik, Marios Avgeris, Dimitrios Spatharakis, Nina Santi, Dimitrios Dechouniotis, John Violos, Aris Leivadeas, Nikolaos Athanasopoulos, Nathalie Mitton, and Symeon Papavassiliou. Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions. *Computer Networks*, 195:108177, 2021. doi: 10.1016/j.comnet.2021.108177.

[12] Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, and Sghaier Guizani. Mobile ad hoc cloud: A survey. *Wireless Communications and Mobile Computing*, 16(16):2572–2589, 2016. doi: 10.1002/wcm.2709.

[13] Graham Kirby, Alan Dearle, Angus Macdonald, and Alvaro Fernandes. An approach to ad hoc cloud computing, 2010. doi: 10.48550/arXiv.1002.4738.

[14] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pages 281–286, 2011. doi: 10.1109/UCC.2011.45.

[15] Terry Penner, Alison Johnson, Brandon Van Slyke, Mina Guirguis, and Qijun Gu. Transient clouds: Assignment and collaborative execution of tasks on mobile devices. In *2014 IEEE Global Communications Conference*, pages 2801–2806, 2014. doi: 10.1109/GLOCOM.2014.7037232.

[16] Andrea Sciarrone, Igor Bisio, Fabio Lavagetto, Terrence Penner, and Mina Guirguis. Context awareness over transient clouds. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–5, 2015. doi: 10.1109/GLOCOM.2015.7417785.

[17] Fangyuan Chi, Xiaofei Wang, Wei Cai, and Victor C.M. Leung. Ad hoc cloudlet based cooperative cloud gaming. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 190–197, 2014. doi: 10.1109/CloudCom.2014.112.

[18] Stéphane Kündig, Constantinos Marios Angelopoulos, Sanmukh R. Kuppannagari, José Rolim, and Viktor K. Prasanna. Crowdsourced edge: A novel networking paradigm for the collaborative community. In *2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 474–481, 2020. doi: 10.1109/DCOSS49796.2020.00080.

User-Provided-Infrastructure at the Edge

Yinan Hu
yinan.hu@aalto.fi

Tutor: Sara Ranjbaran

Abstract

In recent years, the number of user devices has been increasing rapidly, especially in scnarios of the Internet of Things (IoTs). To meet the demand for computation and connectivity of these user devices, fog computing has been proposed. Fog computing enables user devices at the edge of networks to serve as hosts which provide computing resources to other devices in the vicinity, thus supporting mobility and heterogeneity in scenarios of IoTs compared to cloud computing. This paper introduces fog computing from the perspectives of its characteristics and architecture. Furthermore, this paper reviews the major research challenges of fog computing and discusses the proposed solutions. It will be shown that fog computing will be feasible in IoTs with more research conducted to provide solutions that achieve higher scalability.

KEYWORDS: Fog computing, IoTs, Edge computing

1 Introduction

With the rapid growth of demand for resources of servers, storage and various services, traditional forms of computing, which refer to conducting data processing and storing using local computing resources, cannot

meet people's needs to a large extent. In response to this, cloud computing, featuring a pay-as-you-go mode, enables organizations and individuals to have on-demand access to a shared pool of computing resources and pay only for what they consume [1]. However, cloud computing requires sending data to remote data centers for processing, which cannot support mobility in practical scenarios, such as the Internet of Vehicles. Furthermore, the design of cloud computing does not consider the heterogeneity of user devices, which limits its application in the scenarios of the Internet of Things (IoTs) where advanced handheld devices are increasing rapidly.

To address these limitations, the concept of fog computing has been proposed. As an extension of the cloud, the fog refers to a paradigm where heterogeneous user devices at the edge of networks can act as hosts to pool their computing resources, such as storage and communication, to match wide-ranging user requirements [1, 2].

The practical fog computing platform is a middle layer between the end devices and traditional cloud data centers, deploying resources of computation, storage and communication for nearby users [6]. Compared to the cloud, the fog can provide seamless services to users and collaborate with cloud data centers in the remote to carry out a robust performance, which emphasizes the strength of the fog in the domain of IoTs.

This paper reviews the paradigm of fog computing and its architecture. In addition, this paper reviews the main issues of fog computing by categorizing them into network management, resource management and incentivization. This paper also reviews the solutions to the issues.

This paper is organized as follows. Section 2 explains the motivation of fog computing and its strengths over cloud computing. Section 3 presents the typical 3-tier architecture of fog computing. Section 4 reviews the major challenges in fog computing and the proposed solutions to them. Section 5 discusses the feasibility and effectiveness of the existing solutions. Finally, Section 6 provides concluding remarks.

2 Characteristics of Fog Computing

Fog computing was first proposed by Cisco [1] to address the limitations of cloud computing in handling latency-sensitive applications. By introducing computing nodes in the vicinity of user devices, fog computing meets the delay requirement of clients or users. In addition, fog computing shows strengths in other aspects over cloud computing, such as

low latency, support for mobility and decentralized data processing.

2.1 Low latency

In data processing, systems in fog computing architecture can timely respond and conduct tasks [1, 2]. By extending the computing resources to the edge of networks, fog computing decreases the distance of data transmission, thus lowering the communication delay. The low-latency characteristic enables fog computing to have a significant performance in applications that have a high real-time requirement, such as IoTs, Smart Grid and Automation. In these applications, the fog can process data at a comparatively higher speed and provide real-time decisions and responses, which enhances the performance of systems and the quality of user experience.

