

Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN
Computer Science and Engineering
Ciclo XXXIV

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

**Security Enhancements and Flaws of Emerging Communication
Technologies**

Presentata da: Davide Berardi

Coordinatore Dottorato
Prof. Davide Sangiorgi

Supervisore
Prof. Franco Callegati

Co-Supervisore
Prof. Marco Prandini

Esame finale anno 2022

Security Enhancements and Flaws of Emerging Communication Technologies

Ph.D. Thesis in Computer Science and Engineering

Daide Berardi

Copyright © 2021-2022, Davide Berardi ✉ berardi.dav@gmail.com , ✉ davide.berardi@unibo.it & Obsidium S.R.L. 🌐 <http://obsidium.it>
L^AT_EXTemplate released under GNU General Public License v3 and later versions (GPLv3+). You can find the text of the license at the following URL:

<https://www.gnu.org/licenses/gpl-3.0.en.html>

Ph.D. Thesis in Computer Science and Engineering

D.I.S.I. Dipartimento di Informatica, Scienza e Ingegneria, Università di Bologna, via Zamboni 33, 40192, Bologna, Italy.

Licensed under Creative Commons Attribution Non-commercial Share Alike 3.0 (CC BY-NC-SA 3.0)

<https://creativecommons.org/licenses/by-nc-sa/3.0/>



Information for nerds

Build Time: This version of the document was built dozens of times, the last was on Saturday 14th May, 2022 11:40(UTC).

Headers: The images on the headers of the chapters and the background of the cover are randomly generated before every compilation. It is just a random block art with different parameters and colors. I'm not responsible for strange patterns that can be generated unintentionally.

Source: The entire source code of the Thesis (including the polyglot generator pipeline) is available in this document. You can retrieve it by de-zipping the PDF. For instance, you can use the command: `unzip $FILENAME`.

Executable: This file is a PDF/ZIP/Bash script chimera polyglot. The script will print the acknowledgment page encoded in PCAP format of a EtherCAT communication. You can easily verify it by stealing the code from the test directory in the source (included in this document).

```
$ chmod +x $FILENAME
$ ./ $FILENAME
$ # Don't want to spoil it , just a huge hint ;)
$ tshark -r /tmp/acknowledgments.pcap -T fields -e eth.src | ...
ACKNOWLEDGMENTS
...
```


Abstract

The multi-faced evolution of network technologies ranges from big data centers to specialized network infrastructures and protocols for mission-critical operations. For instance, technologies such as Software Defined Networking (SDN) revolutionized the world of static configuration of the network - i.e., by removing the distributed and proprietary configuration of the switched networks - centralizing the control plane. While this disruptive approach is interesting from different points of view, it can introduce new unforeseen vulnerabilities classes. One topic of particular interest in the last years is industrial network security, an interest which started to rise in 2016 with the introduction of the Industry 4.0 (I4.0) movement. Networks that were basically isolated by design are now connected to the internet to collect, archive, and analyze data. While this approach got a lot of momentum due to the predictive maintenance capabilities, these network technologies can be exploited in various ways from a cybersecurity perspective. Some of these technologies lack security measures and can introduce new families of vulnerabilities. On the other side, these networks can be used to enable accurate monitoring, formal verification, or defenses that were not practical before. This thesis explores these two fields: by introducing monitoring, protections, and detection mechanisms where the new network technologies make it feasible; and by demonstrating attacks on practical scenarios related to emerging network infrastructures not protected sufficiently. The goal of this thesis is to highlight this lack of protection in terms of attacks on and possible defenses enabled by emerging technologies. We will pursue this goal by analyzing the aforementioned technologies and by presenting three years of contribution to this field. In conclusion, we will recapitulate the research questions and give answers to them.

Contents

Contents

I Introduction and Motivations

- 1 Introduction** **1**
- 1.1 Software 3
- 1.2 Contributions 4

- 2 Infrastructures and Technologies** **9**
- 2.1 SDN Networks 10
- 2.2 Formal Network Verification 14
- 2.3 Time synchronization between devices 18
- 2.4 Insider Threat 24
- 2.5 Constraint Programming 26

II Contributions

- 3 Analysis** **31**
- 3.1 Microservices security 31

- 4 Defense and detection** **65**
- 4.1 SDN and NFV for Network Monitoring 66
- 4.2 Users' password enforcement 72
- 4.3 Insider Threat Analysis 87
- 4.4 TechNETium: Formal network verification using SDN 95
- 4.5 P-SCOR: Artificial Intelligence and Programmable Data Plane 100
- 4.6 Test Bed and Experimental Results 112

5	Attack and testing	119
5.1	Anagram Attack to password similarity systems	119
5.2	Our Attacks on PTP	121
III Conclusions, Appendices, and References		
6	Conclusions	137
A	TSN protocols	141
B	Taxonomy of PTP TLVs for security	143
C	MicroServices Dataset and R.Q.	147
	Bibliography	151
	Acronyms	199
	List of Figures	205
	List of Tables	209
	Listings	210

Part I

Introduction and Motivations

1

Introduction

This Thesis focuses on *Security*, term which definition is not easy to introduce. That is because Security is a property but it is just a part of the puzzle [297] composed of Actors, Protectors, and Threats. To define Security, the value of the items you want to protect and the kind of actions you consider attacks needs to be analyzed.

Security is therefore a two-faced coin: from one side you can use software to increase the overall Security property of the systems or networks (increasing Protectors against Threats) from the other you can act as an attacker to demonstrate the vulnerability before a real attacker can do so (increasing Actors and Threats against Protectors).

This Thesis focuses on emerging networking technologies, while being an extremely broad topic [10, 153, 21, 436, 477], we will focus on two topics that got research attention in the last years, namely: industrial systems (also known as Industrial Internet of Things) and Data Centers. These two scenarios have different network requirements in terms of bandwidth, real-time and applications, we will get into the details inside the core of this Thesis.

When the first of these topics is introduced, a property called “Safety” can be met in literature [268]. Security and Safety are similar to certain degrees and share some concepts, but they are not considered the same [278]: while Security of a system is the protection from the outside threats, the Safety propriety of a system is the inability to affect the near actors. For instance, Security in the computer world is the protection against unauthorized uses of an Automatic Teller Machine, Safety is the guarantee that the Machine will not close your hand inside the lid.

Therefore, the Threat consideration should consider the system required Safety Integrity Level [300]: that is, if a system is compromised, to which degree its Safety requirements can be compromised?

The application worlds which this Thesis focuses on also uses closed source software and hardware [18]. That is, the protocols and operating systems in industrial applications can be proprietary and not easily analyzable.

This concept is opposed to the concept of Security by obscurity [306]. This point of view enforce

the usage of public algorithms, protocols, and implementations and not rely on the Security of the system because a small set of people knows the internals of the systems. Security through obscurity has been demonstrated to be completely ineffective against malicious agents in this set, the so-called *Insider Threat* [66].

Insider threat was extensively explored in the literature of the last years [497, 262]: it can be defined as a malicious entity inside the organization (e.g., contractors) who can pose a threat with the support of information accessible only to internal users.

From the Actors perspective, the action can be performed in two ways: Reactive, in which a cause triggers an action and Proactive, in which the action is done ahead of time, to (hopefully), remove the possibility of a threat.

Having narrowed the scope of the study, this Thesis aims to reply to the following research questions:

RQ1 *Which are the emerging network technologies that can increase the protection part of the Security of a system in proactive or reactive ways?*

Technologies such as SDNs and network programmability can be used to increase the overall security of the network in terms of monitoring and flow hijacking. The efforts to answer this question can be found in Chapter 4.

RQ2 *Are emerging network technologies introducing new classes of threats?*

While being appealing for the enhanced performances, new network paradigms can introduce security threats. This question was investigated by discovering and implementing attacks on technologies, an answer can be found in Chapter 5.

RQ3 *Are emerging network technologies re-introducing well-known problematics that should be included in a Threat Modelling methodology?*

Aside from novel attacks, security vulnerabilities well known for other fields or protocols (e.g. attacks on HTTPS) can be modified to work on different technologies. These kinds of attacks should be considered when analyzing the security of the infrastructure, e.g. during a penetration test. Re-introduced attacks and threat analysis can be found in Chapter 5.

RQ4 *Can test-bed, resembling the real systems to some degrees, be built to make the process of Security testing in critical scenarios more feasible?*

To demonstrate attacks and effectively measure the consequences of an exploit on real scenarios, the so-called **Digital Twins** can be created. These demonstrators are mainly used for industrial scenario, one of which can be found in Chapter 5.

To answer these questions, this document is structured using the following layout:

- Before the core of the Thesis, is presented a list of the repository with implementations of the results and projects described in the document.
- The first part of the Thesis collects the requirements, dividing them into two parts: Chapter 2, the infrastructure and technologies that we focus on are presented, specifying only the features required for the subsequent Security enhancements proposed from the state of the art or our contributions. This chapter also includes the technologies, with the various tools and theoretical requirements used by our contributions are presented for reference for the next part.
- In the second part of the Thesis, our contributions are presented in terms of analysis of the current attacks and reconnaissance of specific technology such as micro-services in Chapter 3; offensive technologies (attacks) in Chapter 5; and monitoring, detection, and defensive technologies (protections) in Chapter 4.
- The third and last part of the Thesis includes the bibliographical references, the various appendices and conclusions.

1.1 Software

The following software was developed during the years spanning the Ph.D program. The license they are released under may vary, while mainly being Free or Open Source.

5G MEC Test a web application compatible with 5G MEC 011 APIs [6]. The application is a MEC 011 API tester, which can be used to validate the workflow of the APIs automatically. It was presented and tested during one of the ETSI MEC workshops.

<https://github.com/berdav/unibo-test-mec>

PEF and PTP testbed PTP Exploitation framework and testbed, the latter projects can set up a virtual PTP testbed. The former one can be used to attack the system and analyze the security and outcome of the attacks.

<https://github.com/berdav/PTP-Security-Testbed>

cado a capability-based frontend to manage the Linux privileges system.

<https://github.com/berdav/cado>

CTF Infra a demonstrator used to highlight the lacking of insider threat analysis. It is a series of scripts that injects in the cloud various operating systems and creates an infrastructure to

play Capture The Flag (CTF) competitions.

<https://github.com/berdav/ctf-infra>

TechNetium is a network policy formal verification framework. It can be used to verify various policies such as layer 2 and layer 3 reachability, full reachability, and to waypoint. It is developed as a standalone module, while it was mainly used with software defined networks.

<https://github.com/berdav/TechNETium>

1.2 Contributions

During the Ph.D. research, various research papers have been published.

Here we list the contributions to the research we developed and lead to the creation of this Thesis.

Microservices 2022: Microservice security: a systematic literature review

Authors: **Davide Berardi**, Saverio Giallorenzo, Jacopo Mauro, Andrea Melis, Fabrizio Montesi, Marco Prandini

Type: Journal Article (PEERJ. COMPUTER SCIENCE)

Abstract:

Microservices is an emerging paradigm for developing distributed systems. With their widespread adoption, more and more work investigated the relation between microservices and security. Alas, the literature on this subject does not form a well-defined corpus : it is spread over many venues and composed of contributions mainly addressing specific scenarios or needs. In this work, we conduct a systematic review of the field, gathering 290 relevant publications—at the time of writing, the largest curated dataset on the topic. We analyse our dataset along two lines: (a) quantitatively, through publication metadata, which allows us to chart publication outlets, communities, approaches, and tackled issues; (b) qualitatively, through 20 research questions used to provide an aggregated overview of the literature and to spot gaps left open. We summarise our analyses in the conclusion in the form of a call for action to address the main open challenges.

P-SCOR 2021: P-SCOR: Integration of Constraint Programming Orchestration and Programmable Data Plane

Authors: Andrea Melis, S. Layeghy, **Davide Berardi**, Marius Portmann, Marco Prandini, Franco Callegati

Type: Journal Article (IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT 2021)

Abstract:

In this manuscript we present an original implementation of network management functions in the context of Software Defined Networking. We demonstrate a full integration of an artificial intelligence driven management, an SDN control plane, and a programmable data plane. Constraint Programming is used to implement a management operating system that accepts high level specifications, via a northbound interface, in terms of operational objective and directives. These are translated in technology-specific constraints and directives for the SDN control plane, leveraging the programmable data plane, which is enriched with functionalities suited to feed data that enable the most effective operation of the “intelligent” control plane, by exploiting the P4 language.

Microservices 2020: A Survey on Microservices Security: Preliminary Findings

Authors: **Davide Berardi**, Saverio Giallorenzo, Jacopo Mauro, Andrea Melis, Fabrizio Montesi

Type: Conference Abstract (Microservices 2020)

Abstract:

In recent years Microservices have become the state-of-the-art architectural style for distributed systems. Despite its widespread adoption—and possibly due to its recent introduction—we notice the lack of comprehensive guidelines on Microservices Security. Motivated by this observation, we started an ongoing Systematic Literature Review process to categorise the literature on Microservices Security, with the intent to overview the current status of the field, to provide an initial guideline to researchers and practitioners, and to possibly identify uncovered areas and orient future research.

Password Similarity 2020: Password Similarity Using Probabilistic Data Structures

Authors: **Davide Berardi**, Franco Callegati, Andrea Melis, Marco Prandini

Type: Journal Article (JOURNAL OF CYBERSECURITY AND PRIVACY MDPI)

Abstract:

Passwords should be easy to remember, yet expiration policies mandate their frequent change. Caught in the crossfire between these conflicting requirements, users often adopt creative methods to perform slight variations over time. While easily fooling the most basic checks for similarity, these schemes lead to a substantial decrease in actual security, because leaked passwords, albeit expired, can be effectively exploited as seeds for crackers. This work describes an approach based on Bloom Filters to detect password similarity, which can be used to discourage password reuse habits. The proposed scheme intrinsically obfuscates the stored passwords to protect them in case of database leaks, and can be tuned to be resistant to common cryptanalytic techniques, making it suitable for usage on exposed systems.

Goodtechs 2020: Sustainable Infrastructure Monitoring for Security-Oriented Purposes

Authors: **Davide Berardi**, Franco Callegati, Andrea Melis, Marco Prandini

Type: Conference Paper (GOODTECHS 2020)

Abstract:

As computing and communication infrastructures have gained an ever-increasing role in everybody's life, guaranteeing their reliability has become a critical endeavor. In the face of threats that grow more and more sophisticated, we must turn our attention to the techniques that have the potential to match them and scale with the infrastructure complexity. The current trend in the telecommunication industry towards "softwarized infrastructures" by means of new technologies such as Software Defined Networking and Network Function Virtualization may provide innovative and effective solutions from this point of view. In this work, we outline a network security monitoring architecture aimed at striking the best trade-off between effectiveness and efficiency. This result is achieved by exploiting the possibility, already enabled by state-of-the-art, yet well tested components for infrastructural orchestration, of dynamic instantiation and composition of functions. We conclude that efficient detection of some classes of network-based denial-of-service attacks is possible, and open the path to mitigation strategies that optimize the usage of resources by deploying and re-configuring them as needed in real-time.

CCNC 2020: TechNETium: Atomic Predicates and Model Driven Development to Verify Security Network Policies

Authors: **Davide Berardi**, Franco Callegati, Andrea Melis, Marco Prandini

Type: Conference Paper (CCNC 2020)

Abstract:

Fifth-generation (5G) networks will deliver unprecedented levels of quality of service for online gaming and multimedia-rich social interaction, providing virtual environments optimized for vertical applications through innovative approaches to physical resource management. These techniques must consider security aspects in all phases and at every layer. Trusted communications between individuals and reliable platforms running services for social good depend on the resiliency to network-level attacks such as hijacking and denial-of-service. The verification of topological properties represents a well-suited approach to address these issues in a 5G environment. This paper illustrates moves from formal methods existing in literature, namely atomic predicates (AP) and header space analysis (HSA). It describes a method of integrating AP in Software Defined Network architectures, achieving the same expressive power as HSA without its performance hit, to make topology verification viable for real-time security applications.

CSNet 2018: A Policy Checker Approach for Secure Industrial SDN

Authors: Andrea Melis, **Davide Berardi**, Chiara Contoli, Franco Callegati, Flavio Esposito, Marco Prandini,

Type: Conference Paper (CSNet 2018)

Abstract:

Industry 4.0 is a new strategic industrial development that is changing the way business develop communication and management protocols on their networks. Software-Defined Networking (SDN) can help this revolutionary process but to make the most of its potential, more abstract and customizable development paradigms are needed. In this work we present a toolkit whose scope is to allow a system network administrator to implement and verify in a formal way security policies, in the context of an industrial network. The prototype of our tool suite is based on four application plug-ins of the ONOS controller. Our SDN-based toolkit is able to detect compromised network boxes as a result of bogus injected flow-rules, inner loops and black-holes (notoriously difficult to detect via normal network scans), flow-rule replacements or removal and other SDN controller exploitations that may compromise the forwarding activities. We argue that our set of tools is already effective despite being at its development infancy, and its design easily extensible to other use cases.

ITNAC 2018: Security network policy enforcement through a SDN framework

Authors: **Davide Berardi**, Franco Callegati, Andrea Melis, Marco Prandini

Type: Conference Paper (ITNAC 2018)

Abstract:

In this work we present an exploitation of the Software Defined Networking paradigm to implement an architecture allowing a system network administrator to implement and verify in a formal way security policies. The main result is a framework that support the network administrator in the security management process providing services during all this phase, from automated traffic analysis during the prevention phase to tools for the exclusion of malicious traffic from the main flow in the reaction phase. In order to validate the proposed architecture we will showcase an industrial network applied scenario, simulating attacks and countermeasures techniques.

2 Infrastructures and Technologies

Various types of network infrastructures are developed in the literature, micro-services, Software Defined Networking (SDN), Network Function Virtualization (NFV), etc [10, 153, 21, 436].

Mainly, these infrastructures are not usually developed with security as the first principle [117] but focusing on bandwidth, resilience, interoperability between vendors, easiness of configuration.

We will focus mainly on network virtualization, microservices, and virtualization of network functions. These infrastructures are disruptive in traditional terms [207]. For instance: the paradigm is shifted from a distributed to centralized for networking (SDN) to ease the process of logging or managing the control plane. On the opposite side, from centralized to distributed flavor, the microservices approach removes the burdens of a monolithic server that is not easily maintainable or scalable [143].

In these sections, we will introduce the main definitions of the infrastructures.

Micro-services

Micro-services is an emerging development paradigm, where software is built as a composition of multiple services (the “micro-services”). Each micro-service implements the business logic of a component of the application and is independently executable and deployable. Micro-services interact with each other via message-passing APIs [143]. Over the last 6 years, micro-services have become a popular topic and one of the go-to approaches for many cloud computing projects. According to Web of Science, more than 1,000 articles about micro-services have been published since 2014. The year 2020 accounts for more than 400 of them, which points out that interest in the topic is still rising. Micro-services are popular because they bring substantial advantages with respect to scalability in cloud environments and flexibility in the process of software development. By separating application components as independent services, software designers can specialise each component by using a dedicated technology and then integrate all such heterogeneous components via technology-agnostic APIs.

Alas, the advantages of microservices come at a cost: distributed systems are hard to manage, and increasing the number of services of an application gives malicious actors a larger attack surface [143]. Several security concerns that are particularly relevant for microservices have been identified by [98], and early research has already shown that the application of standard patterns for system reliability needs to take new parameters into consideration—like the locations at which the patterns are deployed [317].

2.1 SDN Networks

The Software Defined Networking principle aims at a full decoupling between network control (the configuration of network instruments) and data (the actual data flowing in the network) planes. The control plane is typically implemented with controllers that communicate with switches implementing the data plane that forward messages along the network.

The controller provides switches with a set of flow-rules that instruct them on how to forward the different streams of messages they receive from end hosts or other switches. The main innovations brought forward by SDN are that:

- a flow may typically be identified by a subset of the packet header field that may belong to different protocol layers (for instance IP source and destination and some combination of TCP/UDP ports);
- forwarding rules may change over time as the controller dynamically makes new decisions on how to route individual flows, possibly reacting to some sort of network behavior.

The communication between control and data planes can be implemented in various ways; nowadays, the de-facto standard is OpenFlow [301], and an SDN switch has one or more flow tables, configured by the SDN controller via the OpenFlow API.

In normal operations a flow table contains rules that match a given packet header with common networking actions such as prioritization, queuing, packet switching, etc.; but the same approach can be used to trigger more complex functions, such as intrusion detection, load balancing or firewalling, that are traditionally implemented as dedicated hardware devices with proprietary architectures. Therefore SDN raises the opportunity to develop security-oriented network functions in a software defined fashion that makes them more flexible, general and scalable.

SDN may be integrated with the current virtualization and cloud computing technologies to implement cheaper and more agile network infrastructures, while an automated management of the

lifecycle of such infrastructures is provided by the Network Function Virtualization approach [21]. NFV provides the architecture, components and interfaces specifications to orchestrate and manage virtualized networking functionalities in a coordinated way.

It follows that by exploiting SDN and NFV, security network functions such as Intrusion Detection System (IDS), Intrusion Prevention System (IPS), firewalls, etc. can be deployed as virtual components of a cloud infrastructure orchestrated through the SDN control mechanisms for proper operations, and exploiting NFV for lifecycle management [401, 303].

Finally, they can be combined with a forwarding plane that can be also integrated in the virtualized environment, for instance by exploiting Open vSwitch (OvS), which is now tightly integrated with various cloud platforms, including OpenStack, openQRM, OpenNebula and oVirt [469].

A survey of previous works in the SDN field specifically regarding network performance measurement revealed attempts to overload the SDN controllers [403, 256, 505] in a simulator; however, simulated network components do not provide realistic results that reflect the SDN controller limits.

Indeed, for the same reasons, virtualized switches cannot offer a realistic view of the test environment of the SDN network. The evaluation of network emulator capabilities have been brought from [437] et al. 2015] in the field of network emulation to estimate network boundaries and implement configuration scenarios for experiments

Numerous traffic monitoring tools have been proposed for OpenFlow networks. OpenNetMon [281] was developed to determine whether end-to-end QoS parameters are met and to allow applications to calculate appropriate routes. It is implemented as a module for SDN controllers, polling the edge switches to collect flow statistics at an adaptive rate to determine throughput, packet loss and delay.