2.2 Support for heterogeneity

Supporting for heterogeneity refers to the ability of the fog to effectively integrate and orchestrate devices with different types and performance [1]. These devices can have different hardware architectures, computing capabilities and storage capacities. Compared to cloud computing, which was initially not designed to support heterogeneity, the fog is capable of coordinating the jobs of heterogeneous devices, allowing these devices to collaborate to conduct tasks [7]. This support for heterogeneity enables systems of fog computing to be flexible, leveraging the strengths of various devices, thus adapting to different computing demands and environments, which is a crucial characteristic in IoTs where the number of advanced devices is increasing rapidly.

2.3 Support for mobility

Support for mobility refers to the capability of the fog to provide seamless services as users move [1, 2]. This characteristic allows users to constantly access and use computing resources on the move, without interrupting or rebuilding connection. Fog computing leverages deploying computing resources at the edge of networks to enable the resources to be located near the users, thus offering uninterrupted services. This support for mobility is important for the application of fog computing in specific scenarios, such as the Internet of Vehicles.

2.4 Decentralized data processing

Data processing of cloud computing depends on cloud data centers, which causes problems, such as trade-off issues between time and bandwidth [2]. Compared to this architecture, fog computing has a characteristic of decentralized data processing, i.e., data can be processed at the edge of networks, in the vicinity of data sources, which improves the efficiency of processing data, decreases the response time and reduces the dependencies on high bandwidth. In addition, decentralized data processing can optimize the utilization of computing resources. By distributing computing nodes in a wide range of locations, fog computing systems can process data more efficiently.

3 Architecture of Fog Computing

The prevalent structure of fog computing in existing research follows a three-tiered architecture, in which the fog layer lies between the cloud layer and the device layer. Sun and Zhang [6] describe the architecture of fog computing as a human neural network. Figure 1 shows the architecture, which is comprised of the brain nerve center, spinal nerve center and peripheral nerves, distributed over the body. The brain nerve center refers to the cloud data center. The spinal nerve center refers to the fog computing data center. The peripheral nerves refer to the user devices. This human neural network architecture explains the relationship among the cloud data center, the fog data center and user devices.

Device layer A wide range of user devices, such as sensors, IoT devices, smart phones, are located on this layer and connected to the fog layer. These devices produce data and communicate with the computing nodes of the fog layer.

Fog layer Fog layer serves as the central computing and data processing layer. The computing nodes of the fog layer are distributed at the edge of networks, responsible for receiving, processing and storing data transmitted from the device layer. These nodes can conduct computing jobs, optimize data transmission and provide timely response with low latency. The major goal of the fog layer is to locate the computing resources in the vicinity of user devices, thus offering seamless services and enhancing user experience.

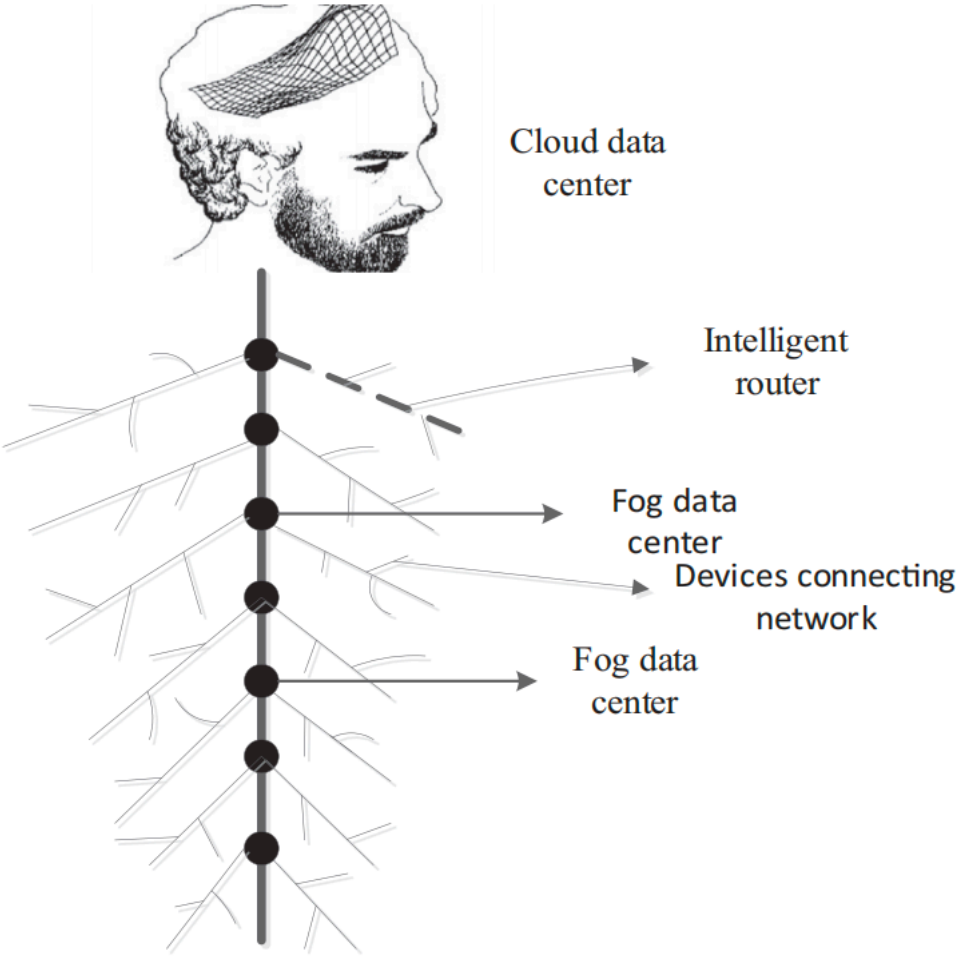


Figure 1. Architecture of fog computing based on the nervous system [6]