It provides a rich set of statistics, but only on predefined endpoints. (in contrast, the approach that we will present in this thesis controls port throughput, rather than flows, between any two points on the network, while not measuring packet loss and delay). As stated in [370] an available bandwidth measurement application has been proposed in work [302]. This application can travel on the network topology and track bandwidth guidance on network connections. As a result of this procedure, it is possible to calculate the available bandwidth between two points on the network.

Another interesting work has been done in [288], where authors proposed a solution for link traffic throughput monitoring for the control plan. In this solution, throughput statistics are computed on each port to provide aggregate data at different levels of resource monitoring. Yet another solutions has been devised in [410], where authors proposed an online approach for flow and monitoring and measurement, port by port, to provide statistics on packet loss in SDN.

Eventually a comprehensive example of infrastructure monitoring with SDN was shown in [414]. Based on the monitoring of the network status in this paper, the authors showed that the status of

monitoring the network against network failures, congestion, incorrect configuration and security attacks should be an integral part of creating reliable network services.

Further research in the area of monitoring the state of the SDN network revealed that the main problem of such tools is the ability to scale following the increasing number of network nodes. Another large SDN security survey [400] showed that logging network events adds high value to network operations and can improve security and infrastructure reliability.

The exploitation of SDN to improve the security awareness of the network is not something new in the literature. Since the first implementations of SDN, the research community has seen a huge potential in security-related applications. These solutions tackle the problem from different directions

RFC8329 [282] proposes a reference model for defining the interfaces to the deployment and configuration of network security functions, be they physical or virtual.

In [498] SDN was used to create either IDS or IPS components as needed, exploiting its ability to dynamically adapt the network topology. The same property was exploited in other works about DDoS mitigation over SDN-based infrastructure.

In [144] authors proposed a novel SDN application that protects SDN networks against DoS attacks and mitigate their impact on the SDN controller performance, on the consumption of the control plane bandwidth, and on switches workload. This tool was designed to mitigate simultaneously these issues by dynamically managing flow routes, rule entry timeouts, and the aggregate flow rule entries based on the flow threat probability provided by an IDS.

In [503] authors instead proposed an SDN solution for monitoring the deployment of an SNMP server, based on security attacks heuristics.

In [406] authors developed a flexible sampling extension for OpenFlow that enables the controller to access packet-level information. In particular, this extension has the advantage of generating a more accurate estimate of traffic statistics in comparison to per-packet sampling methods that were commonly used in previous literature. This kind of solution can also exhibit a better fit for security applications (such as IDS) that need data about short-lived flows, which can be missed by uniform sampling or flow-based sampling techniques that focus on heavy-hitters.

As already outlined, SDN aims at separating the network control plane from the forwarding plane. Controllers interact with forwarding devices via the so called SouthBound Interface. OpenFlow is the de-facto standard for southbound interfaces to date [301].

OpenFlow started simple, with the abstraction of a single table of forwarding rules that could match packets on a dozen header fields (MAC addresses, IP addresses, protocol, TCP/UDP port numbers, etc.) but, over the past five years, the specification has grown and OpenFlow is now more complex and feature-rich [84].

In the first version of OpenFlow, just 4 header features were available, now there are more than 50 [4]. This is not necessarily a positive trend. There is a widespread belief that, rather than repeatedly extending the OpenFlow specification, the future switches should support flexible mechanisms for parsing packets and matching header fields directly, allowing controller applications to leverage these capabilities through a common, open interface [229].

A good example has been shown in [71], where data-center network operators increasingly want to apply new forms of packet encapsulation (e.g., NVGRE, VXLAN, and STT), for which they resort to deploying software switches that are easier to extend with new functionality.

P4

The idea of programmable switches has been around for a long time; in the past it was hindered by the performance degradation of programmable switches, due to the fact that the vendor chips had to adapt to different specifications instead of focusing on a subset of features and making them perform at their best. More recently, thanks to the advances in ASICs design, it was demonstrated [307] that programmable forwarding can be achieved at terabit/s speeds, thus making programmable switches comparable to legacy ones. These are the main motivations that inspired the development of a programming language for the data plane: the *P4 language*.

P4 is an open-source programming language which lets the end users describe how the switch should process the packets. It controls silicon processor chips in network forwarding devices, enabling a paradigm change from a “bottom-up” approach where fixed-function switches are built-in, to a programmable “top-down” approach where the user decides which functionalities to install. [83]

Basically P4 has three main goals.

- Reconfigurability. The controller should be able to redefine the packet parsing and processing in the field.
- Protocol independence. The switch should not be tied to specific packet formats. Instead, the controller should be able to specify a packet parser for extracting header fields with particular names and types and a collection of typed match/action tables that process these headers.
- Target independence. The controller programmer should not need to know the details of the underlying switch. The P4 compiler should translate the program features into target-dependent capabilities.

2.2 Formal Network Verification

Security tools based on formal verification have recently attracted much interest; this is especially true for network verification [52, 205, 16]. A recent work by Beckett et al. [52] proposes a tool called Minesweeper following the study of several tools developed by the research community with the goal of finding network misconfigurations. The studies are classified into two categories: control plane oriented, i.e., able to discover flawed configurations proactively, and data plane oriented, i.e., able to discover misconfigurations reactively, by observing the events happening in the network while traffic is flowing.

Both categories have pros and cons. Proactive approaches are particularly useful to predict potential network misconfigurations that might lead to security issue (e.g., BGP prefix hijacking); however security breaches in the most general sense are hard to predict a priori, therefore, the reactive approach is essential to detect unwanted (dynamic) network behavior occurring as a consequence of malicious activity. Following the same line, we argue that both proactive and reactive approaches should be adopted and combined to exploit synergies between them at best.

At the same time, the spread of the SDN has sparked a debate about the security of *software defined* solutions. Here we will not address the whole of this large subject, concentrating our attention on some issues regarding the correctness of topology and routing. Studies like [16, 251] have analyzed the attack surface highlighting the vulnerabilities and security requirements of the architecture. In [16] it is suggested the combined adoption of *proactive* and *reactive* approaches, however, works like [304] emphasize the importance of reactive solutions for verification of invariant in the data plane: the problem consists in checking if certain (*security policies*) are respected starting from the *Forwarding Information Base*, that is from the rules of forwarding and from the topology of the network.

This is a complex problem since it requires analyzing the behavior of all possible packet headers. The *header space*, as it is called in [238], can be very large (the IP header being at least 160 bit long), and the union and intersection of packets are computationally expensive operations (complexity $O(2^n)$ in the worst case, where n is the number of bits in the header [484]), thus verification of properties in a network with thousands of devices can take a very long time. To improve performance, Yang and Lam propose [484] to reduce the space to be explored: instead of working on individual headers, they use sets of packets which are equivalent from the viewpoint of forwarding, called *Forward Equivalent Class (FEC)*, and a symbolic representation is adopted to reduce computational complexity. The same researchers as well as other research groups [67] performed experiments supporting their idea. The code of these experiments is available at [486].

Atomic Predicates for Transformations

The fundamental idea of the method of Atomic Predicate (AP) consists in the representation of port predicates through the FECs that are forwarded through the corresponding port. The boolean functions, or predicates, used for the match of the packets are represented by Binary Decision Diagram (BDD), an acyclic direct graph data structure whose nodes represent Boolean variables; each node of a BDD has an *low child* and an *high child* and the arcs linked to child nodes represent the assignment of the false and true value to the node variable, respectively. The evaluation of a predicate proceeds recursively by choosing the child up or down based on the value of the variable, up to the result of the evaluation reaching a terminal node *true* or *false* [15]. The representation with BDD is flanked by the set of FECs that pass through the predicate gate. This is possible because within each device the packets belonging to an FEC are forwarded on a single port, therefore the predicates are seen as disjoint sets of FECs. At this point a further step is taken: the FECs are uniquely associated with integer identifiers. It follows that the set of identifiers of the FECs that pass through a port is equivalent to the original information of the predicate and can be used for calculating the properties of the network. The equivalence between these representations is demonstrated in [484, 487] and is indicated in [362] as *equivalence of Yang and Lam*.

In this way the disjunction and conjunction operations on the predicates are mapped on union and intersection operations on sets of integers, with a considerable reduction in complexity.

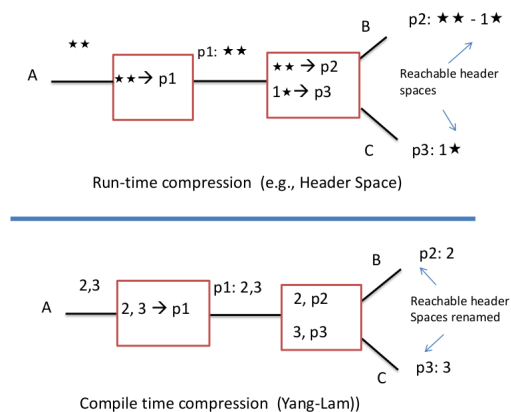


Figure 2.1: Atomic predicates Verifier - Differences between header space and quotient space

In the example shown in figure 2.1, taken from [67], we illustrate the differences between atomic predicates and analysis in header space:

For simplicity, suppose we have 2-bit long headers. In the image we have two routers that adopt

the longest match semantics: in the first one there is only one rule that involves the forwarding of all the input (wildcard expression (**)) of the p0 port on the p1 port, in the second router, is specified that the traffic with prefix (1 *) must be forwarded to port 3, the rest of the traffic must flow to the p2 port instead.

Using NetPlumber [237] we would have 3 *rule nodes*, one for each rule, an *intra-table dependency* in the second router table and an *pipe filter* for port 2 of the type: (** - 1 *). Furthermore, the difference between the two prefixes, which in this simple case is (0 *), must be computed at run time, requiring an expensive operation of intersection between *wildcard expressions*.

Calculation of atomic predicates

Once demonstrated that APs correspond to equivalence classes, we need a way to extrapolate them quickly from a set of port predicates. For this purpose, we need the following two formulas and the algorithm inspired by [484].

Theorem 1. Atomic Predicates, Cardinality Set 1 Given a Predicate p we denote $A(\{p\})$ as the set of APs of the set compose only by the predicate p and:

$$A(\{p\}) = \begin{cases} \{true\} & \text{se } p = true \text{ or } false \\ \{p, \neg p\} & \text{else} \end{cases} \quad (2.1)$$

Theorem 2. Atomic Predicates of the union of two sets of predicates Given $P_1 = \{b_1, b_2, \dots, b_m\}$ e $P_2 = \{c_1, c_2, \dots, c_n\}$ 2 Atomic Predicates set. $AP = \{a_1, a_2, \dots, a_k\}$ is the result of:

$$\{a_i = b_{i_1} \wedge c_{i_2} \mid a_i \neq false, i_1 = \{1, \dots, m\}, i_2 = \{1, \dots, n\}\} \quad (2.2)$$

$AP = A(\{P_1 \cup P_2\})$ in [484].

The following algorithm applies the given formulas to calculate the APs of a set of one element and the union of two sets to obtain the APs of an arbitrary set of predicates.

Algorithm 1 Atomic Predicates calculation algorithm

Input: set of predicates $\{p_1, p_2, \dots, p_N\}$ **Output:** atomic predicates input set, $A(\{p_1, p_2, \dots, p_N\})$

```

1: for  $i = 1$  to  $N$  do
2:   Compute  $A(\{p_i\})$  using (2.1)
3: end for
4: for  $i = 2$  to  $N$  do
5:   Compute  $A(\{p_1, \dots, p_i\})$  using (2.2) with  $A(\{p_1, \dots, p_{i-1}\}) \in A(\{p_i\})$ 
6: end for
7: return  $A(\{p_1, \dots, p_N\})$ ;

```

Alterations in the state of the topology do not modify the APs, however, the policy tree could vary and an update is therefore necessary. When the rules are updated, three operations must be performed:

1. Verifying that altered predicates call for a recalculation.
2. Updating atomic predicates.
3. Updating the reachability tree item.

Operations 2 and 3 can be carried out concurrently.

For port upgrading, the rules are organized in a forest. Each tree is constructed by iterating over the forwarding table of a device and setting the rules whose prefix is contained in the current prefix as daughters of the current rule.

Finally, we need also to consider updates of APs. Here we have to consider of course two cases:

1. **Addition of a predicate:** just apply the formula (2.2), to calculate the APs of the union of two sets.
2. **Deletion of a predicate P_j :** the old set of APs is still representative of the set of port predicates, but some predicates could be redundant.

To minimize the set of APs we consider all the APs that represented P_j : of these the ones that are not used in the description of other gate predicates are eliminated. The complete algorithm is described in [484].

Improvements Introduced

With the update algorithm, we can maintain a temporary tree in the event of a rule update. Reachability queries thus can be answered anytime without waiting for the APs to be updated.

Updating, taking 10 ms on average in our test-beds, occurs concurrently. The temporary tree is expanded with the new unresolved predicates, causing a slight decrease in efficiency as for [243].

If the unresolved predicates are in a number less than a certain threshold (configurable) and there are no changes of APs it is sufficient to calculate the AP representation of the new predicates. Vice versa if the number of unresolved predicates exceeds the threshold or if the set of APs has changed, the temporary tree is eliminated and a new one is calculated.

Some limitations of the FEC-based approach emerge at this point. First of all, the network model excludes a priori any device that modifies packets, cutting out widespread cases such as NAT, MPLS, IP-in-IP tunnels. However, the theory has been extended to include some types of changes in [485], although, to the best of our knowledge, a version of this upgraded library is not yet available. Furthermore, as we have seen, the forwarding rules only use the IP address to construct the match predicates. A solution for the application of the method to the SDN consists in the use of the headers indicated in the OpenFlow specification, instead of just the IP address. Finally, considering the BDDs, [67] suggests the use of a more efficient data structure for the representation of predicates, attributing to BDDs an overhead due to the excessive number of calls to library functions. It is also known from [15] that the BDDs are sensitive to the ordering of the variables: an incorrect ordering can lead to a sensible efficiency reduction of the system.

2.3 Time synchronization between devices

The field of time-synchronization between electronic devices might appear trivial at a shallow glance, but it has many caveats. The typical example is the clock drift: two digital clocks, initially synchronized with a reference time $T1 = \tau$ and $T2 = \tau$ will not increment their internal time at the same rate. That is, defining $\Delta_\tau = T1 - T2$, it may happen that after some time $|\Delta_\tau| \gg 0$. These sort of problem demands some form of distributed communication among nodes to be synchronized.

The Precision Time Protocol (PTP, IEEE-1588) [211] was proposed in 2002 to provide time synchronization for industrial machines and cyber-physical systems. Dedicated hardware is used to reach high resolution and precision. PTP plays a fundamental role in the new IEEE 802 standards on Time-Sensitive Networking (TSN) and includes protocols for packet replication, time synchronization and traffic prioritization [158].

The TSN standard specifies a new kind of IEEE 1588 protocol called Generalized Precision Time Protocol (gPTP), introducing a specific profile of IEEE-1588 PTP which defines time constraints and maximum jitter between packets. Although the standard specifies a number of security aspects (i.e., an optional internal security layer to sign packets with a MAC), neither key distribution nor application-layer authentication are addressed. Prior work on security of PTP focused on attacks on the synchronization mechanism, such attacks are expensive as they require continuous traffic

manipulation [321]. Other aspects of PTP security, such as the management protocol, were not assessed for their security before. In particular, security assessment of management schemes in practical PTP deployments are difficult, as those measures are implementation-dependent. So far, no framework or generic infrastructure is available to perform such general security assessments of PTP implementations.

In this work, we design and implement a PTP testbed for security assessment, and use it to examine the IEEE-1588 PTP standard. We emphasize that implementing a testbed for this sort of infrastructure is challenging either because of closed implementations in specific products or because of lack of implementation details when reporting it [36, 127, 29]. Our testbed implementation is based on the Linux PTP reference implementations and the whole code used to develop it, is released under an open source license.

As a result of our security assessment leveraging our test testbed, we conclude that most of existing products are all stemming from the Linux PTP implementation, and the MAC-based authentication specified in the standard is in general not used. Instead, the solutions choose their own approach to isolate security-relevant API calls, and some do not protect the API calls at all.

We propose a set of attacks and implement them in the testbed, which are successful in altering the clock synchronization using only management frames. In particular, we show that Type Length Value (TLV) frames can be abused by an attacker to reconfigure, manipulate, or shut down time synchronization. TLV frames are defined in the standard to manage the infrastructure. To the best of our knowledge, the security threats posed by such manipulation of TLVs have not been addressed in the literature so far, in spite of requiring minimal bandwidth and allowing the attacker to gain useful advantages to break time-related applications, such as Public Key Infrastructure services.

In this work, we provide an extensive analysis of how TLVs can be abused by an attacker, and demonstrate the effectiveness of such attacks on our open-source-based testbed.

The contributions of this part of work are:

- We show that the PTP standard fails to define appropriate security requirements for TLVs, although PTP will be critical for infrastructures such as 5G. As a result, implementations can be expected to have security issues related to TLV authentication.
- We propose several different attacks that leverage this lack of security requirements in the standard, and are aimed at desynchronizing clocks and introducing clock drifts.
- We design and implement a PTP testbed using *LinuxPTP* [111] and *PTPD* [350] that allows to reproduce the complex PTP setup in a virtual environment.

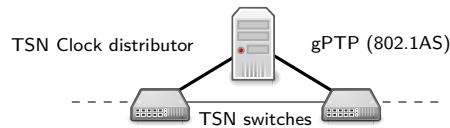


Figure 2.2: A simple example of TSN network using the specialized PTP protocol (IEEE 1588), with a specialized profile called gPTP (802.1AS).

IEEE-1588 (PTP) and TSN

Every systems which operates according to a specific time frame, used to coordinate remote actors, suffers from the clock drift phenomenon. Specific synchronization protocols were designed to this end: the Precision Time Protocol (PTP), standardized as IEEE-1588 being one of the most significant examples. There are three major releases of PTP: IEEE-1588 2002, v1.0, which was the first version, currently deprecated; IEEE-1588 2008, v2.0, which is, at the time of this writing, the main version of the protocol; IEEE-1588 2019, v2.1, which is a new version, typically found in the industrial devices now in production. This protocol family corrects the error introduced by clock drift through continuous handshake between the devices that must be kept synchronous.

To achieve high resolution and precision, a hardware device with packet-time-stamping capabilities can be used. It will process network communications using a high precision clock, without passing the packets through any further stack layer that can alter the effective time and reduce the determinism of the process. The application level of PTP encompasses many PDUs. In this thesis we will focus on PDUs which can be used to configure parameters of the nodes. This family of data units is referenced as Type Length Value (TLV) due to its packet form 2.3.

Currently, TSN standard exploits a specific profile of PTP that is called Generalized Precision Time Protocol (gPTP). A device compliant with this profile must communicate with PTP using Ethernet packets. Which is a constraint introduced to ensure low jitter and maximum delays between two synchronized nodes. Figure 2.2 describes the protocol placement of IEEE-1588 in a TSN with two enabled switches. The clocks of the switches use a single logical time source – pictured as the TSN clock distributor – which grants to the leaf switch devices synchronization references using gPTP.

TLVs are the management PDUs of PTP and are be used to configure the protocol settings. For instance, TLVs can be used to set or retrieve the name of the node set by the network administrator or get the PTP version implemented by the device. Also, these methods, can be used to set critical parts of the systems, such as the state of the port (e.g., by disabling it) or the position of the node in the hierarchy. This part of thesis aims to be a reference for the main security implications of the TLVs of PTPv2.0. A detailed list of TLVs usage can be found in the standards [211] and their

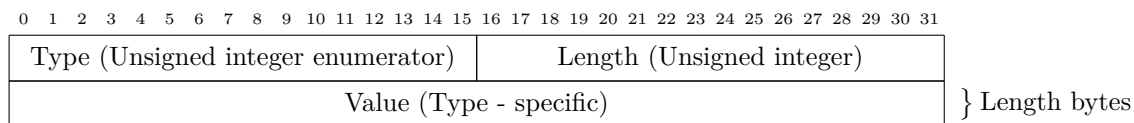


Figure 2.3: TLV packet formats. The mapping between the type name and 16-bit unsigned integer value can be found in the IEEE standard. Type and Length are represented by two unsigned 16-bit integers. Value interpretation is dependent from the Type field and its length is based on the homonym field, interpreted as the number of octets.

security classification can be found in Appendix B.

The PTP and 5G

The 5G standard includes Mobile-Edge Computing (MEC) [240]. MEC is a common standard to exploit virtualization mechanisms such as Docker, Kubernetes and IaaS clouds. These technologies, are employed to provide access to applications designed for the mobile network. This approach allows deployment of user-oriented applications in the network at the most suitable location, alleviating the need for global routing in service provisioning. A common platform facilitates the deployment of applications, that can use a set of standardized APIs. These APIs encapsulate the common infrastructure's properties that are accessed by applications, for instance, methods to search for other registered services with specific features. One of these methods is the query of the wall-clock time and the status of time synchronization with the other applications. These features, as specified in the ETSI MEC standard [6], could be implemented by using standardized protocols, such as Precision Time Protocol (PTP).

PTP Hierarchy

PTP exploits an hierarchical structure, as described in Figure 5.2. Here the Grand Master is a logical node in charge of synchronizing the entire hierarchy.

While a least one Grand Master is always required, the presence of additional Master nodes is optional. Additional Master nodes are in charge of the distribution of the Grand Masters' clocks to the lower layer of the hierarchy. The Ordinary clocks are leaf nodes with no responsibility on redistributing the clock to other part of the network. On the other hand, boundary clocks are bridge devices which interconnect parts of the network and keep their internal clock synchronized with the upper layers of the hierarchy. The *transparent clock* shown in the figure is a passive bridge of the network which is limited to the transmission of network packets. This kind of 'clocks' do not take part in synchronization and do not have internal references. To elect the hierarchy, an

algorithm called Best Master Clock Selection (BMCS) is employed. This algorithm uses parameters configurable in the PTP devices of the infrastructure which will take part to the selection.

The IEEE-1588 standard defines a *Clock manager* entity which is not part of the clock hierarchy. This manager is responsible of the configuration of the network using Management frames (TLVs). These configurations, in IEEE-1588-2008 are, in order of priority, from the most significant to the less significant one:

1. *Priority1*, an integer value configurable by the Clock manager which identifies the priority of the clock.
2. *clockClass*, a integer which identifies the synchronization's type of the clock, this value is configurable by the manager.
3. *clockAccuracy*, an integer which specifies the time resolution at which the clock can operates, configurable by the manager.
4. *offsetScaledLogVariance*, a value that specifies the precision of the clock when not synchronized to other ones, computed by the single nodes.
5. *clockIdentity*, an unique identifier associated with every clock. In a way like Bridge ID in STP for root selection, this is used to resolve tie break among master nodes which have equal proprieties.

The BMC algorithm messages are protected in the same way as the other protocol PDUs, therefore they can be tampered with, producing fake messages, if the network is not segmented or isolated.

Use Case Scenarios

These synchronization can provide to applications low latency requirements such as video and audio streaming in real time or be the core element of safety-related scenarios.

One of these examples (pictured in Figure 5.1) is safety systems composition. Let's consider devices that can analyze spatial regions for intrusions¹. A distributed interaction between such systems can enable two-dimensional vision which can be extended over obstacles providing a more complete view on the observed space. This kind of systems needs a fine-grained time synchronization to build a consistent image and provide the correct measurements. The maximum delay and jitter allowable in a network vary according to the safety level which one wants to obtain [56]. Scanners can therefore exploit synchronization mechanisms provided by the IEEE-1588 protocol.

¹For instance, laser scanners.

Another scenario in which time synchronization could be employed to gain advantages is the Audio-Video Distribution. In this case, multimedia plays from different systems must be synchronized. When clock drifts occurs, the delay between the two plays could be heard by the audience.

PTP and other time synchronization protocols

Previous works in the field of timing synchronization have their roots in the secure time synchronizations protocols, as [166, 167].

The Precision Time Protocol security has been extensively explored by several works: starting from Ullmann's work which explores this protocol compared to NTP [452], culminating with recent works on the security of the PTPv2.1 standard [195, 405]. As these works state, the clocks alterations that can be introduced controlling the infrastructure will still be present in the last PTP standard. Several options that can improve PTP security could be found in works by Moussa et al. [322] and Neyer et al. [332]. These papers address the field of time synchronization and encryption, a theme which will return in the recently emerging TSN infrastructures. As stated by previous works [327], this standard do not specifies any security profile and, like the PTP protocol, relies on specialized standards such as MACSec [312]. This document states also the possibility of interoperation between the TSN network synchronization capabilities and the emerging 5G network infrastructures, focusing on the Ultra Low Latency layer of the communication platform. On a different part of this mobile 5G network infrastructure, time synchronization protocol support can be found in the MEC platform [240].

Finally, most of the presented work is based on the official standardization documents of PTP protocols [210, 211, 212], TSN [213] and 5G MEC API specification [6].

PTP Security

From a security perspective, the PTP protocol can implement a security layer to sign packets with a Message Authentication Code². The distribution of the key is not specified in the IEEE-1588 standard. Moreover the protocol can be also tunneled in secure layers like IPSec. These solutions are far from optimal for a clock distribution service as stated in [452]. Controlling the network infrastructure hosting the PTP communication, attackers can control the speed of PTP PDUs: they can delay or accelerate the clock synchronization packets³, even acting on different ISO/OSI layers.

Exploiting this kind of delays will therefore control the time reference of the victims.

²Standardized as an option in the Annex-K of the IEEE-1588 2008 document [211]

³While delaying packets is trivial, Ullmann et al. in [452] describe a method to accelerate packets by exploiting changes on the infrastructure itself by using faster routes.

PTP attacks

A family of possible attacks to PTP are discussed in [452], where a list of vulnerabilities that affects PTP networks is presented. The main attacks on this kind of networks are classified as:

Byzantine masters. Spoofing a Master node, the dependant clocks can be induced to a time drift or sensible clock skews. If the system is misconfigured, e.g., with max correction applicable to the clocks, ϵ . These configurations can be broad enough to be bypassed by an attacker with active-attack possibility. If no limit is imposed on ϵ , the clock can be manipulated to the point to make it faster or slower than others. That is, an attacker can announce an arbitrary clock offset to the clients, changing all the hierarchy's references.

Boundary clock alteration. An interposed device (boundary or transparent clock) can alter the timing between the packets. This operation can be made on data-link or networking layer, using network routes or dropping packets with a certain rate. These manipulations can effectively change the heuristics used in the BMC selection algorithm or the synchronization between the clocks. Encrypted connection do not mitigate the possibility of this attack [34].

2.4 Insider Threat

Insider threats pose a complex challenge. They represent one of the most expensive security issues for companies [417] in general. Furthermore, they have the potential to obliterate companies working in specific fields where customers' trust is the tool of the trade. A prime example is the offensive security sector, e.g., firms offering penetration testing services, which have a mission to uncover unbeknownst ways to access sensitive data or to interfere with the processes of their customers.

While initial vetting of prospective employees and a sane trust relationship built on close cooperation remain fundamental, it is only prudent to deploy technological aids to detect possible malicious behaviors and to contain their effects.

In a classic cybersecurity framework, e.g., NIST's, five main phases are defined: Identification, Prevention, Detection, Response and Recovery [340].

The first 3 phases are obviously those related to the analysis and monitoring of the behavior of all the agents in the game, trying to discover a malicious one. After that, the Response and Recovery phases focus on how to react to the attack once it is identified, to contain damages and restore normal operations.

When dealing with insider threats, not all phases are created equal. Prevention would require to know in advance which actions – among the ones legitimately granted to insiders to perform their duties – can be used to perform an attack, which is clearly impossible to scale out of the

simplest scenarios. Response and recovery are next to pointless after highly sensitive data have been exfiltrated or devastating vulnerabilities have been exploited.

Identifying insider threats usually requires a mix of detection mechanisms. While for other common threats such as vulnerability exploitation, human exploitation, etc. it is possible to define what a malicious activity is, insider threats require a monitoring phase which must tell malicious from non-malicious intents often appearing as similar activities. Insiders can perform actions that are “syntactically” legitimate but may have a malicious semantics, such as the usage of external USB devices. For this reason, traditional activity tracing must be complemented by a monitoring approach including deep understanding of the context and observation of the user behavior.

Detection tools - deep packet inspection

Deep Packet Inspection (DPI) is considered one of the main techniques for traffic analysis. It resembles packet filtering in that it examines traffic transparently traversing an inspection point. Instead of noting the headers only, it takes a holistic view at the packet data, looking for protocol non-compliance, attempted malware delivery or possible exfiltration of sensitive data.

Result of the DPI can be corrected, marked for the purposes of quality of service, blocked, at a limited rate and/or reported to a reporting agent in a network.

DPI is resource-intensive; it is experiencing an increasing diffusion thanks to the advent of modern virtualized infrastructures, enabling its provisioning as a softwarized component (a Virtual Network Function, or VNF), which can be easily deployed or even invoked “as-a-service”. Authors in [381], for example, proposed an Efficient Verifiable Deep Packet Inspection scheme (EV-DPI) with privacy protection over two non-colluding cloud servers. The DPI was implemented as a service into an outsourced MiddleBox (MB).

Similar work has been done in [187], which is focused mainly on how to create an architecture acting as an enabler for a DPI service.

Also [270] proposed two practical approaches for the implementation of a cloud-based DPI MB. Its services can be invoked to perform payload inspection over encrypted traffic while preserving the privacy of both communication data and inspection rules.

Detection tools - behavioral analysis

Recently, thanks in particular to the advent of Machine Learning techniques, significant work has been done to map the nature of an internal user using advanced AI analysis. This approach looks promising to limit the rate of false positives, which is a typical shortcoming of simple activity detection methods.

Already in 2016, authors in [392] listed several proposed techniques for insider threat detection based

on machine learning classification. Shortly afterwards, an interest emerged for the application of these techniques as a means to integrate environmental and behavioral parameters into the detection process. In [63], insider threat is detected through AI-based behavioral analysis. More specifically, [393] proposed an insider threat detection model that performed monitoring through AI techniques for facial recognition. Another integrated approach to monitoring is shown in [100]. Its authors studied the effect of combining the usual technological techniques with social, political, legal, and cultural influences considerations from disciplines in human behavior. Lastly, [309] proposes a method to identify specific malicious patterns and to predict a user's next move, based on a deep analysis with regression methods.

Contextualization and integration

Literature abounds with guidelines and principles aimed at providing general descriptions of the context and the identity of the insiders [162, 90]. However, experts agree that the strong contextual variance of threats [88, 397] makes providing a general yet precise identification of all possible insiders difficult.

So it's important first to build a set of rules or guidelines on how we could define the malicious behavior of an insider. In [181], for example, authors describes approaches to assemble knowledge about insider threats and to apply this knowledge in support of insider threat assessment.

Similar work has been done in [148] where authors defined a strict methodology to identify insider threats based on a better collaboration process between the Information Technology (IT) management and the Human Resources (HR) department. In [182] authors proposed a research on a comprehensive ontology of sociotechnical and organizational factors for insider threat.

One of the most recent works on this same approach is [183]. This research aims to catalog human as well as technical factors associated with insider threat risks to inform the development of more proactive approaches to insider threat assessment.

In [423], the detection of the insider is based on a real-time testing simulation of real users, generating user data to test the detection of malicious users. We found other updated and relevant resources in [204] which is one of the latest surveys that summarized techniques for insider threat identification and detection.

2.5 Constraint Programming

We will use extensively constraints and Constraint Programming in policy verification.

In Constraint Programming (CP), problems are solved by defining the requirements (constraints) to be applied to the problem variables and the goal is to find a solution that satisfies all the con-

straints. [385]. The main idea behind CP as stated by E. Freuder is: “Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.”

Therefore CP is very suitable to implement a declarative northbound interface, that network operators or network users may exploit to express general constraints and goals in a way that is not bound to the specific underlying technology [165].

The essential step in modelling a real-world problem as a CP model, which can be solved using CP techniques, is to determine the decision variables of the problem and their relationship in terms of constraints, representing in very general terms the restrictions or cross bounds on values that all decision variables can have. Solving a CP model is the action of finding the values of the decision variables that simultaneously satisfy all the constraints. In this case CP problems are also called Constraint Satisfaction Problem (CSP) [49].

However, in many cases, there may be many subsets of the variables domains that satisfy the constraints. The solver program can be tuned to provide the first solution found, without any further processing, or all possible solutions. In addition, if an objective function can be defined on decision variables, the solver program can also be asked to provide a subset of values that maximises or minimises such a function. In this case CP problems are also called Constraint Satisfaction Optimisation Problem (CSOP) [49].

High level languages exists to state problems and constraints, like the open-source Minizinc language⁴, as well as high performance solvers that can solve problems in a very efficient way [75]. Indeed the performance of a CP solver depends on its implementation but also on the description of the problem, with a mixed combination of computation efficiency and human optimization in problem design. The ease of implementation, simplicity, expressiveness and compatibility with many solvers has made MiniZinc the de-facto standard CP modelling language.

⁴<https://www.minizinc.org>

Part II

Contributions

3

Analysis

One of the points of our research was the analysis of the families of attack present in literature, that composes a good state of the art of the micro-services attacks. It can be useful to analyze a future direction for attacks or defences (e.g., where the defences are lacking).

3.1 Microservices security

The importance of security in micro-services creates the need for understanding and analysing the state of the art for securing this kind of architectures. It is particularly important to understand which problems are especially relevant for micro-service systems, and how existing techniques can contribute to addressing them. However, there is still a lack of systematic investigations of studies at the intersection of security and micro-service architectures.

We followed a structured approach, which led us to select and gather 290 peer-reviewed publications. At the time of this writing, this constitutes the largest curated dataset on the topic. We first perform a quantitative analysis on the metadata of the publications, for example, publication outlets and keywords. This provides insight into the communities and key research concepts that currently characterise the field. We then map each publication to a vector of 20 different markers, corresponding to 20 research questions on micro-services security that we formulated based on established security techniques and the field of micro-services as a whole.

Our research questions focused on threat models, security approaches, infrastructure, and development approach. We perform correlation analysis to show that our questions are well-posed (independence), and also to confirm that some topics correlate positively (e.g., Intrusion Detection and Intrusion Prevention, and Agile Development and DevOps as well). Findings from our analysis include: issues with technology transfer from academia to industry on micro-services security; lack of guidelines for adopting security by design in micro-services; lack of appropriate threat models; lack of guidelines for addressing the attack surface given by technology heterogeneity; and security

issues when migrating systems to micro-services. Our data, findings, and discussions form a useful basis for orienting future developments of the field.

In summary, the main contributions of this part of work are:

- the characterisation of Micro-services Security as an early-stage, growing research field in need of systematisation and more mature contributions (“Publication Outlets”, “Types of Publications”);
- the identification of the main research communities on the Micro-service Security field and the clustering of authors (Research Communities);
- a presentation of the trends of the main security attacks involving micro-service architectures, both from the points of view of threat model (Threat Model) and mitigation (Security Approach (Mitigation));
- a report on the current infrastructural security solutions for micro-services (Infrastructure) as well as the interaction between the main micro-services development approaches (such as DevOps and Agile) and security (Development);
- a correlation analysis of the answers to our research questions in papers, which sheds light on relationships among the different aspects of micro-service security (Correlation between Research Questions);
- a summary of the main open challenges that emerged from our study, which form a call for action for the community of researchers and practitioners working in the field of micro-service security (Discussion and Future Directions).

[455] present a systematic mapping that identifies the security mechanisms used in microservice-based systems. Contrary to our work, which provides a general overview on the state of the art of microservices security, the authors narrow their focus on cataloguing the security technologies and mechanisms adopted by developers of microservice-based systems—e.g., authentication and authorisation—leaving out other subjects related to security, like threat models and development methods. Similarly to [455], [24] concentrate on surveying the technologies and standards for security, privacy, and communication used in the area of microservice architectures in the cloud.

Extending our view to articles that, at the time of this writing, are not available as peer-reviewed publications, we mention the work by [197] and [363]. [197] present a systematic categorisation of threats on microservice architectures and propose a selection of possible mitigations. [363] look at how “security smells” affect microservice-based applications and how to mitigate the effects of such smells through refactoring. As for the proposals by [455] and [24], the difference between our work

and [197] lies on generality: [197] narrow their investigation down to the threats identified in the literature. Similarly, the work of [363] focuses on the programming of microservices.

In addition to the related work discussed above, there are quite a few neighbouring surveys with respect to our work that are interesting to discuss: while these studies are not dedicated to the topic of microservice security, they explicitly mention security as an important concern for microservices in different contexts—software engineering, Internet of Things, containerisation, etc. The purpose of reviewing neighbouring related work is twofold:

1. It shows the multifaceted nature of microservice security, giving concrete evidence of the need for an investigation which is both wider and deeper, as we do in this work.
2. It provides a general overview of the challenges and possible uncovered research topics related to security in microservices—which inspired some of the questions presented in Section 3.1.

[143] present an overview of microservices, including a discussion of the origins of the paradigm, its state of the art, and future challenges. They identify a number of trust and security challenges posed by the paradigm. We mention a few examples. Service reuse, one of the key benefits pushed for in the microservice paradigm, requires adopting secure mechanisms for service authentication and authorisation. The increased granularity and heterogeneity of microservice architectures extends considerably the attack surface of these systems. The sophisticated DevOps infrastructure required to operate microservices effectively is a new attack vector.

[169] conducted a preliminary analysis toward a taxonomy of microservices architectures. While not addressing in particular security concerns, [169] reports that the security subject is not extensively addressed, highlighting how monitoring and microservice communication trust chains should receive particular attention.

[228] reviewed approaches proposed in the literature to deal with the various concerns of microservice-based systems. The authors mention the large attack area offered by microservices subject to insider/privilege-escalation attacks and network security issues.

[93] surveyed the topics of European research projects in the area of software engineering. Regarding microservices security, they highlight four main challenges: increasing the usage of software validation and verification methods; improving the trust and interoperability of services through (self/federated)-certification of outputs based on standards; adopting a security-by-design approach on the whole software lifecycle; and helping developers with addressing discontinuities in the chain of compositionality between services and execution environments—e.g., due to data leakages derived from fragile container-host interactions.

[276] investigate and discuss the challenges of migrating monoliths to microservices. They observe that security should be part of the migration planning phase to begin with, and that developers

need models and frameworks to help them elicit, track, and manage the (frequently implicit) assumptions and invariants induced by the migration of the legacy system. These observations are shared with [135], who suggest that the microservice architectural style has a direct impact on the design of a system and that researchers are still investigating how to leverage its characteristics with respect to system quality and security. [135] note that there exists uncertainty about the realisation of microservices, indicating the need for comprehensive references to help programmers in the multifaceted aspects of microservice development.

[343] address the open challenges of interoperability in the Internet of Things (IoT), noting how microservices can constitute a solution for the programming of highly distributed IoT networks and provide two decades worth of research and industrial experience to tackle interoperability in heterogeneous systems. Regarding the general security of IoT systems, [343] note the emergence of security issues (e.g., authentication and access control) when system design permits direct access to resource-constrained devices. Reviewing the many solutions and levels at which IoT interoperability can be tackled, [343] note the challenge of both maintaining and guaranteeing the same level of security when mediating among different technologies.

[299] examine microservice availability tactics to detect, prevent, mitigate, and recover from faults. They highlight how the tactics for the availability of microservices mainly focus on preventing faults, whereas detection, reaction, and recovery are scarcely addressed. Commenting on related challenges, [299] report a deficit of solutions to support the restoration of normal functionalities after a microservice architecture suffered from some faults.

[13] surveyed robust and flexible service management platforms for IoT systems. Like [343], they identify microservice architectures as the most suitable architectural pattern to handle the heterogeneity of IoT systems and that the foremost challenge in the field is the robust integration of different technologies. [13] also report how conventional security solutions and practices are not suitable to handle the expansion, mobility, resource constraints, and new security requirements of the considered systems.

[95] investigate service integration from the perspective of separation of concerns and identify problems with conventional service integration design/technologies. They report that the lack of proper cross-cutting concerns in programming technologies make it difficult to capture and guarantee that invariants of a given microservices—specifically, on security—hold when paired with integration components.

[488] survey how cloud computing systems can help scientific research. In their report, they notice how the (micro)service paradigm is useful to make resources available to collaborating researchers by providing a well-defined interface specifying the operations that can be performed on, or with, a given resource. However, they also report that privacy and trust issues are of particular concern to researchers, especially in fields that are processing sensitive data such as medical

research. For this, appropriate provenance metadata is required, both to understand how and by whom the data was created and modified, as well as to understand where it has been potentially exposed to corruption. Similar comments are shared also by [361] in the context of healthcare cyber-physical systems. In particular, proper encryption is reported as a key component for (real-time) data acquisition.

[413], reviewing the “pains and gains” of microservices in the grey literature, found how security generates pains at design-time. Like [488], [413] comment that microservice-based applications should support the consistent determination of the provenance and authenticity of data, noting the paradox of that being in contrast with the heavily-distributed nature of microservice systems. Another (meta)observation by [413] is how there is a gap between the industrial understanding and state-of-practice on microservices and the state-of-the-art of academic research, one possible reason being that academics have limited access to industrial-scale microservice-based applications.

[134] identify, classify, and evaluate the state of the art on architecting with microservices from the perspectives of publication trends, the focus of research, and potential for industrial adoption. On security, they report that it is attracting insufficient research. The works by [463] and [25] follow similar modalities and results.

[55] surveyed security of containers, a technology frequently paired with microservices. They report how container security is still in an early phase and it faces unsolved challenges. The results presented by [55] match those by [424], who report the presence of a large number of challenges linked to containerisation because OS Kernel sharing introduces security issues absent from virtualisation solutions.

[424] also highlight the importance of enhancing vulnerability management, digital investigation, and container alternatives.

[368] present a survey on the employment of fog computing to support IoT devices and (micro)services. In their study, they report how security is the largest cross-cutting technical concern within critical IoT systems, which necessitates a common baseline and interoperable standards to address security challenges within both hardware and software. In particular, [368] advocate for solutions to provide a full-stack secure chain of trust from devices to fog/cloud components, which has been only preliminary explored (as remote attestation techniques). [446] and [368] report also the importance of addressing the concerns of context-aware security (in IoT systems), especially for authentication and authorisation.

Also [495] surveyed the literature on microservice-based fog applications to elicit the security risks threatening them. The main threats highlighted include: kernel-level leakage vulnerabilities linked to containerised deployment; man-in-the-middle/insider attacks on data-transmission interception; the need to verify when services become compromised/misbehave; and network-level vulnerabilities on data-routing alteration.

Publication	Year	Type	Num.	White L.	Grey L.	Sources
This work	2021	SLR	290	●	○	ACM Digital Library IEEE Xplorer SpringerLink Scopus Science Direct Wiley Google Scholar
[455]	2019	SLR	26	●	○	ACM Digital Library IEEE Xplorer SpringerLink Science Direct Wiley Google Scholar
[24]	2017	Survey	N.A.	●	○	N.A.
[197]	2020	SLR	46	●	○	ACM Digital Library IEEE Xplorer SpringerLink Science Direct Wiley
[413]	2018	SLR	51	○	●	Google Bing Duck Duck Go Yahoo! Webopedia

Table 3.1: Summary table and comparison with related works. For each row/work in the table, we report: its reference; its publication year; its type (Systematic Literature Review (SLR), survey, etc.); the number of publications it encompasses; whether it analyses white (peer reviewed) literature; whether it analyses grey (blog posts, etc) literature; the sources it used to search its dataset.

The table 3.1, shows the differences between these various works, in numerical and boolean terms. As clearly evincible, our work expands the previous works by adding a conspicuous amount of analysed publications; using white literature at its roots and following the trend and methods of the main Systematic White Literature Reviews.