Cloud layer Cloud layer is the highest layer in the architecture, and consists of the cloud data center. The cloud layer is responsible for conducting tasks requiring larger computing capability and storing data on a larger scale. The cloud layer collaborates with fog layer to optimize the overall performance and improve the scalability of the architecture.

4 Research Challenges

The major research challenges of fog computing involve resource management, incentivization and network management [2]. Currently, researchers have been approaching these issues and proposing new solutions to them.

4.1 Resource management issue

The resource management issue of the fog refers to the optimization of resource utilization, which focuses on designing effective scheduling mechanisms and facilitating resource sharing among heterogeneous devices.

As a solution to resource management issue, Mobile-IoT-Federation-as-a-Service (MIFaaS) proposed by Farris et al. [3] realizes the federation of different mobile IoT devices and reduces the cost of deploying data centers on the network edge. To be more specific, MIFaaS utilizes a coalition formation game model to enable the IoT Cloud Providers (ICPs) to converge to a federation based on their own interests.

To efficiently utilize the resources among heterogeneous devices, Tang et al. [7] propose a general framework that enables mobile devices to collaborate and share three types of resources, including communication, computation and storage. This framework can be applied to various device-to-device resource-sharing models. In parallel, candidate algorithms are utilized to address both the offline and online optimization issues under the framework.

To provide users with the computing, storage and communication resources efficiently, Kiani and Ansari [5] design a novel hierarchical model by considering the principles of LTE Advanced backhaul network and introducing concepts of field, shallow, and deep cloudlets. In addition, a two-time scale mechanism is proposed to achieve the efficient allocation of resources.

4.2 Incentive issue

The incentive issue of fog computing refers to motivating end users to participate in resource sharing, addressing challenges of encouraging collaboration in the fog computing paradigm.

To address the incentive issue, Sun and Zhang [6] propose an incentive mechanism involving reward and punishment. The goal of the mechanism is to encourage resource owners to participate in the public pool and share their resources. Furthermore, the mechanism is responsible for supervising resource supporters and ensures their active participation in tasks.

Inspired by effective business models, Iosifidis et al. [4] introduce two types of innovative User-Provided-Infrastructure (UPI) models that focus on network sharing. For each model, Iosifidis et al. [4] investigate the essential features required by an effective incentive mechanism and

propose a potential solution. Furthermore, they introduce a simplified sharing algorithm to achieve a Pareto optimal and incentive-compatible sharing equilibrium, which is suitable for large-scale IoT systems.

4.3 Network management issue

The network management issue of fog refers to managing and optimizing the communication between the user devices at the edge, with the goal of enhancing the network-related quality of experience. More specifically, it includes various types of tasks, such as the design of network topology, selection of communication protocols and management of network traffic.

A network slicing approach proposed by Theodorou and Xezonaki [8] serves as a solution to IoT Infrastructure shareability. The approach exploits 5G Network Slicing and virtualization technologies to realize the division of network traffic and manage the access to services from IoT devices.

5 Discussion

Existing solutions to fog computing mainly focuses on scheduling resources [3, 7, 5], managing networks [8] or motivating users [6, 4]. The feasibility and effectiveness of the solutions in practical scenarios depend on various factors, such as security, scalability, and generality.

In terms of resource management, the general framework proposed by Tang et al. [7] offers a more flexible solution compared with others, by considering three different types of resources. Therefore, the framework can address the heterogeneity of user devices and be applied to real IoT scenarios where the types of devices are increasing rapidly.

With the goal of incentivization, researchers have proposed various reward and penalty mechanisms. Sun and Zhang [6] design incentive schemes for various scenarios, including Provider-assisted Mobile User-Provided Infrastructures (UPIs), Autonomous Mobile UPIs and Large-Scale Systems. Therefore, among the existing solutions, the solution proposed by Sun and Zhang has the best generality and scalability.

Further research can be conducted to validate the feasibility of the existing solutions and the application of the fog to different real-world scenarios. In parallel, more standards and criteria are needed to evaluate the efficiency of the solutions.

6 Conclusion

Fog computing is expected to play a crucial role in IoTs where user devices are heterogeneous. Collaborating with remote cloud data centers, fog data centers can serve as hosts and provide seamless services to users at the edge, thus improving both the quality of experience and robustness of systems. This paper introduces the characteristics and architecture of fog computing. Furthermore, this paper reviews the major research challenges and the proposed solutions. Finally, this paper discusses the feasibility and effectiveness of the solutions.

References

[1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 1316, New York, NY, USA, 2012. Association for Computing Machinery.

[2] Mung Chiang. Fog networking: An overview on research opportunities. *arXiv preprint arXiv:1601.00835*, 2016.