Methodology

Following the guidelines by [411], and as depicted in Figure 3.1, we started by searching and retrieving the literature for relevant publications from several data sources by using the same keyword query. We then performed a manual revision process of the automatically selected publications to exclude publications out of the scope of this study and perform snowballing—i.e., recursively adding to the dataset relevant publications cited by the already selected publications. The result-

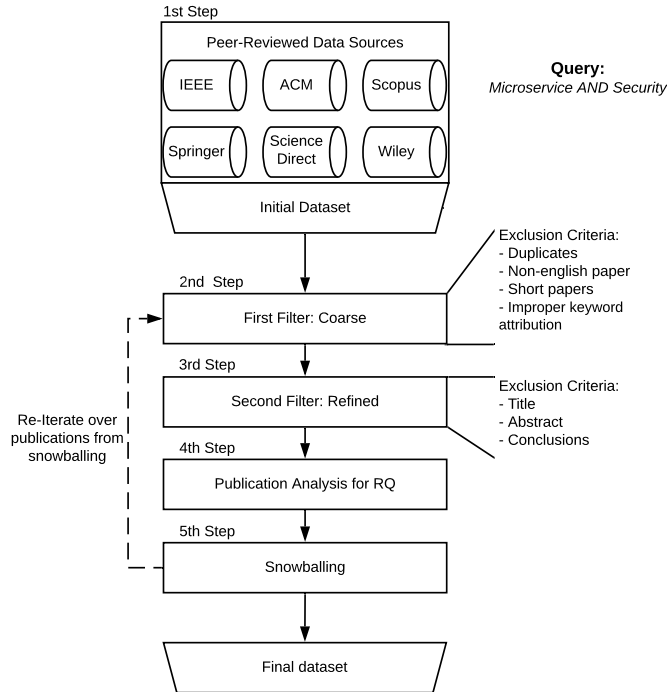


Figure 3.1: Schema of the method followed to gather the dataset for this review.

ing dataset consists of 290 publications. We analysed these publications to collect statistical and transparent answers to our research questions, which are detailed in Section 3.1.¹

Security in microservices includes complex and heterogeneous topics, ranging from development to infrastructural concerns. In our choice of a selection query to gather an initial dataset, it was important to pick a sufficiently general query. For this reason, we adopted the query “Microservice AND Security” for our initial search, capturing all the publications containing both terms in any of their title, abstract, or body.²

[134, 361, 413] reported how publications on the topic of Microservice started in 2014. Taking into account this fact, we limited our research to contributions published since 2014.

During the seven years covered by our work, the body of knowledge on this topic has grown significantly. For this reason, we deemed it useful to consider white literature only: in terms of

¹The list of the publications and their bibliography information is publicly available at <https://doi.org/10.5281/zenodo.4774894>.

²We performed experiments with potentially more inclusive queries, such as “Microservice AND (Security OR Authorisation)”, as well. The tried queries, however, did not extend the search in any useful way since the term “Security” proved to be general enough to cover specialised aspects like authentication, authorisation, and (safe) communication.

quantity, it represents a very meaningful sample of the research produced during the considered time frame, and in terms of quality, it allowed us to rely on peer review. Thanks to the more uniform organisation of white literature, we are also more confident in the level of consistency of our choice and application of the selection criteria. This is not to say that grey literature is not worth investigating. Blog posts, personal websites, technical reports, white papers, etc., are often the preferred venues for practitioners to share ideas. However, as also pointed out in [412], “it is very difficult to uniquely measure the quality of grey literature when conducting a systematic, controllable, and replicable secondary study” and we are not aware of a standard method for the evaluation of grey literature. Analysing the grey literature was beyond the quality goal of this article and we leave it as future work.

Accordingly to this strategy, we collected publications from 6 different publishers, focusing on peer-reviewed publications. We did not, for example, use Google Scholar or arXiv, since they also list resources that are not peer-reviewed.

- ACM³, 478 publications;
- IEEE explore⁴, 181 publications;
- Springer⁵, 345 publications;
- Scopus⁶, 134 publications;
- Science Direct⁷, 358 publications;
- Wiley⁸, 208 publications.

This gave us an initial dataset of 1,704 publications in total. We collected publications published up to the 31st of December 2020, using the academic subscriptions provided by University of Bologna and University of Southern Denmark. To guarantee the same level of trustworthiness and authenticity, we retrieved the publications only from the official entries, avoiding external sources such as the authors’ personal websites.

³<https://dl.acm.org/>

⁴<https://ieeexplore.ieee.org/>

⁵<https://link.springer.com/>

⁶<https://www.scopus.com/home.uri>

⁷<https://www.sciencedirect.com/>

⁸<https://onlinelibrary.wiley.com/>

Publications Triage

The publications retrieved from the publishers were processed in three steps to check if they should be excluded according to distinct exclusion criteria. Graphically, in Figure 3.1, these steps are labelled as 2nd, 3rd, and 4th Step(s).

In the 2nd Step, we looked at whether the keywords “Microservice” and “Security” were used. We excluded a publication if the keywords appeared only in the bibliography. Moreover, we excluded the publication if it was too short (less than two pages), publications not written in English, and duplicate publications already listed in another publisher source.

In the 3rd Step, we looked at the title, abstract, and conclusion of each publication. Publications that do not treat or discuss topics related to micro-services and security were excluded. In this step, we also excluded publications in which the security topic was orthogonal or incidental. In this way, we excluded publications where “microservices and security” was one of the possible application scenarios, but not the main subject of the study.

We also excluded cases in which the work tangentially mentioned the satisfaction of some security aspects, without detailing the design/development of the security technologies to accomplish them. For example, we excluded publications focusing on blockchain technologies where the authors incidentally mention authentication and integrity protection as inherent security properties of blockchain-based implementations.

In the 4th Step, we performed an analysis of the publications, answering to the Research Question (RQ) detailed in Research Questions. No publications were excluded at this step.

At this point, the following publications remained in the dataset (268 in total):

- ACM, 67 publications;
- IEEE explore, 59 publications;
- Springer, 46 publications;
- Scopus, 28 publications;
- Science Direct, 53 publications;
- Wiley, 15 publications

Snowballing

As the last (5th) step for the systematic literature review, we performed a backward snowballing process [473] with the objective of identifying additional relevant references for our study from the works cited by the already selected publications.

All references collected in this way underwent the triage by following the Steps 2, 3, and 4. Each referenced publication accepted for inclusion by these steps was then added to the dataset of selected publications. Snowballing was recursively performed on these newly added publications until reaching a fixed point; i.e., until no new publications were added to the dataset.

The outcome of repeatedly applying the snowballing process led to the following results:

- 40 references in the first round, from which we selected 9 publications;
- 22 references in the second round, from which we selected 8 publications;
- 5 references in the third round, from which we selected 5 publications;
- 4 references in the fourth round, where we selected 0 publications.

The 4 cycles of snowballing yielded 22 additional publications that were included in the dataset to reach the final size of 290 publications.

Research Questions

In this section, we detail the research questions that guided our systematic review. Usually, the research questions for systematic literature reviews are fairly broad and do not amount to more than six. In our case, we chose to adopt more questions (20) but dichotomous (i.e., with yes-or-no answers), to favour precision and objectiveness. To define the questions and seek guidance in categorising the relevant security issues for microservices, we took inspiration from the related work presented in Section 3.1, as well as from the state of the art in standards and methods, namely the NIST Special Publication 800-204 “Security Strategies for Microservice-based Application Systems” [98].

Our questions are collected in four macro groups (Gs), each covering a different concern.

- G1: Threat Model. Questions on threat modelling and how threats are dealt with.
- G2: Security Approach. Questions on the security approach, e.g., whether it is preventive, adaptive, proactive, or reactive.
- G3: Infrastructure. Questions on the infrastructure that micro-services run on.
- G4: Development. Questions on the development process.

The questions in each group are reported in the remainder of this section.

G1: Threat Model

Mapping the usage of threat models is important to see gaps when a security violation must be handled, or if known models are outdated and need to be adjusted. The NIST report, for instance, hints at the importance of identifying the threats looming over a microservices architecture [98].

The usage of a formal threat model has proven to be extremely useful in the identification of attack types and their strategic countermeasures [126].

Several threat models exist in the literature. The most famous one is STRIDE [247] named after the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege security threats. Other threat models however exists, such as PASTA [451] or OWASP [349].

In our review and with this first group of questions, we aimed to understand whether a publication followed a known model, strategy, or guideline. Alternatively, we wanted to know if new security models were proposed.

This group consists of the following questions:

- Q1** Does the publication mention STRIDE, or at least consider all of its aspects?
- Q2** Even without explicitly mentioning STRIDE, does the publication involve at least one of its aspects (Spoofing, Tampering, ...)?
- Q3** If STRIDE aspects or equivalent are considered, does the publication propose/ discuss a concrete implementation/solution (either developed by the same author or one taken from the literature)?
- Q4** Does the publication consider or follow another threat model rather than STRIDE without introducing a new one?
- Q5** Does the publication mention policies, workflows, or guidelines to handle violations?

In particular, with question Q1 and Q3 we looked for the adoption of STRIDE, being the most popular threat model. In the remaining questions, we investigate if the publication defined some threat model—either from the literature or a newly one introduced in that publication—or at least discussed equivalent principles or guidelines without mentioning STRIDE.

G2: Security Approach

Many related works cite the usage of preventive measures to secure microservices [299, 455, 169, 24, 13, 413] while some indicate the need for further research in the other directions of proaction, reaction, and adaptation [455, 299]. With this second block of questions, we wanted to go deeper

into the security aspects, considering the specific security approaches, solutions, and also the role that micro-services play.

This group consists of the following questions:

- Q6** : Does the publication mention Intrusion Detection System (IDS) functionalities?
- Q7** : Does the publication mention Intrusion Prevention System (IPS) functionalities?
- Q8** : Does the publication mention Threat Intelligence?
- Q9** : Does the publication mention Exfiltration Leaks?
- Q10** : Does the publication address Insider Threats?
- Q11** : Are micro-services part of the solution?
- Q12** : Are privacy and GDPR considered?

G3: Infrastructure

The NIST report by [98] dedicates a large part of its content to infrastructural security solutions for microservices. Similarly, the majority of the mentioned related work in Section 3.1 presents or at least cites infrastructural solutions for security, acknowledging that the infrastructure of microservice systems is typically complex, encompassing concerns that span from service deployment and service-to-service coordination (discovery, composition, consistency) to the definition of security-specific mechanisms (authorisation, authentication).

In this group of questions, we aimed at finding information on the infrastructure configurations considered in the publication. This group consists of the following questions.

- Q13** : Does the publication specify how the proposed architecture is controlled or managed (e.g., in a centralised, decentralised, or hybrid way)?
- Q14** : Does the publication mention Infrastructure-as-a-Service?
- Q15** : Does the publication mention service discovery?

G4: Development

Micro-services are often associated with software development practices like DevOps and Agile [45, 453] which, in turn, are heavily influenced by the inclusion of security-oriented practices [93, 276, 95, 413].

In this last set of questions, we aimed at checking the extent to which these practices are used also in the setting of security, for example by verifying whether specific development processes and security standards are considered.

This group consists of the following questions:

Q16 Does the publication mention DevOps, Continuous Integration, Continuous Deployment, or Continuous Delivery?

Q17 Does the publication mention Agile, or how security experts are integrated from a development process point of view?

Q18 Does the publication mention Domain Driven Development?

Q19 Does the publication mention Model Driven Development?

Q20 Does the publication mention certifications, such as ISO27000 ⁹, or technological standards such as X.509 ¹⁰?

Review Results

In this section, we present the outcome of the literature review. We start by presenting quantitative results obtained from the metadata of the publications in our dataset. This is useful to map the trends over time and the current shape of the field, in terms of the number of contributions, type (proceedings, articles), communities, and keywords (and their relations). Then, we present results derived from the analysis of the types of contributions (theoretical, applicative, etc.) and of the relation between the selected dataset and our research questions (cf. Section 3.1). This part is aimed at providing a detailed insight on existing research patterns, gaps, and uncovered areas of the field. We close the subsection with a correlation analysis of the questions, providing a quantitative look over the relationships between them. For reference, we also report our dataset in tabular form, where each entry is associated with the positive answers given to our research questions.

Insights. In the following subsections, we highlight in separate paragraphs (like this one) the main insights that emerge from our analysis. Each insight motivates an open challenge, which we write in bold as the heading of the insight. We will use these challenges in “Discussion and Future Directions” to structure our discussion about useful future directions for research on micro-service security

Metadata Results. We start our quantitative analysis of the collected dataset by presenting in Figure 3.2(a) the time distribution of the selected publications. As expected, security in microservice

⁹<https://www.iso.org/isoiec-27001-information-security.html>

¹⁰<https://tools.ietf.org/html/rfc5280>

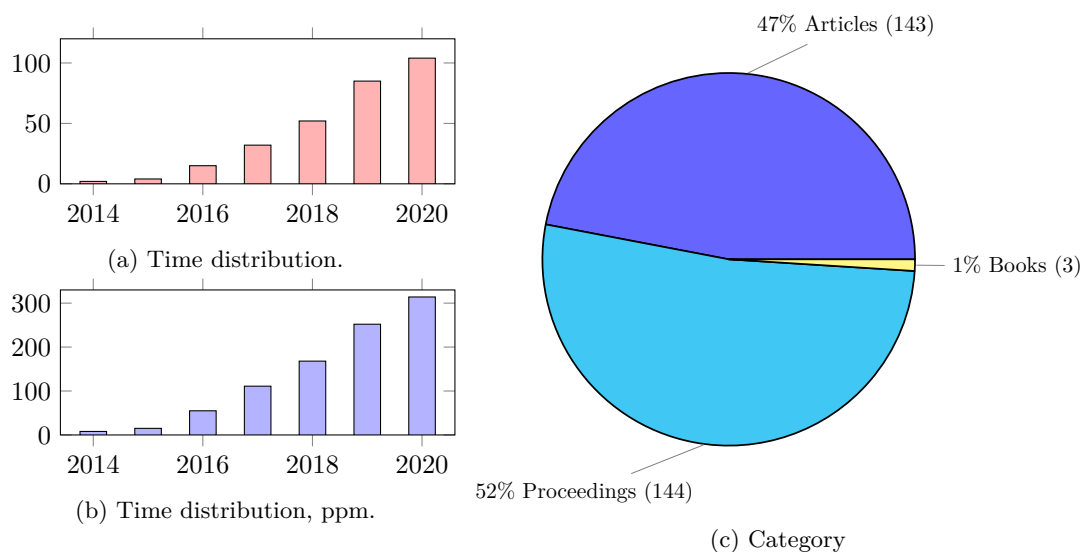


Figure 3.2: Time and category distribution of publications.

systems gained a lot of academic interest in the latest years. This is reflected by the sharp increase in the number of publications since 2014. In Figure 3.2(a), we report the number of collected publications per year. As a reference to indicate the degree of growth of the field, we report in Figure 3.2(b) the yearly ratio (in parts per million) between the collected publications and the overall number of publications in computer science¹¹.

Publication outlets. From the plot in Figure 3.2(c) we see that conferences and journal venues are the most common outlets, while books/collections are underrepresented. This last fact indicates the early stage of the field, where established references are still lacking. However, conference proceedings are almost matched by journal articles, marking a maturing trend of results that are solid enough to constitute material for more structured contributions, as those found in peer-reviewed journals.

We now concentrate on the specific conferences and journals where the publications in our dataset have been published. In Figures 3.33.4, we report this result in two versions: *i*) in tabular form, on the left-hand side of Figures 3.33.4, with the acronym, the full name, and the number of contributions in our dataset of the venues with the most contributions and *ii*) on the right-hand side of Figures 3.33.4, showing the data on the left as a pie chart.

Regarding the distribution of publications over the different categories of venues, we note how the audience of journals and conferences vary. In fact, there is no predominance of security-oriented

¹¹Source: <https://dblp.org/statistics/publicationsperyear.html>.

Acronym	Name	# in dataset
ARES	International Conference on Availability, Reliability and Security	2
CCGRID	IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing	2
EuroS&P	European Symposium on Security and Privacy	2
ICSA	IEEE International Conference on Software Architecture	3
IFIP	The International Federation for Information Processing Conference	3
MEDES	ACM Conference on Management of Digital EcoSystems	2
NOMS	Network Operations and Management Symposium	2
SEC	Security Conference	3
STAF	Software Technologies: Applications and Foundations Interoperable Systems	2

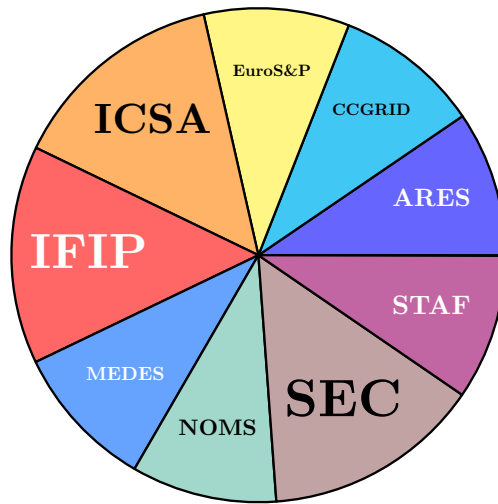


Figure 3.3: Conferences with the largest number of publications in our dataset.

Acronym	Name	# in dataset
CC	Cluster Computing	4
CCPE	Concurrency and Computation: Practice and Experience	4
ESE	Empirical Software Engineering	2
FGCS	Future Generation Computer Systems Conference	7
FI	Future Internet	2
IEEE Access	IEEE Access Multidisciplinary open access journal	5
IEEE IC	IEEE Internet Computing	3
IEEE PDS	IEEE Transactions on Parallel and Distributed Systems	3
IST	Information and Software Technology	2
JSS	Journal of Systems and Software	8
MNA	Mobile Networks and Applications	2
MTA	Multimedia Tools and Applications	2
PCS	Procedia Computer Science	3
Queue	ACM Queue	3
SICS	Software-Intensive Cyber-Physical Systems	2
SPE	Software: Practice and Experience	4
Sensors	IEEE Sensors Journal	3

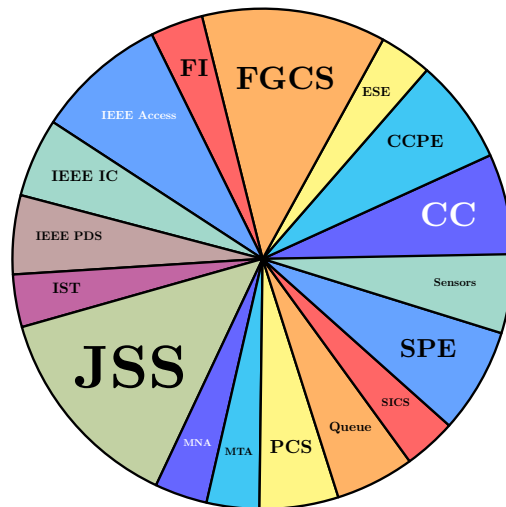


Figure 3.4: Journals with the largest number of publications in our dataset.

or even software engineering venues, which could have been the most likely targets. Instead, the analysed publications appear at venues addressing a broad range of topics, from networking to cloud computing, and on open journals such as IEEE Access and ACM Queue. Furthermore, there is no clear preferred venue that dominates the others, but contributors are rather scattered over many neighbouring venues.

We give a twofold interpretation to the phenomenon. On the one hand, this fact can indicate that micro-service security is perceived as of cross-disciplinary interest, each contribution seeing it from the lens of its specific area (whether it be software engineering, networks, sensors, cloud computing, etc.). On the other hand, we notice the lack of specific venues dedicated to micro-services, and least of all, dedicated to micro-service security.

Insights. Fragmentation of outlets: there are no reference venues for the area of micro-service security (neither journals nor conferences). This makes it difficult for researchers and practitioners to keep up with the state of the art, as well as to find dedicated conventions where they can discuss this topic

Research Communities. To add more insight on the communities of the field, we also perform a network analysis to identify and explore the clusters of the most prolific authors and their research collaborations. Specifically, we are interested in analysing the networks of collaboration of “core authors”, i.e., prolific authors that, by working with different people, act as a liaison among separated groups of authors.

To find the clusters of core authors in our dataset, we consider all the authors in the dataset and we aggregate them in clusters such that each member of a cluster has at least one contribution published with one of the members of the cluster. Since we are interested in “core authors”—i.e., authors with more than 2 works in the dataset—we remove all those clusters formed around just one work—i.e., where the maximum number of publications published by the most prolific author is one.

Our analysis extracted 16 clusters from our dataset. We report in Figure 3.2 the result of our analysis, labelling each cluster from **A** to **P**. For each Cluster, we report the name of the author, the number of publications (# pub.) in our dataset and their affiliation.