[3] I. Farris, L. Militano, M. Nitti, L. Atzori, and A. Iera. Mifaas: A mobile-iot-federation-as-a-service model for dynamic cooperation of iot cloud providers. *Future Generation Computer Systems*, 70:126–137, 2017.

[4] George Iosifidis, Lin Gao, Jianwei Huang, and Leandros Tassiulas. *Incentive Schemes for UserProvided Fog Infrastructure*, pages 129–150. 2020.

[5] Abbas Kiani and Nirwan Ansari. Toward hierarchical mobile edge computing: An auction-based profit maximization approach. *IEEE Internet of Things Journal*, 4(6):2082–2091, 2017.

[6] Yan Sun and Nan Zhang. A resource-sharing model based on a repeated game in fog computing. *Saudi Journal of Biological Sciences*, 24(3):687–694, 2017. Computational Intelligence Research & Approaches in Bioinformatics and Biocomputing.

[7] Ming Tang, Lin Gao, and Jianwei Huang. Communication, computation, and caching resource sharing for the internet of things. *IEEE Communications Magazine*, 58(4):75–80, 2020.

[8] Vasileios Theodorou and Maria-Evgenia Xezonaki. Network slicing for multi-tenant edge processing over shared iot infrastructure. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 8–14, 2020.

Exploring data-driven optimal ventilation control using CO2 concentration monitoring and control

Yinda Xu
yinda.xu@aalto.fi

Tutor:Matti Huotari

Abstract

As many people spend the majority of their time indoors, the energy consumed by buildings for heating, ventilation, and air conditioning (HVAC) is a key contributor to global energy use. It is evident that balanced CO2 levels are also one of the IAQ factors, as it has pronounced effects on occupant health and performance. This paper presents a comparative review between traditional ventilation control techniques and data-driven methods like ML and CNN, as well as advanced techniques in AI such as LSTM and RL. Thus, the proposed AI-based models seem to be efficient enough to dynamically regulate the ventilation rates and optimize energy consumption while maintaining satisfactory IAQ levels.

KEYWORDS: CO2 concentration, HVAC system, Optimal control, data-driven

1 Introduction

In today's world, a significant part of the population spends a considerable amount of their time inside buildings, such as offices, homes, or other facilities [11]. These environments are tailored to offer comfort regardless of the outside climate. Yet, it's essential to recognize that these struc-

tures play a substantial role in the overall energy consumption, making up about 30% to 40% of the global energy use [17]. A closer look at the energy use in buildings indicates that heating, ventilation, and air conditioning (HVAC) systems are responsible for almost half of this consumption [26].

Interestingly, despite the high energy use, dissatisfaction with indoor environmental conditions persists among occupants. Studies highlight the significant influence of indoor environmental factors such as air temperature, humidity, and Carbon dioxide (CO2) levels on the health and productivity of individuals [27]. CO2 plays a crucial role in evaluating Indoor Air Quality (IAQ) [15]. Elevated levels of CO2 are a sign of inadequate indoor air quality, which may negatively impact human health, decrease productivity, and impair the overall well-being of occupants [8]. As CO2 concentration is highly related to occupancy and IAQ, CO2 level data is also becoming a popular indicator for estimating indoor air conditions.

As a result, the challenge transforms into an optimization issue focused on keeping CO2 concentrations under a specific limit while minimizing energy consumption of HVAC systems.

In order to evaluate and feel how modern technology influence indoor ventilation control, this paper provides a comprehensive review of the most recent approaches for optimal ventilation control in relation to CO2 concentration.

The organization of the paper is as follows: Section 2 offers an overview of CO2 concentration and HVAC systems. Section 3 firstly explores various studies that have employed traditional methods in ventilation control, then delves into a range of contemporary approaches highlighted in recent research. Section 4 expresses key insights and discusses potential avenues for future research in this field. Finally, Section 5 concludes the paper.

2 CO2 concentration and its impact

In practical applications, the concentration of CO2 is commonly utilized as a metric for assessing indoor air quality [15]. Elevated CO2 levels typically indicate inadequate ventilation, as they reflect a lack of sufficient exchange with fresh external air or the absence of air purification. Such conditions not only lead to the accumulation of CO2 but also facilitate an

increase in concentration of other pollutants, including volatile organic compounds (VOCs) and particulate matter. These pollutants can originate from various sources within the indoor environment and have the potential to compromise the health and comfort of occupants. Therefore, maintaining appropriate CO2 levels is essential as it serves as a proxy indicator for the presence of various pollutants and the overall effectiveness of a building's ventilation system.

Although CO2 is a byproduct of human respiration and not inherently a pollutant, elevated levels of CO2 in indoor environments have been shown to adversely affect human health and productivity [9]. Research indicates that an increase in CO2 concentration can impact the cardiovascular and respiratory systems [23]. One study suggests that exposure to CO2 levels ranging from 900 ppm to 2800 ppm can lead to instability in heart rate and increased vascular resistance [25]. Furthermore, at a concentration of 5000 ppm, significant increases in blood pressure and blood pH levels have been observed [6]. Prolonged exposure to such elevated CO2 conditions can result in subjective symptoms such as headaches and surges in cerebral blood flow, posing significant health risks to individuals [22]. Moreover, even with a CO2 concentration level a bit higher than 1000 ppm, people's ability to recognize has been proven to be decreased [7]. To limit the effects of CO2 concentrations on building occupants, most countries set up their own standards for building designers. The CO2 concentration limit of 1000 ppm is commonly recommended in different countries as standards for the management of generic IAQ concerns.