The measure gives some interesting insights. First, clusters **F**, **G**, **J**, and **L** are totally localised in one country or the same University/Institute, they are relatively small (compared to the others in the Table), and include some of the most prolific authors (**J** and **L** in particular). Four other clusters follow a different trend: **C**, **H**, **P** and **I**. They are big-size clusters (respectively 6,10,8 and 6), they count one core author (respectively with 3,3,4 and 3 publications) but they are rather homogeneous, the first mainly including authors from Brazil, Finland and the fourth one is from Portugal. Clusters **A**, **B**, **D**, **K**, **M**, **N** and **O** are the most varied. Cluster **A**, is the largest

Cluster	Author	# pub.	Affiliation	Cluster	Author	# pub.	Affiliation
A	Fetzer Christof	3	TU Dresden	G	Makitalo Niko	1	University of Helsinki
A	Brito Andrey	2	Universidade de Campina Grande	H	Jin Yike	1	Unknown affiliation
A	Kopsell Stefan	2	TU Dresden	H	Yu Dongjin	1	Hangzhou Dianzi University
A	Pietzuch Peter	2	Imperial College London	H	Zhang Yuqun	1	Southern University
A	Pasin Marcelo	2	University de Neuchâtel	H	Zheng Xi	3	Xi'an Jiaotong University
A	Felber Pascal	2	University of Neuchâtel	H	Zhang Chong	2	Chong Qing Hospital
A	Fonseca Keiko	1	Universidade do Paraná	H	Liu Xiao	2	Tsinghua University
A	Rosa Marcelo	1	University of Melbourne	H	Li Rui	2	Facebook
A	Gomes Luiz	1	Arizona State University	H	Liu Huai	2	University of Washington
A	Riella Rodrigo	1	Universidade do Paraná	I	Donahoo Michael J	2	Carnegie University
A	da Silva MS Leite	1	Universidade Campina Grande	I	Cerny Tomas	6	Baylor University
A	de Oliveira SV Fernando	1	Universidade de Campina Grande	I	Sedlisky Filip	1	University In Prague
A	Kelbert Florian	1	Elastic	I	Walker Andrew	2	Carnegie University
A	Gregor Franz	1	TU Dresden	I	Svacina Jan	2	Baylor University
A	Pires Rafael	1	University of Sao Paulo	I	Bushong Vincent	2	Baylor University
A	Schiavoni Valerio	1	University of Neuchâtel	I	Bures Miroslav	2	University In Prague
A	Mazzeo Giovanni	2	MDM-IMM-CNR lab	I	Tismovsky Pavel	2	University In Prague
A	Oliver John	1	UC Berkeley	I	Frajtak Karel	2	University in Prague
A	Romano Luigi	1	Università della Campania	I	Shin Dongwan	2	Korea Institute of Energy Research
A	Brenner Stefan	1	TU Braunschweig	I	Huang Jun	2	Duke University
A	Hundt Tobias	1	UCL Institute of Child Health	J	Yarygina Tetiana	4	University of Bergen
A	Kapitza Rudiger	1	TU Braunschweig	J	Otterstad Christian	3	University of Oslo
B	Artac	1	Necmettin Erbakan University	J	Lysne Olav	1	Simula Research Laboratory
B	Casale Giuliano	2	Imperial College London	J	Hole Kjell J	1	Simula Research Laboratory
B	Van Den Heuvel W.J	2	Tilburg University	J	Ytrehus	1	University of Tromso
B	van Hoorn Andre	5	University of Stuttgart	J	Aarseth Raymond	1	University of Tromso
B	Jakovits Pelle	1	University of Tartu	J	Telnes Jorgen	1	University of Bergen
B	Leymann Frank	1	University of Stuttgart	J	Bagge Anya Helene	1	University of Bergen
B	Long Madeleine	1	University of Oslo	K	Cecconi Alessio	1	Vienna University
B	Papanikolaou Vicky	1	National School of Public Health	K	Di Ciccio Claudio	1	Sapienza University of Rome
B	Presenza Domenico	1	University of Rome	K	Dumas Marlon	1	University of Tartu
B	Russo Alessandra	1	University of Catania	K	Garcia-Banuelos Luciano	1	Tecnologico de Monterrey
B	Chesta Cristina	1	University of Chester	K	Lopez-Pintado Orlenys	1	University of Tartu
B	Di Nitto Elisabetta	1	Politecnico di Milano	K	Lu Qinghua	3	university of delaware
B	Gouvas Panagiotis	2	University of Athens	K	Mendling Jan	1	Humboldt-Universität zu Berlin
B	Stankovski Vlado	2	University of Ljubljana	K	Tran An Binh	1	CSIRO
B	Symeonidis Andreas	1	University of Thessaloniki	K	Weber Ingo	3	TU Berlin
B	Zafeiropoulos Anastasios	2	University of Athens	K	Binh Tran An	2	CSIRO
B	Soldani Jacopo	1	University of Pisa	K	O'Connor Hugo	2	CSIRO
B	Avritzer Alberto	4	eSulabSolutions	K	Rimba Paul	2	CSIRO
B	Ferme Vincenzo	3	Kiritech S.p.A.	K	Xu Xiwei	2	National Institute of Natural Hazards
B	Janes Andrea	3	The James Hutton Institute	K	Staples Mark	2	CSIRO
B	Russo Barbara	3	Free University of Bozen-Bolzano	K	Zhu Liming	3	CSIRO
B	Schulz Henning	3	Novatec Consulting GmbH	K	Jeffery Ross	2	Mayo Clinic
B	Menasche	3	University of Rio de Janeiro	L	Mirri Silvia	2	University of Bologna
B	Rufino Vilc	3	UFRJ	L	Melis Andrea	4	University of Bologna
B	Trubiani Catia	1	Gran Sasso Science Institute	L	Prandi Catia	2	University of Bologna
B	Bran Alexander	1	University of Exeter	L	Prandini Marco	4	University of Bologna
C	Rocha Carla	1	Rutgers University	L	Salomoni Paola	2	University of Bologna
C	Leite Leonardo	3	University of São Paulo	L	Callegati Franco	3	University of Bologna
C	Kon Fabio	3	University of São Paulo	L	Giallorenzo Saverio	2	University of Bologna
C	Milojicic Dejan	1	Hewlett Packard Labs	L	Delnevo Giovanni	1	University of Bologna
C	Meirelles Paulo	3	University of São Paulo	L	Monti Lorenzo	1	University of Bologna
C	Pinto Gustavo	2	University of São Paulo	M	Panicchella Annibale	4	Delft University of Technology
D	Hou Kaiyu	3	Northwestern University	M	Jan Sadeeq	1	Technology Peshawar Pakistan
D	Wu Xiaochun	3	Zhejiang University	M	Arcuri Andrea	1	Kristiania University College
D	Leng Xue	3	Zhejiang University	M	Briand Lionel	1	University of Ottawa
D	Li Xing	3	University of Chicago	M	Olsthoorn Mitchell	2	Delft University of Technology
D	Yu YinBo	1	Wuhan University	M	van Deursen Arie	2	Delft University of Technology
D	Wu Bo	3	Google Inc.	N	Zimmermann Olaf	5	HSR University of Rapperswil
D	Chen Yan	3	Lunghwa University	N	Stocker Mirko	1	HSR University of Rapperswil
D	Yu Yinbo	2	Wuhan University	N	Zdun Uwe	3	University of Vienna
E	Nikouei Seyed Yahya	3	Binghamton University	N	Lubke Daniel	1	Leibniz Universität Hannover
E	Xu Ronghua	2	Binghamton University	N	Pautasso Cesare	1	University of Lugano
E	Chen Yu	3	University of Singapore	N	Kapferer Stefan	2	Witten/Herdecke University
E	Blasch Erik	2	Air Force Research Lab	N	Wittern Erik	2	Witten/Herdecke University
E	Aved Alexander	2	US Air Force Research Lab	N	Leitner Philipp	2	University of Gothenburg
E	Nagothu Deeraj	1	Binghamton University	O	Michalas Antonis	1	Tampere University of Technology
E	Faughnan Timothy R	1	Binghamton University	O	Paladi Nicolae	1	Research Institutes of Sweden
F	Sukaridhoto Sritrusta	3	Politeknik Surabaya	O	Dang Hai-Van	3	University of Westminster
F	Panduman YY Fridelin	1	Politeknik Surabaya	O	DesLauriers James	2	CNRS
F	Tjahjono Anang	1	Politeknik Surabaya	O	Kiss Tamas	2	CNRS
F	Falah Muhammad Fajrul	2	Politeknik Surabaya	O	Ariyattu Resmi C	2	Carleton University
F	Al Rasyid MU Harun	2	Politeknik Surabaya	O	Ullah Amjad	2	Carleton University
F	Wicaksono Hendro	2	Politeknik Surabaya	O	Bowden James	2	Carleton University
G	Kilamo Terhi	1	Aalto University	O	Krefting Dagmar	2	HTW Berlin
G	Lwakatare Lucy Ellen	1	University of Helsinki	O	Pierantoni Gabriele	2	University of Westminster
G	Karvonen Teemu	1	University of Helsinki	O	Terstyanszky Gabor	2	University of Westminster
G	Heikkila	1	University of Oulu	P	Basso Tania	1	Universidade Estadual de Campinas
G	Itkonen Juha	1	Aalto University	P	Antunes Nuno	3	University of Coimbra
G	Kuvaja Pasi	1	Aalto University	P	Vieira Marco	1	University of Coimbra
G	Mikkonen Tommi	2	University of Helsinki	P	Santos Walter	1	Universidade Estadual de Montes Claros
G	Oivo Markku	1	University of Oulu	P	Meira Wagner	1	Universidade Federal de Minas Gerais
G	Lassenius Casper	1	Aalto University	P	Flora Jose	4	University of South Carolina
G	Kalske Miika	1	University of Helsinki	P	Goncalves Paulo	2	Universidade de São Paulo

Table 3.2: Cluster Authors Correspondence.

(22 authors) and most heterogeneous one: it includes 6 core authors from 5 different countries (Brazil, Germany, Italy, Switzerland, and the UK) and 12 co-authors from 4 countries different from those of the core authors (Australia, France, Portugal and the US). Cluster **B** includes 6 core authors over 24 members, distributed over just 5 countries (Brazil, Germany, Italy, Greece and Switzerland). Cluster **D** includes 8 authors, of which 6 are core and come both from either China or the US. Cluster **K** is another big cluster of 16 authors with include 3 core authors from the US and Germany. Clusters **M**, **N** and **O** follow the same trend of cluster **D**. This means that these clusters are built around 2 core authors which represent the main affiliation provenance, respectively Holland, Germany and Switzerland, US and UK.

Overall, the communities of core authors in the dataset is distributed among three types of clusters:

- “open” clusters (**A**, **B**, **D**, **K**) of co-authors linked by a few (if not one) core authors and diverse affiliations;
- “semi-open” clusters (**C**, **G**, **M**, **N** and **O**) of localised collaborators with sporadic, external collaborations;
- “closed”, localised clusters (**F**, **L**, , **P**) that tend to be small but whose core authors tend to be the most prolific (**L**).

Given their larger reach, semi-open and open clusters have a better chance to gather an impactful community around the topic. Our call to the authors in the field (particularly the closed clusters that tend to be prolific but rather localised) is to establish international collaborations and coordinate to foster the advancement and growth of the field.





Concepts and Keywords

We conclude our quantitative analysis by providing a graphical representation of the main keywords present in the abstract of the contributions in our dataset. To conduct our analysis, we used VOSviewer by [456], a software that offers text mining functionalities for constructing and visualising co-occurrence networks of important terms extracted from a given corpus. Specifically, we ignored basic words and copyright statements and performed a full count of the words present in the text. We considered only words occurring more than fifteen times, sizing them by their relevance in terms of occurrences. The resulting graph, however, is still too large and dispersive to convey useful information: for the sake of clarity, we present here a visualisation including only the top 60% most-occurring words.

We report the visualisation of the analysis in Figure 3.5.

VOSviewer automatically clustered the words in 4 areas using its modularity-based clustering algorithm, which is a variant of the cluster algorithm developed by [109] to detect communities (clusters) in a network that also considers modularity.

We can interpret the clusters as follows:

- The blue () area in the right of the figure marks the main terms of this study, grouping words like *microservice* and *system*. The result does not surprise, since those words describe the design of the systematic selection we performed.
- The green () area at the bottom marks technical terms as *container* or *attack*.
- The red () area in the left identifies application terms, e.g., the targets or reasons of the research, if it is an industrial or research-focused article. We find for instance the word *Internet-of-Things*, as it is mainly cited with industry and research applications rather than along with terms like *container* and *cloud*.
- The magenta () area at the top includes words that identify the subject of a study, whether it be some *tool*, *data* (of the system, of the users), *users*, and they *privacy*. The word *tool* here is peculiar, as it acts as a bridge between the other areas. Also, this finding is somehow expected, as the field of microservice security is marked by a fairly practical orientation towards automatisisation of processes and control.

Publication Context Analysis

In this section, we discuss trends and considerations derived from reading the selected publications and the research question detailed in Section 3.1.

Types of Publications

In Figure 3.6 we report the distribution of the type of research contribution—whether theoretical, practical, mixed or a review.

More precisely, regarding the type of research contribution, we mapped every publication in our dataset to one of the following types:

- *Theoretical* for publications that present an approach for a specific problem without any implementation artefact.
- *Applicative* for publications that describe an implemented application possibly with its validation.

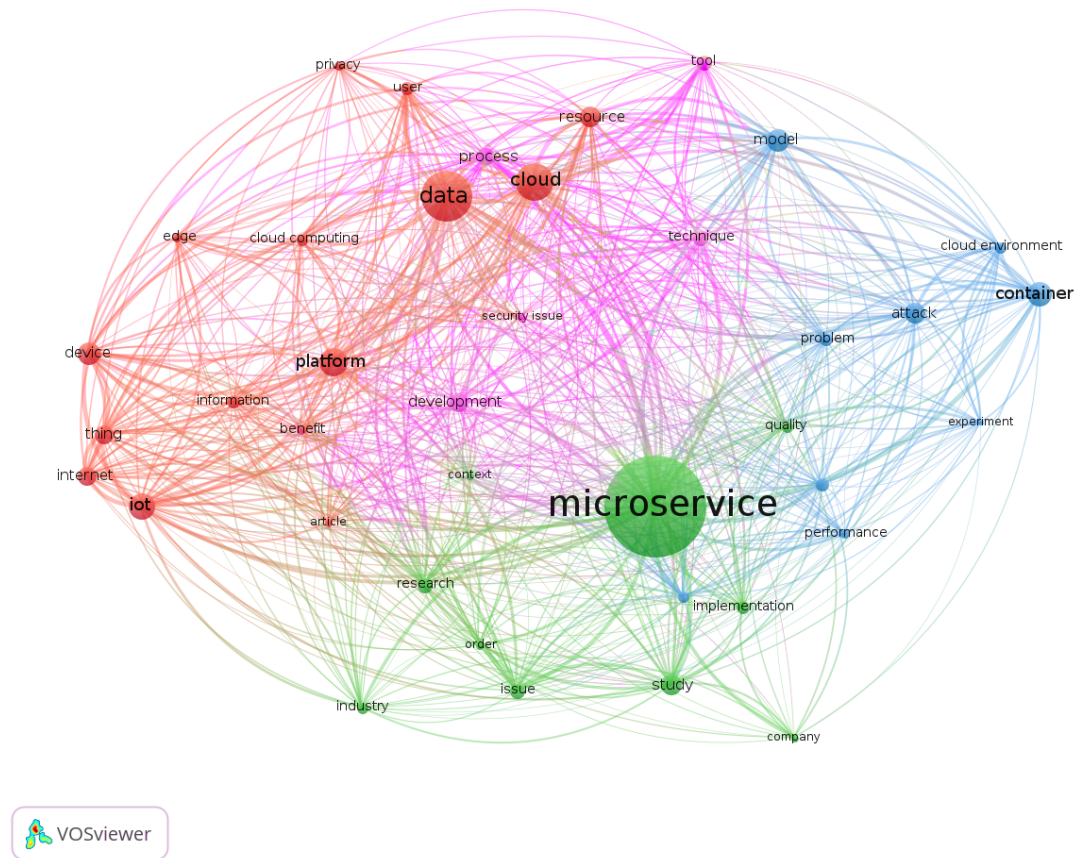


Figure 3.5: Word-Net of the abstracts in our dataset.

- *Theoretical and Applicative* for publications that develop a theory and provide a practical tool, framework, program, or application.
- *Review* for both literature reviews and social studies (e.g., on developers).

Reviews constitute 15% of the works, marking the fragmented shape of the field, which is in rapid expansion and in need of studies to map its research landscape. Besides reviews, the other contributions in the field are distributed among a 52% share that introduces new theoretical results, a 20% share that contributes by pairing new theoretical proposals with implementations, and the remaining 11% describing pure applications. The fact that the main publications in the field are theoretical is surprising, given the prominently applied nature of microservices. Indeed, excluding

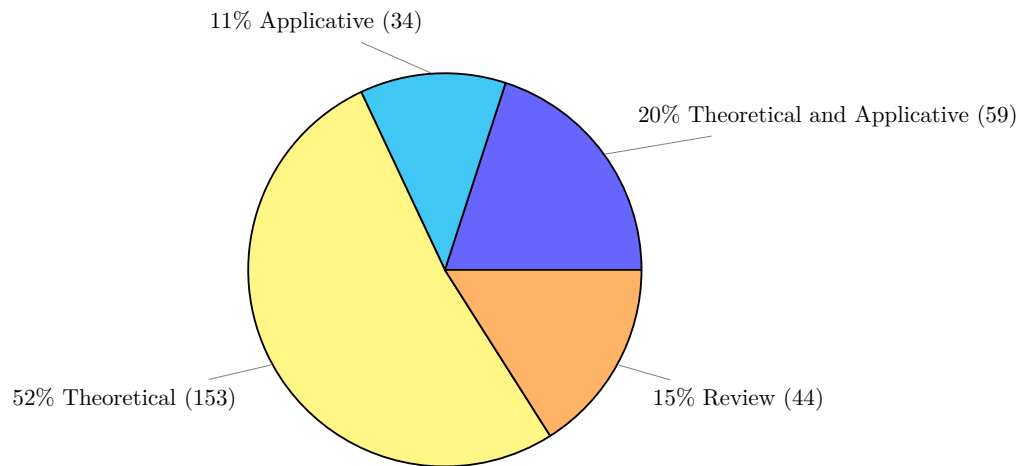


Figure 3.6: Type of publications.

reviews, we have that for every 5 publications slightly more than 3 (64% of them) are purely theoretical. We attribute this figure to two phenomena. The first marks the current exploratory trend of the field, which is still engaged in proposing new ideas and in evaluating and maturing them into models amenable to implementation. The second phenomenon relates to the impact that microservices have at the processes/organisational level, with works that are intrinsically theoretical because their contribution can be hardly crystallised into automated implementations, e.g., for proposals of attack models or techniques for handling security within organisations and development teams. Notwithstanding the possible explanations above, it is worth noting the (quantitative) distance between contributions from academia and applications available to practitioners and the industry, which is an indicator of untapped potential for joint synergies between the two communities.

After having characterised the type of publications in the field, we proceed by exploring the results from the answer of the research questions following the 4 macro-groups presented in Section 3.1.

Insights. Technology transfer: the field of microservice security is still in the early phase of new idea proposals. There are just a few implementations of these ideas, which hinders industrial adoption.

Threat Model

176 publications (ca. 65% of the dataset) give a positive answer to at least one question of this category. However, only 53 publications among those 120 (ca. 30% of the total dataset) mentioned the usage of at least one threat model to analyse or classify threats. The reason for those

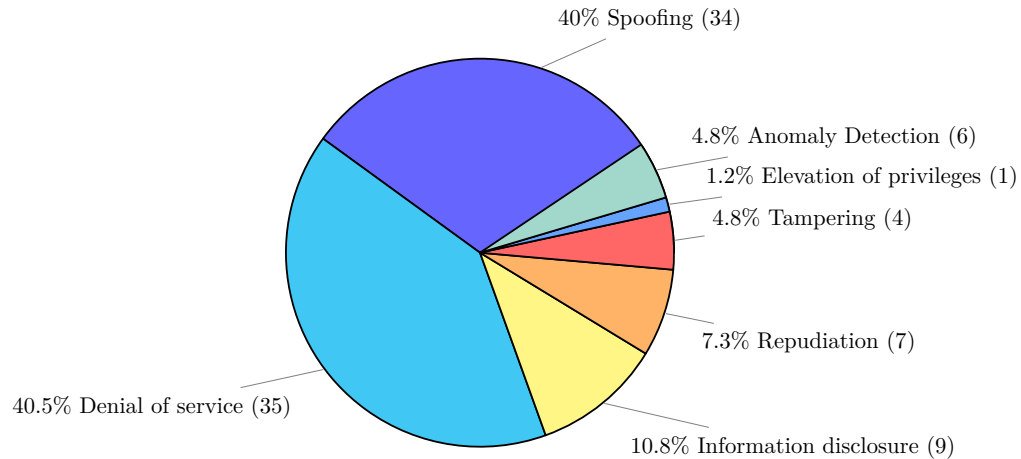


Figure 3.7: Attack type identified following the STRIDE classification.

publications to adopt a threat model vary, from publications that use the model to motivate their proposed solutions to reviews that use the model to structure their overview of the state of the art. Interestingly, in ca. 80% of those publications that mention the usage of at least one known threat model, the model is tailored to work on a specific application scenario. This is an indication of the lack of usage of a generic threat model for microservice security. We conjecture that this lack of usage of generic threat models is due to the fact that the majority of research done on microservice security comes from the software (engineering, languages) side of the field, rather than from the side of security, which advocates for a security-by-design approach.

A complementary explanation of that phenomenon is that there is no affirmed threat model for microservices, e.g., due to the difficulty of making the model specific enough for microservices yet avoiding the infamous problem of threat explosion, where the effort required to prioritise and consider all threats starts exceeding the benefits of proposing methods to manage them [478]. Threat explosion is a known problem of neighbouring areas to microservices, like cloud, edge, and fog computing [135, 208, 186, 283, 159, 449, 388] where the authors resorted to defining smaller, customised threat models rather than adopting standard ones, due to the problem of requiring conspicuous adaptation efforts to tailor them to such complex and multifaceted architectures.

Regarding the possible attacks addressed in the publications, Figure 3.7 categorises the publications based on the STRIDE threats, following up on question Q2 asking if the publication involves at least one of threats of the STRIDE classification. The most commonly tackled attacks are of the “spoofing” and “denial of service” kinds. This is an effect of the push for fine-granularity and independence of services advocated by microservices, where applications result from several small (in size), independent software components that communicate with each other. Such decentralised

communication/coordination is one of the most important attack vectors for microservice applications, in particular, the possibility to disguise a communication from an unknown source as being from a known, trusted source, which matches the spoofing attack category. Such attacks, along with tampering and repudiation ones (which together represent more than half of the attack types found in our collection), entail the need for solutions to address attacks centred around exploits of data provenance.

A similar consideration can be made for denial-of-service attacks, where the flexible scalability of microservices allows malicious intruders to, e.g., scale up peripheral microservices and hit more central and well-protected components with (distributed) overpowering attacks.