3 Ventilation control using CO2 monitoring

It is widely recognized that individuals spend a considerable portion of their time indoors, with estimates suggesting this could be as much as 90% [12]. Concurrently, the building sector is a major consumer of energy. According to the US Energy Information Administration (EIA), this sector is responsible for over one-third of final energy consumption across residential, commercial, and industrial domains [16]. The European context shows a similar trend, with buildings consuming over 40% of produced energy [20]. HVAC systems are particularly significant, often being the largest single category of energy use within buildings. In regions with extreme temperatures, such as the Arabian Gulf, the energy demand for

air conditioning alone can constitute up to 80% of a building's energy consumption [3]. Therefore, enhancing the energy efficiency of HVAC systems is not only of environmental importance but also economically beneficial, providing a significant opportunity for energy conservation in the sector.

The ventilation system can be mainly divided into two categories: natural ventilation and mechanical ventilation. Even though many studies focused on developing natural ventilation design for buildings in recent year, mechanical ventilation systems are still the most widely used. There are many different types of ventilation systems: mixing ventilation, displacement ventilation, personalized ventilation, hybrid air distribution, stratum ventilation, local exhaust ventilation, and piston ventilation. Each system has its specific set of advantages and trade-offs regarding cost, energy consumption, installation complexity, and maintenance needs.

3.1 Traditional ventilation control

Compare to conventional ON–OFF control, demand-controlled ventilation (DCV) becomes more and more popular as it shows great energy-saving advantages. Plenty of ventilation control system aim to use CO2 concentration data as a indicator to employ DCV, as CO2 concentration shows a strong relation with the presence of occupancy and IAQ. At the meantime, CO2 detector protects people's privacy more compared to optical sensors [10].

PID and rule-based control

In the past, conventional control strategies such as Proportional-Integral-Derivative (PID) and rule-based systems have been extensively implemented across various domains. A PID controller continuously calculates the difference between the CO2 setpoint and the current CO2 concentration, then applies a correction based on proportional, integral, and derivative terms. The proportional part is the fundamental one; it makes an in-time response to drag the controlled value to the setpoint. If the CO2 levels have deviated from the setpoint for a while, it means there is a consistent error. The integral part of the controller will accumulate the error over time and try to get the CO2 level back to the setpoint. The derivative part will foresee the trends of CO2 level changes and apply the corresponding corrections. The rule-based control follows pre-determined

rules. For example, if the CO2 concentration is below a specific level, it will apply minimal ventilation; if the CO2 concentration is above a specific level, it will maximize ventilation. It is simpler than PID control but has easier deployment. Time-based control, which is very common in reality, is also belonging to rule-based control.

Nonetheless, the inherent complexity of real-world scenarios and the requisite adaptability pose significant challenges to the effective deployment of these methodologies. Specifically, PID controllers risk instability unless meticulously tuned, while basic rule-based controls often falter in accurately addressing the nonlinear dynamics characteristic of CO2 management processes [5].

Model Predictive Control

Model Predictive Control (MPC) can be a traditional control method but can also be a data-driven control method. It depends on how the predictive model is obtained and used. If the MPC replies with a mathematical model of the system, it is a traditional one. In this context, the model is determined by physical laws and the system's mechanics. MPC considers various factors, both the ventilation system aspect and the environment aspect. For example, the ventilation efficiency, the outdoor air efficiency, and the occupancy pattern. Nevertheless, it at least takes current and historical CO2 concentrations as inputs, forecasts future air quality conditions, and makes decisions based on those predictions.

However, despite its predictive capabilities, MPC has several disadvantages. On one hand, it usually has poor generalization capability. In detail, MPC relies on the accurate model of the system, it is not easy to make the model generalize across different operational conditions or systems. Or in other words, a model developed for one system might not perform well when applied to different systems. On the other hand, it is hindered by computational demands and inherent latency in execution, making it less viable for real-time applications [19].

Other traditional control

Not like other methods that try to explain complex systems, fuzzy logic control, on the other hand, simplifies complex systems into more manageable sub-systems through the designer's insight. Yet its utility is curtailed by the subjective nature of rule formulation and the intricacies involved in its architectural design. Moreover, fuzzy controllers lack the capacity for interactive learning in practical settings [13]. In contrast, Genetic

Algorithms (GAs) offer a novel approach by simulating the evolutionary process to identify optimal control strategies. This method involves generating a diverse set of potential solutions, assessing their suitability, and employing genetic operations to refine them iteratively [4]. However, the primary drawback of GAs, particularly when addressing problems of a larger magnitude, is their substantial computational requirements.

3.2 Data-driven ventilation control

When implementing data-driven ventilation control using CO2 concentration data, it usually means using CO2 data to predict the occupancy rate and then adjust the control strategy according to the demand.

Tradition machine learning approaches
Start with basic machine learning approaches, support vector machine (SVM) is a supervised learning model used for classification and regression analysis. It operates by finding the hyperplane that best divides a dataset into classes in the feature space. The author of [29] uses this model to achieve optimal control by predicting occupant. The author of [21] also uses another supervised classification model, random forest (a typical structure illustrated in Figure 1), to predict occupancy. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (for classification) or mean prediction (for regression) of the individual trees.