Insights. Adoption of security-by-design: security in microservice frequently comes as an afterthought, whereas it should be one of the main concerns for their engineering.

Data provenance: the quantity of spoofing, tampering, and repudiation attacks highlights the need to address the general problem of data provenance in microservices.

Dedicated attack trees and threat models: while there are attacks that specifically pertain to microservices, such as those that leverage the scalability of microservice architectures to cause denial of service, there are no dedicated threat models to help developers become aware of those particular threats.

Security Approach (Mitigation)

In terms of mitigation solutions to security issues proposed by the publications (questions Q6–Q10), the most common approach (45 publications) is to address specific problems, such as authentication or exfiltration, rather than suggesting a general approach. Publications dealing with architectural aspects rarely address the overall picture (only 25, roughly 8%, publications focus on IDS, IPS, Exfiltration Leaks and Threat Intelligence). Again, they focus on local threats like intra-communications or authentication (question Q11). These observations suggest that there is a lack of security approaches that address applications across the full stack.

As far as privacy and GDPR are involved (question Q12), surprisingly, only 9 publications consider privacy protection as relevant or worthy of analysis. In particular, only one publication [43] considers the GDPR as a guideline to follow in order to protect the privacy of users. Examples of this kind of guideline application are shown in [462]. Considering that many of the solutions included in the dataset are Cloud-based solutions, it is surprising to note that only one publication claims to be GDPR compliant.

Insights. Global view/control: the distributed nature of microservices introduces the need for technologies that provide global yet decentralised observability and control, i.e., tools that aid in the enforcement of security policies over a whole architecture without single points of failure.

React & recover techniques: while we found solution to prevent and detect attacks, there are only a few proposals about how microservice systems could react to and recover from them.

Comprehensive technological references: microservices use diverse sets of technology stacks, each characterised by peculiar exploits. To secure microservice architectures effectively, implementors need dedicated technological references to avoid known threats.

Infrastructure

We start the discussion by first focusing on the type of microservice infrastructure used by the various contributions. Specifically, we have 205 publications in our dataset that answer positively to question **Q13**. The breakdown of the answers is:

- 39% (80) describe a centralised approach;
- 24% (49) use a decentralised approach;
- 17% (35) resort to a hybrid approach;
- 20% (41) do not specify which approach they use.

The most widely adopted turns out to be the centralised one. We conjecture two explanations behind this observation. First, the centralised approach has the merit of simplifying the definition, deployment, monitoring, and evolution of policies holding over all the components in a given architecture—traded off with scalability issues and single-point-of-failure concerns. Second, we note that, among the approaches that appeared early in the literature, many focused on converting monolithic applications into microservice applications. Clearly, having a centralised controller that manages the orchestration of microservices helps this process and is closer in spirit to the monolithic workflow. However, the advent of federated, multi-cloud solutions (that prevent the identification/deployment of a centralised authority over the whole peer network) as well as new distributed-consensus technologies (e.g., blockchains), has led to a decentralisation of control, making new decentralised or hybrid solutions emerge (in our dataset) starting from 2018. As an example, in 2015 and 2016, we find publications such as [89] and [290] which presented centralised approaches to enable security in microservice platforms, while starting from 2018 hybrid and decentralised solutions appear like [353] for certificate-based authentication or [32], [31] where authors propose a decentralise high-fidelity city-scale emulation to verify the scalability of the authorisation tier.

We notice that the advent of new distributed-consensus technologies also affected the orchestration approach of microservice solutions. For example, works such as [480] propose a decentralised,

blockchain-based data-access control for microservices. Recent contributions also tackled the problem of authentication and authorisation in decentralised settings, e.g., [46] develops a workflow-oriented authorisation framework to enforce authorisation policies in a decentralised manner, [431] presents a new algorithm that distribute tasks on clusters of vehicular ad-hoc networks, [504] proposes a secure decentralised energy management framework, and [443] describes a decentralised data-centric SECurity as a Service (SECaaS) framework for elastic deployment and provisioning of security services. Another interesting work has been done in [156] where authors brought the concept of a digital twin to show how a microservice infrastructure approach can speed up the process of deploy complex infrastructure components.

Infrastructure as a Service (IaaS), which is the focus of question Q14, is also a recurrent topic in our dataset, with 66 publications yielding a positive answer. IaaS include solutions that provide and manage low-level infrastructural components, like computing resources, data storage, network components, etc. We notice that IaaS is mentioned mainly as the modality used to deploy the solution but is not studied as a security subject/mechanism per se. Works such as [424] emerge as exceptions; their authors analysed the security benefits obtained using a container-based infrastructure exposed as a service.

Question Q15 investigates Service Discovery, i.e., the automatic detection of services and their functionalities available in a given architecture/network. 16 publications mention Service Discovery in the context of security. Mainly, they propose architectures that support reactive mechanisms for the detection of security issues. Of those, only 2 mention service registration procedures that include data for performing the preventive analysis of the composition, with the goal of statically finding and fixing possible vulnerabilities and misconfigurations: [90] and [232].

Insights. Global view/control: while there is not a definitive approach to microservice security control (whether it be centralised, decentralised, or hybrid), there is a recognised need for applying security control policies in a consistent way across all microservices belonging in the same architecture.

Development

DevOps and Agile are recurring topics in our dataset. Based on the answer to question Q16, 76 publications used the DevOps approach, while, answering to Q17, 57 used Agile methods—of those 99 publications which represent the 40% of all publications in our dataset, 10 mention both approaches. There is a common consensus in these publications that Agile/DevOps is important in security because microservices seem to be the perfect match for this type of software development model [461, 206]. In particular, microservices align with the tenet of both approaches: to assign dedicated, independent teams to the development of small and independent components within the

architecture Continuous Integration (CI) process. However, the majority of the selected publications provide no in-depth security analysis of any of the two development approaches, but rather indicate the inclusion of generic security measures in the steps of the development methods. Only three works, namely [295], [33] and [254], propose concrete and specific variants of the DevOps approach that tackle security issues—in particular [295] explicitly cites the guidelines of DevSecOps [206].

Migration is one of the main challenges faced in this context; migrating applications introduces important security concerns [289] that are difficult to track, due to the lack of appropriate devices (both organisational and linguistic) to elicit them from the source codebase and make sure they hold in the migrated one. Another major challenge is the coordination between development teams in the context of privacy-handling issues [188]. Also, security becomes a challenging aspect since the (small, independent) teams need to know many aspects of security [265] and those DevOps criteria for testing, building, and deployment automation are often neither properly followed in industrial environments [70], nor for automated scans [106].

When considering domain- and model-driven approaches (questions Q18 and Q19), 16 publications consider domain-driven approaches and 26 consider model-driven ones, such as [235, 40]. These topics are therefore not as widespread as DevOps. Moreover, all citations in these cases are just brief references of the development approach, and lack a discussion on how one of the two approaches can be used in a security context on microservices.

The last question in this category, Q20, concerns security standards, i.e., curated sets of technologies, policies, concepts, safeguards, guidelines, assessments, procedures, training programmes that should be adopted to reduce security risks and mitigate attacks. The answers we gathered for this question surprised us. Indeed, security standards are a staple element of industries and organisations that want to impose and guarantee a certain level of security on their members and collaborators (often also for certification purposes— [419], [277]). Despite their widespread use in practice, only 7 publications mention security standards. In particular, [415] mentions the usage of X.509 to verify a secure method for key exchange between microservice. In [74] the authors show a solution for securing microservices through the SGX Intel Standard. The authors of [459] analyse the concept of Small-Cell-as-a-Service, i.e., a technological paradigm for the development of Virtualised Mobile Edge Computing Environments, using several mobile standards for 5G and SDN networks (e.g., MobileFlow [358] and VNFs [9]). Finally, [490] performs a deep analysis on securing microservices, citing and analysing several know standards for both microservice management and security purposes.

Insights. Migration to microservices: there are no established techniques to help developers migrate legacy systems to microservice architectures, and in particular to identify the possible security threats that come from such a migration.

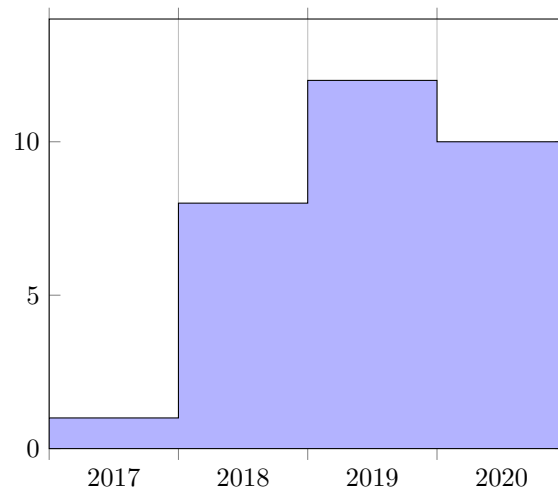


Figure 3.8: Blockchain trend.

DevSecOps: agile and DevOps practices are widely used when developing microservices, yet only a few publications address how security is addressed and combined in these practices.

Additional considerations

By analysing our dataset, we were surprised to find many citations to blockchain technologies (as reported above) as well as the lack of mainstream technologies like service mesh and serverless.

Regarding blockchain technologies, we found 31 publications mentioning or explicitly using blockchains. The decentralisation and independence of microservices constitute a good pairing for the usage of blockchain technologies. Figure 3.8 presents also the trend of publications using blockchain in the dataset. There is an increasing interest in blockchain applications for microservice architecture. Examples of that pairing include works such as [326, 480], where the trust-chain of the blockchain is combined with a decentralised microservice architecture to create strong smart contract systems or [285] where authors proposed a model-driven engineering approach for blockchain applications with microservice.

New approaches for microservices design and usage such as service mesh [272], i.e., a dedicated infrastructure layer for facilitating service-to-service communications between microservices is just mentioned by 3 works: [353], where the authors indicate a service mesh architecture for authenticating services—securely adding information to their executables and validating the correct execution of distributed entities with such certificate-based approach—and [427], which mentions the service-mesh sidecar pattern used to control security. Another interesting work regarding service mesh is [192] where authors analysed under several scenarios issues and challenges in Service Meshes

Similarly, serverless [202] is mentioned only in 4 publications. We did not expect to find (50%) more citations of serverless than those regarding service mesh. Serverless is a cloud computing execution model in which the cloud provider dynamically manages the allocation/scaling of machine resources depending on inbound requests. Indeed, while the service mesh is a technology born within the (micro)service-oriented context, serverless is a more neighbouring concept to that of stateless microservice deployment.

In this context, the most relevant publication is [92], which presents the results of a European research project to develop a model-driven DevOps framework for creating and managing applications based on serverless computing. Its main result consists in designing applications as fine-grained and independent microservices that can efficiently and optimally exploit the serverless paradigm. The serverless term, despite starting to get momentum, is still loosely related to microservices.

Given their increasing importance and impact in the industry and their close relation with microservices, we argue that both service mesh and serverless will attract the general attention of the research community in the near future, as well as that of security research.

Insights. Comprehensive technological references: the progressive adoption of new technologies in the world of microservices (such as blockchains, service meshes, and serverless) calls for dedicated investigations and reports on their impact on the security of these systems.

Correlation between Research Questions

The amount of data collected in our dataset is large enough to represent a statistically-relevant sample. In this section, we leverage this to study correlations between our research questions, by way of the answers that the publications in our dataset give to each of them. Correlations can be used to understand which of the different aspects of microservice security are most commonly in a positive correlation (paired) in the dataset, and which ones are negatively correlated (mutually exclusive).

We report in Table 3.3 the correlation matrix—excluding research question Q1, since no publication answered it. While the obtained matrix is symmetric and we could report just one half, in Table 3.3 we report the full matrix for convenience, to provide a more immediate view of how each question correlates with all of the other ones.

We conditionally colour the cells of the Table, first, attributing colour intensity according to correlation absolute value—maximal intensity for 100% and degrading towards 0%—, second, setting a transition threshold above 30% (absolute value) from green to orange, to help to spot relevant correlations. Looking at the Table, we notice the predominance of light-coloured cells. This result can be interpreted as an indication that the research questions used in this work are mostly orthogonal, and thus suited to cover the reviewed subject with almost no wasteful overlap.

	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
Q2		27.11%	32.80%	8.10%	13.75%	3.19%	-7.74%	12.36%	0.41%	24.68%	-4.12%	-22.74%	-8.06%	6.27%	-3.88%	-6.71%	8.45%	0.19%	0.41%
Q3	27.11%		28.59%	7.37%	18.93%	29.11%	7.49%	8.68%	12.81%	16.69%	8.54%	0.75%	5.51%	15.10%	-6.93%	-10.82%	3.57%	2.26%	12.81%
Q4	32.80%	28.59%		12.18%	6.30%	5.05%	10.50%	6.05%	8.45%	17.28%	9.01%	-10.39%	-7.34%	-0.42%	1.05%	1.34%	6.88%	-6.90%	8.45%
Q5	8.10%	7.37%	12.18%		6.15%	8.26%	12.31%	5.58%	13.51%	-12.44%	5.04%	-2.41%	5.31%	9.24%	-10.48%	-7.12%	-13.61%	-9.07%	8.06%
Q6	13.75%	18.93%	6.30%	6.15%		77.49%	22.89%	14.23%	14.86%	3.83%	5.58%	0.88%	17.76%	14.23%	-0.42%	2.99%	6.90%	4.96%	-5.83%
Q7	3.19%	29.11%	5.05%	8.26%	77.49%		20.77%	12.55%	17.68%	1.97%	0.88%	0.78%	15.67%	12.55%	4.41%	-1.48%	10.17%	7.44%	-5.14%
Q8	-7.74%	7.49%	10.50%	12.31%	22.89%	20.77%		10.03%	14.15%	-5.27%	20.31%	20.12%	31.15%	13.76%	8.28%	9.01%	4.83%	-4.89%	-8.03%
Q9	12.36%	8.68%	6.05%	5.58%	14.23%	12.55%	10.03%		25.72%	3.19%	13.09%	-0.84%	15.70%	14.01%	-0.66%	3.25%	8.28%	14.01%	-3.80%
Q10	0.41%	12.81%	8.45%	13.51%	14.86%	17.68%	14.15%	25.72%		8.88%	10.14%	0.37%	7.54%	6.04%	5.95%	3.53%	2.93%	6.04%	12.17%
Q11	24.68%	16.69%	17.28%	-12.44%	3.83%	1.97%	-5.27%	3.19%	8.88%		0.36%	5.13%	-17.71%	0.15%	12.62%	9.16%	9.02%	6.24%	8.88%
Q12	-4.12%	8.54%	9.01%	5.04%	5.58%	0.88%	20.31%	13.09%	10.14%	0.36%		-1.44%	9.26%	4.38%	-1.62%	6.16%	1.34%	-4.32%	-2.81%
Q13	-22.74%	0.75%	-10.39%	-2.41%	0.88%	0.78%	20.12%	-0.84%	0.37%	5.13%	-1.44%		26.24%	9.08%	11.22%	13.12%	7.16%	5.77%	5.29%
Q14	-8.06%	5.51%	-7.34%	5.31%	17.76%	15.67%	31.15%	15.70%	7.54%	-17.71%	9.26%	26.24%		22.90%	10.67%	12.47%	11.75%	8.50%	-3.18%
Q15	6.27%	15.10%	-0.42%	9.24%	14.23%	12.55%	13.76%	14.01%	6.04%	0.15%	4.38%	9.08%	22.90%		13.07%	10.85%	18.85%	14.01%	-3.80%
Q16	-3.88%	-6.93%	1.05%	-10.48%	-0.42%	4.41%	8.28%	-0.66%	5.95%	12.62%	-1.62%	11.22%	10.67%	13.07%		57.34%	25.21%	19.94%	0.85%
Q17	-6.71%	-10.82%	1.34%	-7.12%	2.99%	-1.48%	9.01%	3.25%	3.53%	9.16%	6.16%	13.12%	12.47%	10.85%	57.34%		33.08%	10.85%	-2.13%
Q18	8.45%	3.57%	6.88%	-13.61%	6.90%	10.17%	4.83%	8.28%	2.93%	9.02%	1.34%	7.16%	11.75%	18.85%	25.21%	33.08%		40.00%	-4.94%
Q19	0.19%	2.26%	-6.90%	-9.07%	4.96%	7.44%	-4.89%	14.01%	6.04%	6.24%	-4.32%	5.77%	8.50%	14.01%	19.94%	10.85%	40.00%		-3.80%
Q20	0.41%	12.81%	8.45%	8.06%	-5.83%	-5.14%	-8.03%	-3.80%	12.17%	8.88%	-2.81%	5.29%	-3.18%	-3.80%	0.85%	-2.13%	-4.94%	-3.80%	

Table 3.3: Correlation matrix among research questions.

No anti-correlation was found, i.e., negative correlations over the 30% threshold in absolute value. In the following, we comment on all positive correlations above 30%.

Q2–Q4 (32,80%) The questions relate the use of STRIDE threat model with one of is identified specific threats. This seems to be an obvious correlation since we are looking for a specific STRIDE path or at least one of his threats.

Q7–Q6 (77,49%). The questions ask if the publication mentions IPS or IDS functionalities respectively. The strong correlation indicates how IPS and IDS are strictly related. Indeed, in practice, IDS may exist without IPS, but not the opposite, because prevention mechanisms are typically built as a reaction to a detected attack;

Q8–Q14 (31,15%) The questions relate Threat Intelligence functionalities with Infrastructure as a Service deployment, which can define a campaign strategy for a Threat Intelligence analysis.

Q17–Q16 (57,34%) The questions relate the Agile development practice with DevOps and Continuous Integration. As also emphasised in other studies like [289], this correlation can be easily explained by the fact that DevOps is sometimes considered an Agile method or its evolution. Processes adopting DevOps, therefore, adopt also Agile;

Q19–Q18 (40,00%) The questions relate Domain-Driven Development and Model-Driven Development. We conjecture that this correlation is present because mentions of Domain-Driven

Development often mentions Model-Driven Development as an alternative approach and vice versa;

Q18–Q17 (33,08%) The questions relate Domain-Driven Development and Agile methods, indicating a correlation, mainly because often Agile methods employ Domain-Driven Development.

Threats to validity

Our study is subject to limitations that can be categorised into construct validity, external validity, internal validity, and reliability following the guidelines of [387].

Construct validity “reflects to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions.”. To mitigate a potential misinterpretation and making sure that the constructs discussed in the interview questions are not interpreted differently by the researchers, we adopted various triangulation rounds using online meetings and we designed a set of binary research questions to foster objectivity in answering them.

Another potential risk regards whether we were exhaustive during data collection, i.e., whether we may have missed any significant publication in our review. This risk cannot be completely mitigated but to minimise this risk we deliberately chose to have simple and broad keywords giving more initial hits that later were further filtered out. Moreover, we conducted a snowballing process to extend our initial dataset looking for potentially relevant publications that our query did not select.

External validity regards the applicability of a set of results in a more general context and is not a concern for this study since we focus on the intersection of the fields of microservices and security without any attempt of generalising the findings to a broader context. We do not claim that either our qualitative or our quantitative findings should also hold for other large fields.

Internal validity is of concern when causal relations are examined when there is a risk that the investigated factor is also affected by a third factor. This thread is not a concern for this study because we presented only correlations between different factors but did not examine causal relations.

Reliability concerns to what extent the data collection and analysis depend on the actual researchers. This risk has been partially mitigated by selecting as many objective criteria as possible for the filtering and by requiring at least a two-people consensus in case of more subjective decisions. In particular, the retrieval of the publications was performed by using search engines. The first filtering of the results (Step 2, cf. Section 3.1) was conducted by running a script that uses objective criteria such as counting the number of keywords present and the length of the publication. These

automatically computed results were double-checked by at least one person (including This thesis author) to prevent problems due to the parsing of PDFs and to make sure that the language of the publication was English. The second filtering (Step 3, cf. Section 3.1) performed by reading the title, abstract, and (if needed) the body of the publication, was performed in parallel by two persons (including the Thesis author). Decision conflicts were solved by discussion involving at least two persons (including the Thesis author) until a consensus was reached. For the publication analysis (Step 4, cf. Section 3.1), due to the binary nature and formulation of the questions, the 20 research questions were answered by the author assigned to the publication. To detect possible observer bias and errors, we selected a random subset of 15 papers and had a different author answer to the research questions. The calculation of the kappa index of agreement as proposed in [112] over the two result sets yielded a value of $\kappa = 0.99998$, giving us statistical confidence over the perceived precision of questions and objectiveness of answers.

The reliability of the study is strengthened by being open and explicit about the process of data collection and analysis. For transparency, reproducibility, and reuse, we report the data used in this study at <https://doi.org/10.5281/zenodo.4774894>, which includes both the final dataset with the answers to all the research questions and also the set of rejected publications along with the reason for exclusion.

We also report in the Appendix C each entry of our dataset and its answers to our research questions.

Discussion

In the following, we draw a summary of the main open challenges that emerged from our study, which forms a call for action for the community of researchers and practitioners working in the field of microservice security and its neighbouring areas.

Data provenance: the distributed nature of microservices calls for the certification of their outputs, which other federated services receive as input and need to trust. However, there is a lack of best practices and/or standards for such a task.

Technology transfer: there exists a sensible amount of research on microservices security, but transferring those results—e.g., viable methods and tools for validation and verification—to the industry is difficult and applications are almost non-existent.

Security-by-design adoption: while many advocate for adopting security-by-design at all stages of a microservice lifecycle (from design to monitoring), there are no established references nor guidelines on how these principles can be reliably adopted in practice.

Dedicated attack trees and threat models: threats in microservice systems can come from multiple sources, from the interaction of the layers of a chosen technology stack to how microservices interact with each other—e.g., in an exclusive network, on a federated basis, on the Web, etc. Practitioners lack dedicated attack trees and threat models to help them consider and tackle the multifaceted attack surface of microservice architectures.

Comprehensive technological references: microservice development entails the use of (heterogeneous) technology stacks, whose combinations and interactions give way to exploits at different levels. These include data leakage due to host-container interactions, threats to encryption reliability due to interacting heterogeneous standards and data-format conversions, as well as surreptitious attacks through software libraries hijacking. Besides the lack of dedicated threat models, there is also a need for concrete references to secure specific technology stacks.

Migration to microservices: several works provide structures and methods to migrate legacy systems to microservices architectures. However, there are no established techniques to elicit the assumptions and invariants (e.g., on shared-memory communication, runtime environment, concurrent/interleaved database accesses, etc.) of the legacy system that the developers of the microservices must deal with—least of all considering how those factors impact the security aspects of the migrated architecture. An additional step in this direction would benefit from following principled security-by-design disciplines.

Global view/control: the distributed nature of microservices makes it difficult to check the correct implementation of architecture-wide security policies, especially when each microservice has a dedicated security configuration. The issue is further exacerbated by the DevOps practice of having different teams deal separately with all aspects of the microservices they develop, including the implementations of their security policies. This fact highlights the need for tools that provide global overviews and guarantees on the security policies, protocols, and invariants of microservice systems.

React & recover techniques: while the literature on preventive and detective measures against attacks abound, little has been done on how microservices should react to attacks and, as a consequence, recover their normal behaviour.

DevSecOps: Agile and DevOps practices are widely used when developing microservices, yet there is no established reference on how these approaches should integrate security in all their aspects (from team culture, management and communication to develop technologies and techniques) and into the lifecycle of microservices.

Fragmentation of outlets: researchers (and practitioners) working on microservices security do not have reference venues (neither journals nor conferences). This has at least two negative consequences. First, it makes it more difficult to gather the relevant work that constitutes the current state-of-the-art of their field—a need to which this study provides a partial solution, in the form of a snapshot of the current field landscape. Second, reference venues work also as gathering and exchange points for researchers to discuss current problems and new ideas, form interest groups, and concretise new contributions and projects to advance the knowledge in the field. Here, our call for action is at the community level, advocating for the establishment of a few reference, high-quality venues able to focus, inform, and orient the agenda of the field.

4

Defense and detection

Monitoring, policies, detection and prevention are mechanisms to increase the defense of a system or a network. These mechanisms can sometime be enforced or implemented by using the previously introduced network technologies and infrastructures.

In this chapter, we will present the main contribution we implemented for this side of security and the metrics used to evaluate the security improvements in order to answer to RQ1: *Which are the emerging network technologies that can increase the protection part of the Security of a system in proactive or reactive ways?*

These contributions are implemented to cover vulnerabilty which are common in complex systems, mainly:

Insider Threats and password managements : which are addressed by a novel system to analyze password security in Section 4.2 and an infrastructure to perform penetration test and minimize the possibility of insider threats, presented in Section 4.3.

Denial of Services : network programmability such as SDN, NFV and P4 can be employed to fight DoS attacks. In Section 4.1, and 4.5 we present two systems built with different technologies and capabilities. These systems can be used – singularly or by combining them – to defeat DoS attacks.

Policies : when updating network-flows trough firewalls or hijacking paths, configuration errors can occur. While not directly being classified as attacks, this can also be reconducted to malicious behaviours (e.g. insider threats) or an involuntary DoS. To prevent this kind of problems, in Section 4.4 we present a formal verification system to check if the network configuration meets formal constraints.

4.1 SDN and NFV for Network Monitoring

Microservices is used to separate the various business logic in different functions and services. In this Chapter we move from this business logic to the lower layer: the network infrastructure, network metrics, virtual functions managements and control plane configuration. The main technologies used for this kind of tasks are Software Defined Networking (SDN) and Network Function Virtualization (NFV).

A very commonplace scenario of a network-based attack is a DoS in which the attacker overwhelms the resources of the network, by firing packets or trying connections to given services at a rate exceeding the available capacity of links and devices.

As a part of this thesis, we propose an NFV-based architecture to detect and respond to such attacks, and describe how its deployment can be fully automated. The final goal is to show how such a system can be easily deployed with high level orchestration and how it can provide automatic detection of network traffic spikes, as well as mitigate the attack with redirection of the traffic or automatic activation of firewalls, load balancers, or additional instances of the attacked service. Using these techniques, it is possible to reach a degree of automation that guarantees reaction times fast enough to withstand the attack and keep the systems available.

The framework components we used to monitor the network are: Open Source MANO, an open source MANagement and Orchestration infrastructure for NFV, fully compliant with the ETSI specifications; Prometheus, a Time Series Data Base (TSDB); and Grafana, a visualization and alerting platform. Using these tools we built the proposed NFV-based monitoring system. To simplify the complex installation process we configured an automatic installation procedure that can be exploited to automatically reproduce the system for experiments, but also to rapidly deploy production environments.

This infrastructure is designed for modularity: the various parts may be easily interchangeable, by replacing the TSDB or the visualization platform. Another benefit of this modular approach is that functionalities can be extended by chaining, if they logically support this model. For example, traffic analysis could be performed in stages, and any engine (e.g., a machine learning-based one) compatible with the network probe can be added to the analysis pipeline.

It is worth to outline here that, besides the standard interfaces and protocols, the virtualization of the network infrastructure is a core feature of the proposed design. Using this technology in conjunction with SDN, network configuration, network scaling, and provisioning can be achieved automatically, taking the burden of micro-management of the single machines or the single network devices away from the system/network administrator. Again, the final goal is to achieve reaction times that are as short as possible, thus reaching reactivity performances that prove effective to

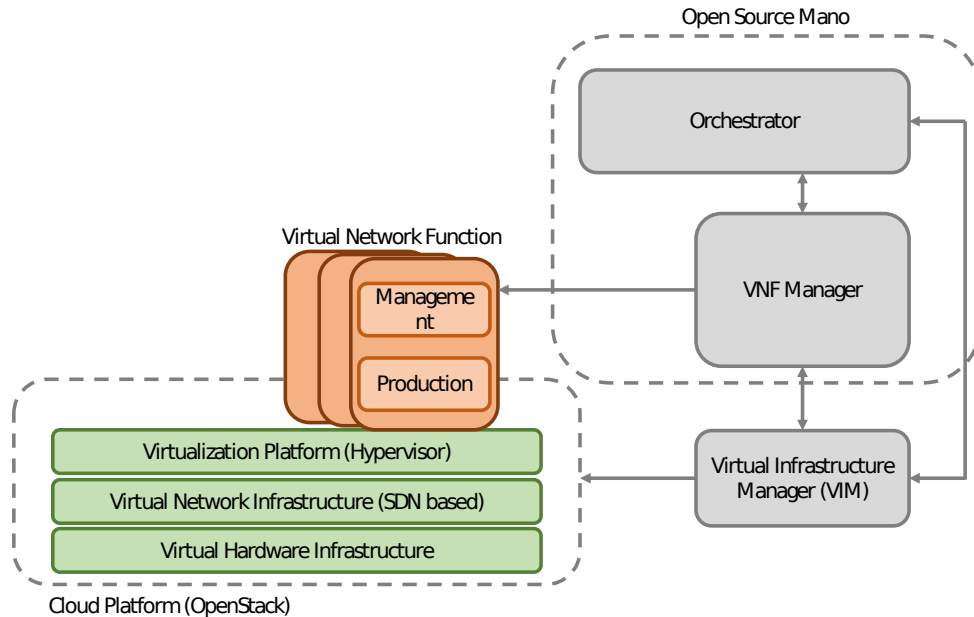


Figure 4.1: Schematic of the building blocks and interactions of the NFV architecture

contrast realistic attacks.

Fig. 4.1 shows a logical schematics of the overall architecture. The orchestrator is in charge of the various actions that are needed to create and run the virtual infrastructure. Virtual Network Function (VNF) are virtualized components that include both a management part, in charge of the communication with the Virtual Network Function Manager (VNFM) which manages their lifecycle, and a production part in charge of the actual tasks of the VNF. Finally the the Virtual Infrastructure Manager (VIM) is the logical component that will run the virtualized infrastructure, for instance OpenStack in case of a single cloud etc., an SDN controller in case of a network section etc. In this work we used OpenSourceMANO (OSM) [154] as our VNF manager. OSM is an open source implementation of the ETSI MANO specifications [153] supported by the ETSI itself and is the reference implementation for this kind of tasks. At the time of writing, it is compatible with many local and cloud-based VIMs like OpenStack, OpenVIM, VMWare, and Amazon AWS. For the management of the network, it can configure different open-source and commercial solutions such as Open Network Operating System (ONOS), OpenDayLight (ODL), and Floodlight.

Installing and running the whole infrastructure depicted in 4.1 is not trivial. The installation procedure of the various components is time consuming and also aligning their configuration for

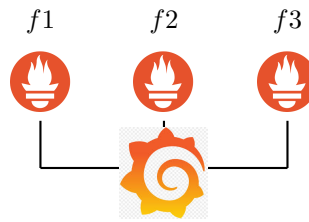


Figure 4.2: Testbed with database and network probes placement

smooth inter-operation is not simple. In the case of OSM, Canonical¹, the company behind Ubuntu, has developed a fully containerized package which can be installed in modern Ubuntu GNU/Linux distributions. This installation procedure² contains the most common free and open-source packages to setup and maintain a demonstration environment to run OSM. The software bundle contains, among others:

JUJU an application and service modelling tool. This tool is in charge of easing the installation and configuration of the software packages. It manages configurations so that they are reproducible between different builds and offers an easy interface to the customization of the various installations [99]. The modules of this tool are called *charms*, therefore this bundle is called charmed-OSM as the installation procedure uses JUJU to install and set OSM up.

microk8s a self-contained Kubernetes³ setup. In the current Ubuntu distribution version, the installation procedure is based on snap, a system based on GNU/Linux containers, which can automatically enable the distribution and dynamic update of packages on GNU/Linux distributions.

microstack a minimal OpenStack⁴ setup also installed via snap. This OpenStack setup is already configured to be managed by OSM. Its main focus is the easy installation for the developers, which therefore do not need to configure the entire OpenStack infrastructure, especially the parts which require deep knowledge of the network infrastructure, such as Nova or Neutron.

osm-k8s a complete installation of Open Source Mano running inside a container in kubernetes. This part of the system can be enabled to communicate with the installed microstack VIM using JUJU configurations.

¹<https://canonical.com>

²<https://jaas.ai/tutorials/charmed-osm-get-started>

³...

⁴An Open Source containers orchestration platform developed by Google.

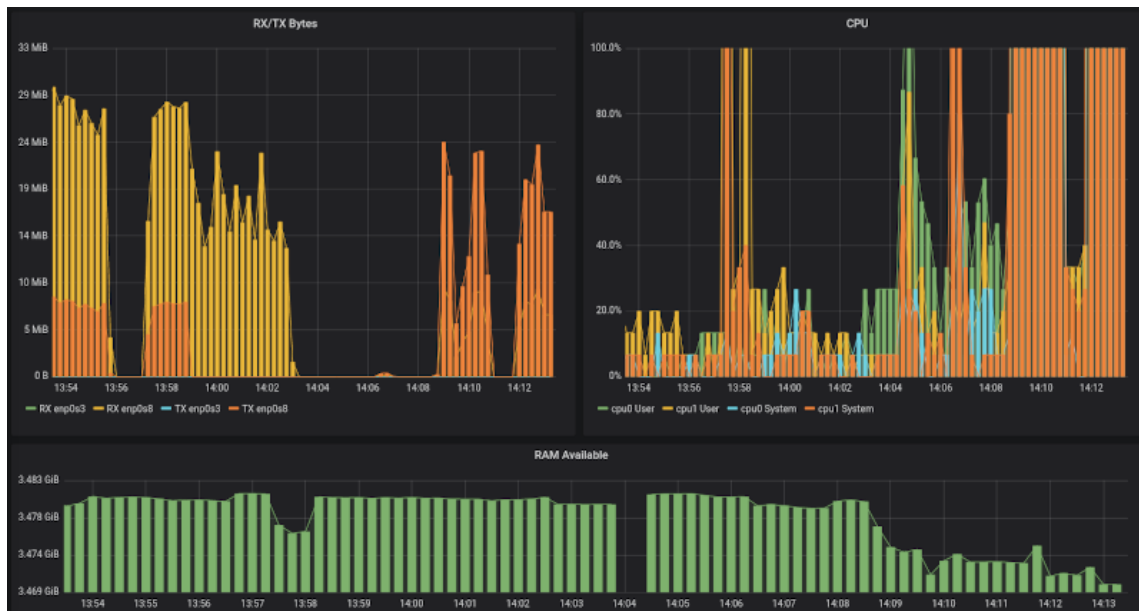


Figure 4.3: An example Grafana dashboard with network and system loads.

To detect significant network spikes and uncommon load patterns, we have considered a minimal infrastructure setup as the one described in figure 4.2. Prometheus is currently organized in two parts: the database, which is an implementation of a TSDB [323] and a collector which, in our case, is the mon collector, placed on the machines. Many collectors can be used as network probes to replace mon if needed, e.g.: telegraf, collectd, node-exporter, zabbix-agent; the analysis of the differences and the performance of this software in specific scenarios is outside of the goal of this document.

When packets are received, the present load is saved in the database along with a timestamp, enabling clients to perform queries over a time window. This is the case of Grafana, which will query the databases and visualize the graphs in a real-time fashion, allowing administrators to immediately analyze and check the detected anomalies. The Grafana administrator can also declare customized thresholds and callbacks for values, for example to send an e-mail to the operation center if the network load crosses a pre-determined threshold or if the disk of the network function is reaching its capacity limit. In figure 4.3 an example Grafana dashboard is presented. This image covers only the visualization part of this tool.

To setup the entire infrastructure, we used a set of provisioning tools. This approach, called Infrastructure as Code (IaC), also enables the reproducibility of the test. Infrastructure as Code (IaC) is a process based on the concept of maintaining the infrastructure description as a source

code, enabling the use of techniques such as versioning (e.g., with software like git) and code auditing. That is, this approach can be pictured as a series of scripts to configure the system. These configuration scripts are commonly written in languages that check whether the run of an instruction is required or the result of the command would not have any new effect on the system, making the scripts idempotent. Commonly used languages are, among others:

- ansible: <https://www.ansible.com/>
- chef: <https://www.chef.io/>
- puppet: <https://puppet.com/>
- salt-stack: <https://www.saltstack.com/>
- cloud-init: <https://cloud-init.io/>

For our system we selected ansible and cloud-init, mainly because the former does not have as cumbersome dependencies as the other ones presented above (e.g., puppet, chef, and salt-stack require the configuration of a daemon on the target machine), while the latter is automatically shipped with the VNFs created by OSM, enabling automatic provisioning of the instantiated functions.

These two tools are used for different phases: while cloud-init is ideal for first-time initialization, it is more limited and hard to use for run-time configuration. Thus, to allow run-time configuration of our infrastructure we chose to use ansible.

To test the effectiveness of our setup we have instantiated a virtual machine, managed by OSM and interconnected using a network managed by ONOS. This machine network behaviour was analyzed using an external probe, implemented as a second virtual machine and using Prometheus as traffic analyzer. A third virtual machine with Grafana was configured and connected to the Prometheus TSDB. The various virtual machines are equipped with two vCPUs and 4GB of RAM each. They run the Ubuntu 18.04 GNU/Linux operating system, as this distribution is one of the most commonly found in cloud infrastructures. The hypervisor was installed on a hexa-core Intel Core i7-8700 CPU, with a nominal clock frequency of 3.20GHz.

To analyze and simulate real-world workloads, the `Nginx`⁵ web server was installed on the target virtual machine. To analyze security problems, a denial of service attack was simulated using `hping`⁶, `ab`⁷ and `hey`⁸.

⁵<https://www.nginx.com/>

⁶<http://www.hping.org/>

⁷<https://httpd.apache.org/docs/2.4/programs/ab.html>

⁸<https://github.com/rakyll/hey>

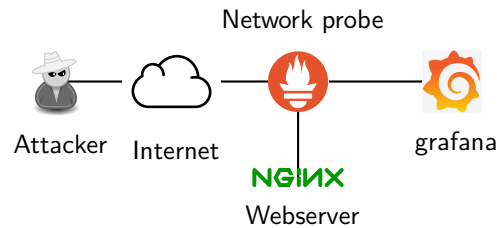


Figure 4.4: Network testbed

This attack was profiled on a realworld scenario we have observed some months before our analysis. The first tool can perform a set of TCP related attacks like SYN flooding, transport-related attacks like XMAS scan, and other kinds of network reconnaissance operations; the second and the third ones are tools to analyze the performance of a running web-server by flooding it with requests.

Figure 4.4 represents this layout, in which the attacker can reach the web server from the internet, traversing the network probe. In this case, the probe, Grafana and the web server are instantiated as network functions, to enable the application of countermeasures like network reconfiguration.

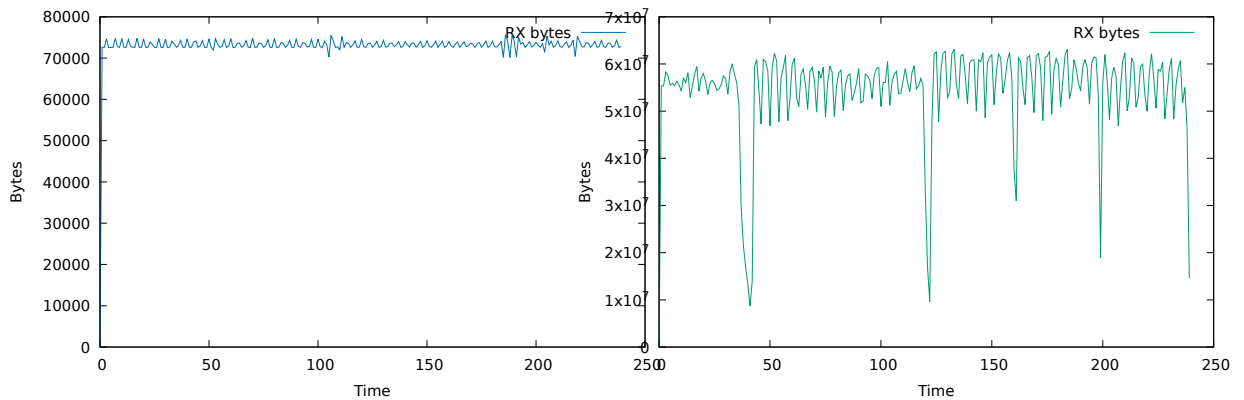
As this layout is relatively easy to instantiate with some sort of automated provisioning tools⁹, it can be extended and analyzed from a security perspective, as the configuration is never done by hand but, instead, using IaC.

When a denial of service attack is detected (e.g., by looking at network spikes), an automatic callback can be called by Grafana. In this case, a simple callback was registered, alerting the network administrator when the system was under attack.

As shown in graphs 4.5a and 4.5b, which respectively represent a simulation of standard network load and of a denial of service attack over a time span of one hour, the network load between the two cases is considerably different and easily detectable by Grafana.

Monitoring complex infrastructures is a relevant topic nowadays, and a challenge for the future. The impact that an effective, innovative monitoring approach can have on a complex infrastructure reverberates in different areas. It can improve resource utilization, positively affecting energy consumption. It can enable smarter usage of virtualization hardware, reducing hardware waste and pollution. All of this, at the same time pursuing its main purpose, i.e., preventing and mitigating attacks on the infrastructure that can cause disruptions in services that modern society gives for granted (or that are outright essential).

⁹This kind of provisioning is actuated by cloud init



(a) Network load detected by network probes in normal situation. (b) Network load detected by network probes under simulated DoS attack.

Figure 4.5: Statistics extracted from the network load scenarios. The DoS attack was simulated using tools described in Section 4.1.

4.2 Users' password enforcement

Text-based passwords are still the most common way to authenticate users against services [396]. According to a typical authorization mechanisms classification, they fall in the “what you know” category. The other categories, “what you are” and “what you have”, are most commonly used only as a second factor in the so-called Multi Factor Authentication (MFA) schemes [291]. An example of MFA is the, today very popular, one time password available on a personal device (e.g., smartphone) at each login attempt into a critical service. Ideally, passwords should be easy to remember, but hard to guess. In a long term game of cops and robbers, users try to base them on dictionary words [420] in order to make them easier to remember, and system administrators write policies to block these attempts, to limit the probability of password guessing. As a common reaction, users make simple variations to hide the base, easy-to-remember common word. This makes the password harder to remember, so users try to stick to the same one forever, but system administrators add expiration times to their policies. Users then adapt by making the smallest possible change at each password update. Unfortunately this still results in an insecure behavior, making life much easier to malicious actors trying to guess current passwords, because old ones are often available through leaked databases. The countermeasure to discourage this behaviour is to prevent choosing a password too similar to the previous one(s). The problem with the simplest approaches to similarity estimation—for example based on the Levenshtein distance [267] between two clear-text strings—is that passwords are sensitive and personal data. Storing them to enable similarity checks exposes users to an additional high risk of breaches into their accounts. This

demands for tools to implement similarity check among passwords of different age that can work without storing the password history.

We proposed a Bloom Filters based system to solve this issue, which checks passwords similarity over obfuscated data. The effectiveness of the schemes depends on the tuning of the filter which is thoroughly analyzed to determine the values of the parameters that ensure password secrecy, yet allowing an effective detection of similarity. The scheme natively allows the integration of a cryptographic access control method.

The foundations for this work can be traced back to the work of Schnell et al. [398], which describes a method for querying private data over a Bloom Filter structure, with focus on medical records. The goal is to extract data which cannot be linked to the owner, which is different from the password similarity scenario considered here. In particular an attack such as the one presented in Section 5.1, could lead to a complete compromise of the scheme. Our work propose then a solution which can address the Anagram attack. Another work presented in [19] describes the application of privacy methods like differential privacy to probabilistic data structures such as Bloom Filters. This approach is vulnerable to an attack called the “profile reconstruction attack”, a weakness that is due to differential privacy methods (which are not considered in our work) and not to the filter itself. Anonymized datasets, using techniques such as differential privacy, purposely introduces errors and noise in the data in order to hide the presence of specific information or to ensure that links between users and their correspondent data cannot be established. These controlled errors are compensated in large dataset: the errors do not effect the quality of the evaluations. In our use cases, users—even if forced to change password regularly—cannot generate a password dataset big enough to compensate the introduced noise. That is, anonymization techniques can affect the quality of password similarity queries, with several false positives compared to the proposed Bloom Filter approach. RAPPOR [150] is a system used by Google to get data from the Chrome browser. The data is hashed in a Bloom Filter, anonymized by introducing a perturbation on the values, and then retrieved and reconstructed at server side.