The study [18] proves that decision tree and hidden Markov model (HMM) algorithms are well suited for occupancy detection. Decision tree is also a supervised learning algorithm used for classification and regression tasks, which is also the fundamental element in random forest model we discussed before. The HMM is a statistical model that assumes the system being modeled is a Markov process with unobserved (hidden) states. This assumption enables the method being very powerful in exploring sequential characteristics and capturing temporal dynamics of observed data.

Neural-network-based machine learning methods
As internet of things in buildings becomes more and more popular and the great hardware development in larger scale computation. There are some innovative methods based on neural network emerging.

Multiple layer perceptron (MLP) (structure is illustrated in Figure. 2,

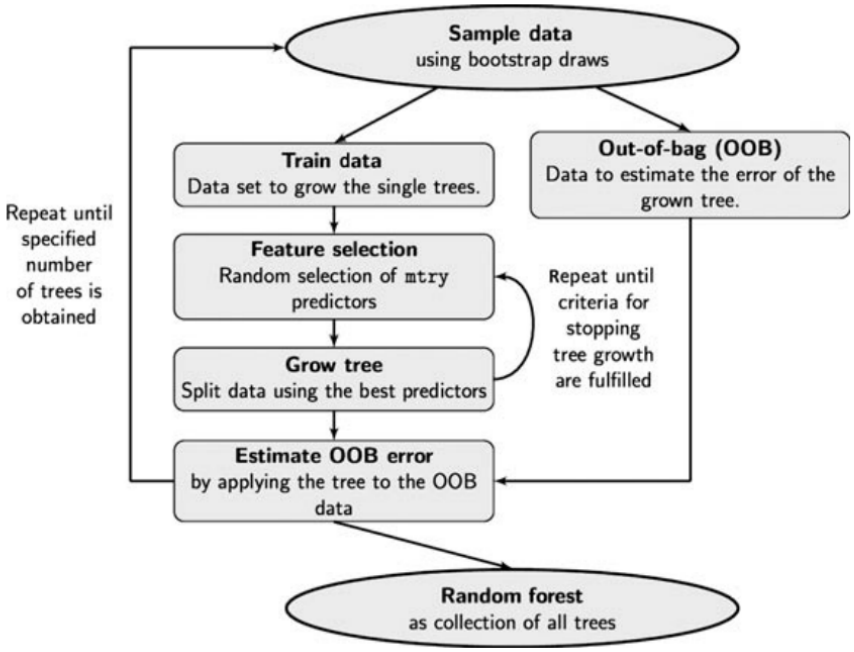


Figure 1. Typical random forest algorithm structure [2].

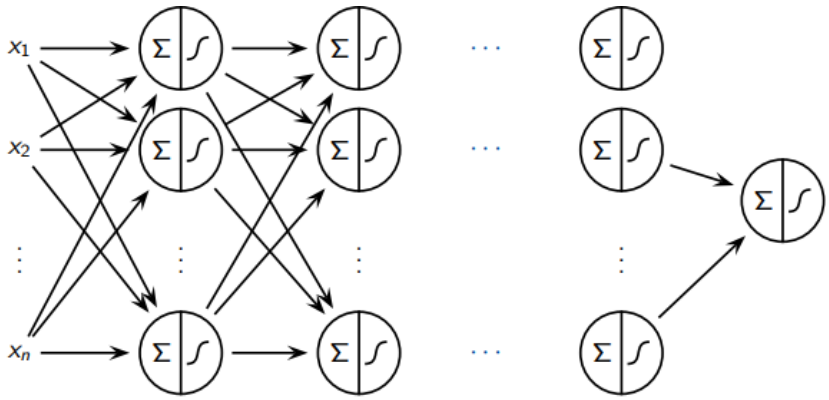


Figure 2. Multilayer perceptrons structure. [29]

is the pioneering method using neural network, and has been proven being capable of solving highly non-linear and complex problems. The MLP has multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Those layers can explore the relation between inputs and outputs. In study [24], the research team proves the MLP can strongly predict the volatile CO2 behavior. They took past passive infrared sensor data, temperature, Dewpoint, and humidity as input features, successfully forecast CO2 concentration in advance of 1, 6, and 24 h. A Convolutional Neural Network (CNN) is a deep learning algorithm specialized in processing image-like data. However, to my best knowledge, there is no CNN using environmental features like CO2 concentration to predict occupancy or to control the ventilation optimally.

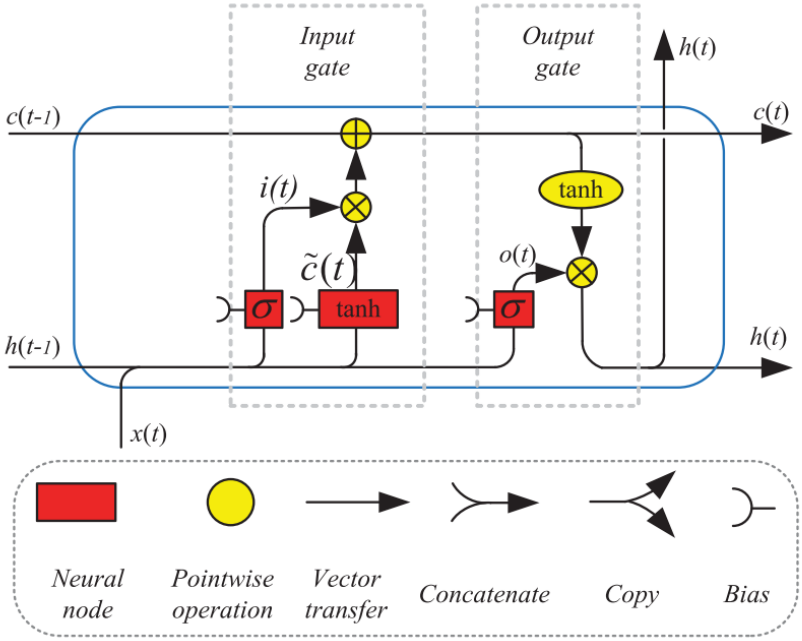


Figure 3. Original LSTM structure. [28]

Long Short-Term Memory (LSTM) models [28] are a special kind of Recurrent Neural Network (RNN) capable of learning time-series data. Compared to RNN, it overcomes the problem of vanishing and exploding gradients in typical RNN models. The author of [14] explored the way using LSTM model to predict occupancy. They collected data on indoor and outdoor temperature, humidity, and CO2 levels, and implement the LSTM model to infer the occupancy situation based on those data. Even though their work did not test the energy saving capability with ventilation control, it still shows the potential in HVAC system automation.

Reinforcement Learning (RL) in ventilation control using CO2 monitoring is a sophisticated approach that employs the principles of reinforcement learning to optimize indoor air quality and energy efficiency. In this method, an agent, usually the ventilation control system, to interact with the indoor environment to learn the best ventilation control strategy. It aims to achieve optimal air quality with minimal energy use. The authors of [5] and [3] explored the reinforcement learning implementation in ventilation control with field studies. In a supervised learning framework, the agent is first trained using the baseline. Then, the agent learns in an actor-critic framework using proximal policy optimization. According to the findings [3], it can result in healthier CO2 concentrations, a 44% improvement in thermal comfort, and a 21% decrease in energy consumption.

Recently, large language models (LLMs) are very popular. For example, ChatGPT is the current hottest topic. It is is based on the Transformer architecture. The transformer architecture, which solely relies on an attention mechanism to identify global dependencies between input and output, marks a substantial divergence from earlier sequence learning models. The study [1] firstly explored the way using the ChatGPT in HVAC control. By using it, the HVAC system is able to make decisions in a sequential manner according to ChatGPT's suggestions.

4 Discussion

By investigating those traditional ventilation control methods and data-driven ventilation control methods, we can find the common disadvantages of both sides. From our perspectives, data-driven methods usually have the advantages of high accuracy and efficiency in predicting demand, thus saving more energy while still providing good indoor air quality. However, these methods usually require sensors to collect enough data and also have computational requirements. In this way, traditional methods are quite cheap, which is also the reason they are still dominating the current ventilation control markets. In conclusion, we think data-driven methods have a bright future as the cost of hardware will reduce and more and more powerful models are emerging.

5 Conclusion

In this paper, we first review the importance of CO2 concentration and ventilation. Then, we offer a thorough analysis of the most recent methods for the best ventilation control in relation to CO2 concentration. We go through both traditional ventilation control methods and data-driven ventilation control methods. We find data-driven methods have more promising potential in future ventilation control applications.

References

[1] Ki Uhn Ahn, Deuk-Woo Kim, Hyun Mi Cho, and Chang-U Chae. Alternative approaches to hvac control of chat generative pre-trained transformer (chatgpt) for autonomous building system operations. *Buildings*, 13(11):2680, 2023.

[2] Anne-Laure Boulesteix, Silke Janitza, Jochen Kruppa, and Inke R König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):493–507, 2012.

[3] Yassine Chemingui, Adel Gastli, and Omar Ellabban. Reinforcement learning-based school energy management system. *Energies*, 13(23):6354, 2020.

[4] Velimir Congradac and Filip Kulic. Hvac system optimization with co2 concentration control using genetic algorithms. *Energy and Buildings*, 41(5):571–577, 2009.

[5] Patrick Nzivugira Duhirwe, Jack Ngarambe, and Geun Young Yun. Energy-efficient virtual sensor-based deep reinforcement learning control of indoor co2 in a kindergarten. *Frontiers of Architectural Research*, 12(2):394–409, 2023.

[6] David A Gortner, AA Messier, E Heyder, and KE Schaefer. *The Effects of Elevated Atmospheric CO2 on Acid-Base Balance and Red-Cell Electrolytes of FBM Submarine Crew Members*. Submarine base, US Naval Submarine Medical Center, 1971.

[7] Ulla Haverinen-Shaughnessy, DJ Moschandreas, and RJ Shaughnessy. Association between substandard classroom ventilation rates and students' academic achievement. *Indoor air*, 21(2):121–131, 2011.

[8] Tyler A. Jacobson, Jasdeep S. Kler, Michael T. Hernke, Rudolf K. Braun, Keith C. Meyer, and William E. Funk. Direct human health risks of increased atmospheric carbon dioxide. *Nature Sustainability*, 2(8):691–701, August 2019. Publisher Copyright: © 2019, Springer Nature Limited.

[9] Tyler A Jacobson, Jasdeep S Kler, Michael T Hernke, Rudolf K Braun, Keith C Meyer, and William E Funk. Direct human health risks of increased atmospheric carbon dioxide. *Nature Sustainability*, 2(8):691–701, 2019.

[10] Wooyoung Jung and Farrokh Jazizadeh. Human-in-the-loop hvac operations: A quantitative review on occupancy, comfort, and energy-efficiency dimensions. *Applied Energy*, 239:1471–1508, 2019.

[11] Neil Klepeis, William Nelson, Wayne Ott, and John Robinson. The national human activity pattern survey (nhaps): A resource for assessing exposure to environmental pollutants. 01 2001.

[12] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. The national human activity pattern survey (nhaps): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science & Environmental Epidemiology*, 11(3):231–252, 2001.

[13] Chuen-Chien Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on systems, man, and cybernetics*, 20(2):404–418, 1990.

[14] Xiguan Liang, Jisoo Shim, Owen Anderton, and Doosam Song. Low-cost data-driven estimation of indoor occupancy based on carbon dioxide (co2)

concentration: A multi-scenario case study. *Journal of Building Engineering*, 82:108180, 2024.

[15] Lidia Morawska, Joseph Allen, William Bahnfleth, Philomena M. Bluyssen, Atze Boerstra, Giorgio Buonanno, Junji Cao, Stephanie J. Dancer, Andres Floto, Francesco Franchimon, Trisha Greenhalgh, Charles Haworth, Jaap Hogeling, Christina Isaxon, Jose L. Jimenez, Jarek Kurnitski, Yuguo Li, Marcel Loomans, Guy Marks, Linsey C. Marr, Livio Mazzarella, Arsen Krikor Melikov, Shelly Miller, Donald K. Milton, William Nazaroff, Peter V. Nielsen, Catherine Noakes, Jordan Peccia, Kim Prather, Xavier Querol, Chandra Sekhar, Olli Seppänen, Shin ichi Tanabe, Julian W. Tang, Raymond Tellier, Kwok Wai Tham, Pawel Wargocki, Aneta Wierzbicka, and Maosheng Yao. A paradigm shift to combat indoor respiratory infection. *Science*, 372(6543):689–691, 2021.

[16] Zhihong Pang, Yan Chen, Jian Zhang, Zheng O'Neill, Hwakong Cheng, and Bing Dong. How much hvac energy could be saved from the occupant-centric smart home thermostat: A nationwide simulation study. *Applied Energy*, 283:116251, 2021.

[17] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and Buildings*, 40(3):394–398, 2008.

[18] Seung Ho Ryu and Hyeun Jun Moon. Development of an occupancy prediction model using indoor environmental data based on machine learning techniques. *Building and Environment*, 107:1–9, 2016.

[19] A Ryzhov, Henni Ouerdane, Elena Gryazina, Aldo Bischi, and K Turitsyn. Model predictive control of indoor microclimate: existing building stock comfort improvement. *Energy conversion and management*, 179:219–228, 2019.

[20] OA Seppänen, WJ Fisk, and Mar J Mendell. Association of ventilation rates and co2 concentrations with health andother responses in commercial and institutional buildings. *Indoor air*, 9(4):226–252, 1999.

[21] Adarsh Pal Singh, Vivek Jain, Sachin Chaudhari, Frank Alexander Kraemer, Stefan Werner, and Vishal Garg. Machine learning-based occupancy estimation using multivariate sensor nodes. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2018.

[22] U. Sliwka, J.A. Krasney, S.G. Simon, P. Schmidt, and J. Noth. Part one: Effects of sustained low-level elevations of carbon dioxide on cerebral blood flow and autoregulation of the intracerebral arteries in humans. *Aviation Space and Environmental Medicine*, 69(3):299 – 306, 1998.

[23] Ge Song, Zhengtao Ai, Zhengxuan Liu, and Guoqiang Zhang. A systematic literature review on smart and personalized ventilation using co2 concentration monitoring and control. *Energy Reports*, 8:7523–7536, 2022.

[24] Saman Taheri and Ali Razban. Learning-based co2 concentration prediction: Application to indoor air quality control using demand-controlled ventilation. *Building and Environment*, 205:108164, 2021.

[25] Tommi Vehviläinen, Harri Lindholm, Hannu Rintamäki, Rauno Pääkkönen, Ari Hirvonen, Olli Niemi, and Juha Vinha. High indoor co2 concentrations in an office environment increases the transcutaneous co2 level and sleepiness during cognitive work. *Journal of occupational and environmental hygiene*, 13(1):19–29, 2016.

[26] Yu Yang, Guoqiang Hu, and Costas J Spanos. Stochastic optimal control of hvac system for energy-efficient buildings. *IEEE Transactions on Control Systems Technology*, 30(1):376–383, 2021.

[27] Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. A review of deep reinforcement learning for smart building energy management. *IEEE Internet of Things Journal*, 8(15):12046–12063, 2021.

[28] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[29] M.S. Zuraimi, A. Pantazaras, K.A. Chaturvedi, J.J. Yang, K.W. Tham, and S.E. Lee. Predicting occupancy counts using physical and statistical co2-based modeling methodologies. *Building and Environment*, 123:517–528, 2017.