RAPPOR is used to collect binary statistic, therefore cannot be used to detect similarities between responses. It is based on binary queries and not on difference between sets. These exact matches (without false negatives and with a probability of false positives), are considered in other works. That is due to the construction and the use of the Bloom Filters. One of the closest work to this one, referenced as SSDD, is presented in [163]. It employs Homomorphic Encryption to compute the Jaccard coefficient between two Bloom Filters. One of the downsides of Homomorphic Encryption is the exponential-order complexity of the algorithms, which makes it unfeasible for many systems such as embedded ones. As the authors say, it can leverage on pre-computed values, that can be saved in a sort of cache that can speed up the encryptions and decryptions. Unfortunately if the output of the pre-computation is not saved in a secure way, this can leak information

to an attacker. In SSDD as well as in the method proposed in this manuscript, the filters are used to return a real number which represent the similarity, not a boolean value, placing at the opposite side of the spectrum, with respect to RAPPOR.

Moreover in general the aforementioned work do not explicitly consider the use of secure hash functions (such as cryptographic secure hashes). The added security of these methods is therefore not in the scope of the cited papers. Other related works, which are not directly based on Bloom Filters, are the ones which employs fuzzy hash functions. The main work in this area is called Context Triggered Piecewise Hash (CTPH) [9]. CTPH fuzzy hash function is extensively used in its reference implementation, *ssdeep*¹⁰, by platforms such as VirusTotal¹¹. This implementation is particularly useful to check the similarity of uploaded samples with known malware files and to perform forensic analysis [273]. To the best of our knowledge, the security of this hash function was not extensively compared to “classic” families of hash functions such as SHA512 and MD5. To provide a more straightforward comparison of the characteristics of existing methods and our proposal, in Table 4.1 we lists the main peculiarities of the various proposed approaches, and compare the different goals.

Table 4.1: Comparison between similar approaches. The ● symbol defines a full compliance to the row, the ◐ symbol denotes a partial compliance, and the ○ symbol denotes an absence of compliance to the row. The acronyms legend is the following one. BF: Bloom Filter, HE: Homomorphic Encryption, PHFs: Piecewise Hash Function.

Peculiarity	RAPPOR	SSDD	Schnell et al. [398]	CTPH	Our Method
	Úl. Erlingsson et al. [150]	S. Forman et al. [163]		J. Kornblum [248]	
Detect exact matches	●	○	○	●	○
Detect similarities	○	●	●	●	●
Can be (natively) encrypted locally	◐	◐	◐	○	●
Uses or can use secure hash function	◐	◐	●	◐	●
Main focus	Crowdsourcing	Documents	Medical Records	Malware analysis	Passwords
Main technology	BF (binary)	BF + HE	BF	PHF	BF

While these works are mainly related to privacy scenarios, in the password strength evaluation implementation we often see clear-text mechanisms to check the similarity of the last passwords. Another approach is to require the last password, but, if lost, it can result in a total lockout for the user or a subvert of the password change mechanism. Current guidelines of the National Institute of Standards and Technology (NIST) [180] do not provide any rule against password reuse, only for temporary secrets.

To the best of our knowledge, in the literature there is not a use case of application of Bloom Filters to a password management scenario. Nowadays, companies and research groups try to deal

¹⁰<https://ssdeep-project.github.io/ssdeep/index.html>

¹¹<https://virustotal.com>

with such attacks by mixing different strategies. Companies such as Facebook claimed to have bought black-market passwords in order to analyze the similarity among passwords in order to defeat *The No. 1 cause of harm on the internet* [2].

Other research groups otherwise proposed the idea of changing the structure of the password file in such a way that each user would have multiple possible passwords, sweetwords and only one of them is real. The false passwords are called honeywords. As soon as one of the honeywords is submitted in the login process, the adversary will be detected.

Despite these attempts, we claim that research has not yet provided an extensive analysis of this field. For this reason the aim of this part of thesis is to study it, illustrate the advantages it brings, and discuss the security-related issues that it introduces.

Password similarity

It is widely known that password reuse is a common behaviour which can turn into a threat if passwords get leaked online [216]. Password leaks are a common form of information leak that happens regularly¹². A similar threat appears when users are allowed to choose a new password with little variations from the previous one (e.g., `password2020` changing password from `password2019`). This behavior is tempting especially in corporate settings which enforce a policy of frequent password expiration. This can make brute force attacks very effective, since the new password is easily computed by a limited number of mutations starting from a dictionary of leaked ones. We can describe the password mutation as a perturbation of the password with slight variants. Tools such as password crackers or word-lists generators like `cupp` (A password generator based on personal and open source data:¹³ or `johntheripper` (`johntheripper` is a common password cracker and generator¹⁴ implement various combination methods for password generation; passwords can also be generated by neural-based techniques such as adversarial generation [280]. Against these approaches, choosing a similar password is almost as insecure as choosing a dictionary password [475]. It is worth noticing that common methods to guide users to choose “robust” passwords are focused on avoiding the direct use of dictionary words. These methods have two shortcomings: users resort to variations that are easily discovered by mutation by the aforementioned tools, and there is no detection of password similarity when a password change is mandated.

For the purpose of this work, password similarity can be informally defined as the structural similarity of the text composing two password being compared, i.e., it has nothing to do with the possible underlying meaning of the string. This definition of password similarity can be used to guarantee that a user is not recycling what, in terms of actual entropy, can be considered

¹²<https://us.norton.com/internetsecurity-emerging-threats-2019-data-breaches.html>

¹³<https://github.com/Mebus/cupp>

¹⁴<https://www.openwall.com/john/>

the same password every time. A straightforward method to detect password similarity over a meaningful time span would require saving old passwords in clear-text into a database, which is obviously a despicable approach. We propose a solution based on Bloom Filters that overcomes this shortcoming.

Bloom filter for Text Similarity

A Bloom Filter, denoted in this work as β , is a probabilistic data structure which can be used to cache data and speed up operations such as lookup in databases [68, 311, 184]. It is composed by:

- a bucket which can be an array of bits initially set to the false value (0), we reference to its size in the number of bits as κ ;
- Γ , a set of hash functions which will be used to insert and check values.

The relevant operations on a filter to measure similarity are:

- $Create(\Gamma, \kappa) \rightarrow \beta$ which generates a Bloom Filter β using the hash functions present in the set Γ with a bucket of size κ .
- $Insert(\beta, s)$ which inserts the bit string s in the Bloom Filter.
- $Check(\beta, s) \rightarrow Boolean$ which checks if the value s is not present in the filter or if it collides with a value which is already there.
- $QInsert(\beta, s, \nu)$ that inserts the string s splitting it in ν -grams.
- $Distance(\beta_1, \beta_2) \rightarrow Real$ that returns the distance between two Bloom Filters. To be comparable, two Bloom Filters must have the same bucket size κ , and need to use the same set of hash functions Γ .

An insertion operation ($Insert(\beta, s)$) of a string s in a Bloom Filter β is performed according to the following steps:

- the value that must be inserted into the bucket is hashed using the set of hash functions; The hash functions output must be re-mapped to provide indexes in the co-domain of cardinality κ .
- every bucket slot indexed by the keys got using the hash functions is set to the true value (1).

This operation therefore inserts the hashed value into the filter, setting the corresponding hash values to the true value. The process is described in Figure 4.6 which pictures an insertion of the strings *password1234* and *password123!*.

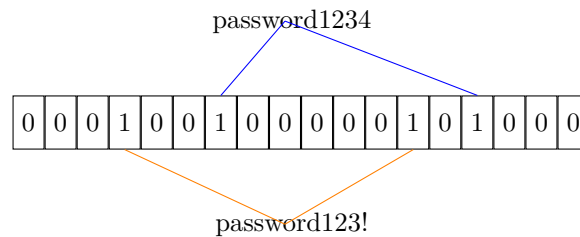


Figure 4.6: Insertion procedure with two strings. The strings *password1234* and *password123!* are hashed independently. This insertion procedure processes the passwords as single items, leading to different hashed values.

The verification process of the string s presence in the filter β ($Check(\beta, s) \rightarrow Boolean$) is analogous to the insertion case:

- The element s is hashed against all the functions to get a list of indexes;
- If any index points to a false value, then the element is not present in the filter for sure. The Bloom Filter never exhibits false negatives.
- Otherwise the value could be present in the filter, but due to the collision possibility of the hash functions, the result can be a false positive.

This procedure is illustrated in Figure 4.7.

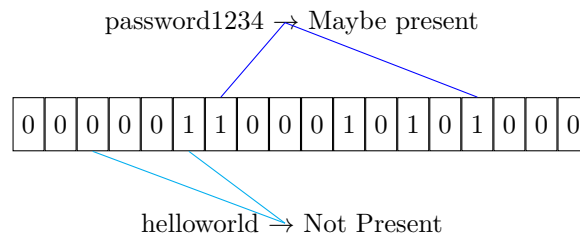


Figure 4.7: Check procedure with two strings. The strings *password1234* and *helloworld* are hashed independently and the resultant indexes from the hash functions are checked in the bucket. If the lookup lead to a 0 value, the string is not present in the filter. Otherwise, it can be a value that is present in the filter or can be a collision (a false positive).

Bloom filters can be used for text similarity using distance-sensitive hash functions [39] such the ones introduced by Kirsch et al. in [245], or using an n-gram approach. This latter technique divides the string in n-grams and hashes every resultant n-gram with the hash functions present in the set, constructing a solution similar to locality-sensitive hashing [209, 178]:

The hashing procedure to enable the measure of distance is presented in Figure 4.8. The similarity of the two sets can be calculated by using various distance definitions, by extending the check procedure to yield the number of n-grams which are the same in the two strings, independently of the order. The similarity distance of two Bloom Filters can be expressed using distance and similarities coefficients [80]. The main ones are The Jaccard coefficient [217] (which we will denote with J), the Dice coefficient [140] (which we will denote with δ) and the Cosine similarity [48] (denoted by ϕ). These coefficients can be calculated as functions between two Bloom Filters and are defined as:

$$J(\beta_1, \beta_2) = \frac{\gamma_{\beta_1, \beta_2}}{k_{\beta_1} + k_{\beta_2} - \gamma_{\beta_1, \beta_2}} \quad \delta(\beta_1, \beta_2) = \frac{2\gamma_{\beta_1, \beta_2}}{k_{\beta_1} + k_{\beta_2}} \quad \phi(\beta_1, \beta_2) = \frac{\gamma_{\beta_1, \beta_2}}{\sqrt{k_{\beta_1}} * \sqrt{k_{\beta_2}}}$$

where $\gamma_{\beta_1, \beta_2}$ is the common number of true values in the sets of the two Bloom filters β_1 and β_2 , and k_{β_1} and k_{β_2} are the number of true values of, respectively, the β_1 filter and β_2 filter.

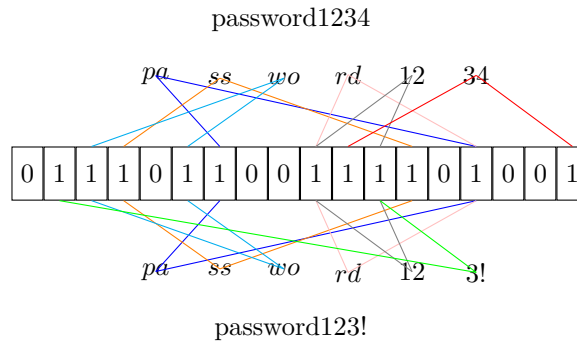


Figure 4.8: n-gram insertion procedure with two strings. The strings are divided into n-grams (in this case bi-grams) and hashed using an *Insert* operation for every n-gram.

The Jaccard coefficient between two filters has already been exploited, and found to be an effective method, for computing Bloom Filter similarity. This coefficient, introduced in [217] by Paul Jaccard, is shown to be suited to calculate the difference between two bit-sets, such as the one at the basis of the structure of the filter. Therefore, this coefficient is a common root of the systems based on the similarity between sets and Bloom Filters [341, 460, 335]. The main related previous works for its applications are [163, 383]: these exploit the Jaccard coefficient for private documents, but are not tied to the password similarity field. The other functions were extensively analyzed in the works which we reference [398, 311, 224]. The choice of the best distance function is related to the application case. We propose a comparison between their performance in the field of password similarity using Bloom filters in Section 4.2.

As stated before, the structure of a Bloom Filter makes possible to have no false negatives (values that are returned as not inserted in the filter while they actually were there) and a number of false positives (values that are returned as inserted in the filter but no insertion procedure was applied on them). The percentage of false positives can be limited using the formulae presented in this section which are based on the choice of the hash function set Γ and the size of the bucket κ . The tuning of these parameters is essential to achieve a satisfactory trade-off between the utility of the query and the number of false positives. A wrong sizing of buckets or a choice of low-randomness hash functions can easily lead to a vulnerable filter (as detailed in Section 5.1) or to an unstable filter that exhibits too many false positives. This latter case is formally defined by the formulae presented in Section 4.2 and experimentally analyzed in Section 4.2.

Privacy Guarantees

To size the filter, some criteria can be derived from the following formula, as stated in the “classical” work by Burton H. Bloom [68],

$$fpp = \left(1 - \left(1 - \frac{1}{m} \right)^{kn} \right)^k \quad (4.1)$$

where fpp is the false positive probability of the filter. This can be calculated a-priori from the variables m , k and n , defined as:

- m , the cardinality of the set on which the filter is built;
- k , the number of different hash functions that are used to hash values into the filter;
- n , the number of elements which will be inserted into the filter.

The formula is derived from the evaluation of the probability of an item to be represented by only 0 in the Bloom Filter set. In cases of uniform distribution, as it happens with hash functions used by a Bloom Filter, this probability is defined as $1 - \frac{1}{m}$. This value needs to be exponentiated to the number of hash functions k multiplied by the number of elements. That is a consequence of the construction of the *Insert* operation: an operation of this kind is required for every inserted value. A single *Insert* operation will hash a single input string for every hash function in the hash-set Γ . The resultant false positive probability is therefore the probability of having only value set to 1 for every result of our hash function set.

From this formula we can derive the optimal values for the filter size and the number of hash functions to use. For a specific number of elements, that will be inserted in the filter, we can choose a fixed fpp to get a desired percentage of false positives. To get the formulae to calculate

these optimal values, we first need to approximate the value of fpp to $fpp \approx (1 - e^{-kn/m})^k$. As explained in [77] the optimal number of hash functions can be found from the derivative of the fpp. Let us declare $g = k \ln(1 - e^{-\frac{kn}{m}})$, minimizing this function g will lead us to the optimal (minimum) number of hash function. Therefore we can find that the global minimum is defined as: $k' = \frac{m}{n} \ln(2)$. From this formula we can derive the optimal value for the size of the array. Replacing k' to k in the Equation (4.1) we find

$$fpp \approx (1 - e^{-\frac{kn}{m}})^k = (1/2)^{\frac{m}{n} \ln(2)}$$

Which can be resolved for m , obtaining its optimal value, which is:

$$m_{opt} = -\frac{n \ln(fpp)}{(\ln(2))^2}$$

Obviously, these values must be integers, therefore we can apply a ceil to the result of the formulae, obtaining:

$$m' = \lceil m_{opt} \rceil = \left\lceil -\frac{n \ln fpp}{(\ln(2))^2} \right\rceil \quad k' = \left\lceil \frac{m'}{n} \ln(2) \right\rceil$$

The rationale behind these sizing formulae comes from the observation that as the size of buckets increases, the probability of collisions decreases. Using this approach, a Bloom Filter with a controllable number of false positives can be tuned to fit any specific scenario. Conversely, as it is useful for the proposed application in order to enhance password confidentiality, a Bloom Filter can be designed in such a way to have a big number of false positives, thus obfuscating data by forcing collisions. Password confidentiality could be protected also by adopting privacy-preserving approaches, such as δ -presence [330] or differential privacy [147]. These approaches take in consideration the amount of data stored into a database, or the filter in this case, and try to anonymize the data among many false positives. In this case the approach is directly applicable to the filter which can gain a lot of advantages in terms of privacy from this approach. Specific metrics for these scenarios were studied by Bianchi et al in [65], which proposed γ -deniability and after Xue et alii in [482].

Analysis of the Hash Function Family

Another key component of the filter, which impacts on confidentiality, is the distribution of the values returned by the chosen hash function family. In this kind of probabilistic data structures, it may be necessary to deploy a huge number of hash functions. Using a different algorithm for every index can be unfeasible, and in any case a large variability within the hash function set hinders a precise analysis of the randomness of the generator. To overcome these problems, the hash function

set can be generated by using the well-known salt approach, as described in Figure 4.9. When a value has to be inserted, many random numbers (salts) are generated. Prepending these numbers to the value, and using a fixed hash function, the effect is analogous to having adopted different, random hash functions. In our implementation we used MD5 as the base hash function. The salts can be generated using different hash functions, such as Murmur hash [37] or secure cryptographic hash functions, such as SHA512. These are easily interchangeable, as libraries such as OpenSSL¹⁵ implement the various methods using interfaces that facilitate high-level use of an hash function. In Section 4.2 we propose a compared analysis of the speed of two hash cryptographic functions: MD5 and SHA512. The security of the hash function was studied in numerous works [185, 177, 244], in this case, the security of the system also relies on the division in n-grams. The salts are saved in the same location as the filter set, to allow reloading the state in subsequent invocations of the algorithm. We acknowledge that this trivial solution is vulnerable to attacks if the file is saved in clear-text form, like the one described in Section 5.1.

Saving salts in a secure way seems not too challenging, and is the subject of current and future investigation aimed at making the filter immune to this kind of attacks. For example, a salt can be generated using a cipher like AES using an user provided key and a fixed payload similarly to AES-CTR mode [28]. AES, in this mode, generates an encryption stream starting from a counter c . The encryption stream is derived by encrypting this counter, initialized to a random IV. After this generation step, the encryption stream can be combined with the payload, using a *xor* operation. Similarly, we can extend the hash functions set of the filter with a function $GenerateHashes(key) \rightarrow \Gamma$ which generates the set of hash functions Γ starting from the key key . The salt of the functions is generated and encrypted with a key using a symmetric cypher such as AES. The security in this case relies on the key, which must be chosen by the user so as to withstand known attacks such as brute force and dictionary attacks. Using this technique, the filter set can be saved without specific protections, since it can be verified only using the chosen secret key. An in-depth analysis of the security of the encryption scheme, particularly concerning the peculiarity of having very short payloads due to the division in n-grams, will be the focus of ongoing research work. These approaches to the generation of hash functions are described in Figure 4.9. In the first hash set we can see that $h1$ and $h2$ are two functions generated with a random padding applied to the MD5 hash function. In this scenario we suppose to have a $Random(n)$ function that can generate a random string of length n . This generation, when re-applied will lead to a totally different set of hash functions, making the distance function inapplicable. That is the concept of the construction of $h1'$ and $h2'$. These functions can return different results from the functions $h1$ and $h2$ described before. Reusing the same value for the salt applied to the functions will lead to

¹⁵<https://openssl.org>

the same set of results, making the distance calculation possible. This is the concept of the third figure (bottom-left), which describes how, applying a fixed salt to a symmetric cipher using a secret key k the filter will lead to the same set of results.

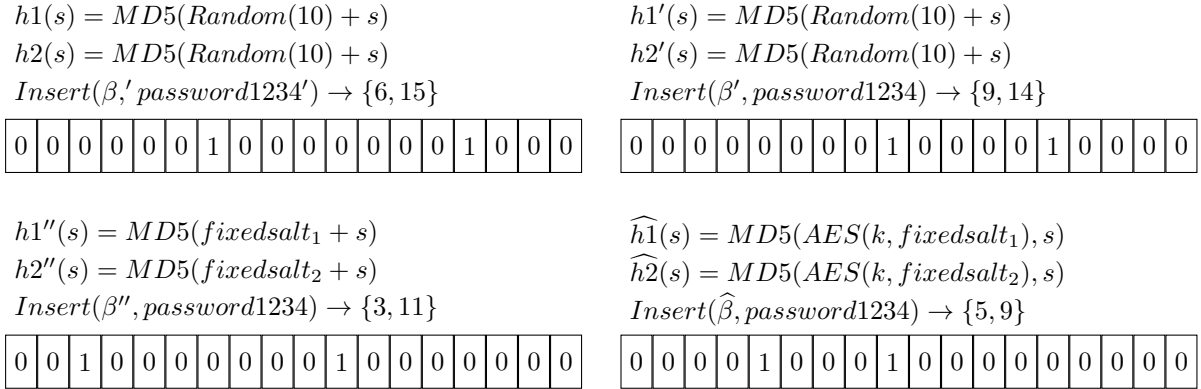


Figure 4.9: Different hash functions generation. These functions will lead to different use-cases. In the first two figures (the top-right and top-left ones) we have a fully random generation which leads to different cases every time as we cannot predict the output got from *Random* function. The fourth case employs a cryptographic function and a set of fixed strings ($fixedsalt_n$) to generate the same set of hash function based on a secret key k .

Experimental Analysis

The specified method to check password similarity was been implemented in C language. The hash functions used was the standard OpenSSL¹⁶ implementations of hash functions, in this case MD5. The system was checked on a Ubuntu 18.04 system running in a VirtualBox virtual machine with 2 virtual CPUs and 1GB of memory. The hypervisor ran over an Intel core i7-8700 CPU which clock frequency runs at 3.2 GHz, and the host was used exclusively to run the test VM. The random data were provided by `/dev/urandom` to avoid blocking behaviour [191] and read to generate random hash functions. We used the following queries used to check the filter:

1. $\beta \leftarrow Create(\Gamma, \kappa)$
2. $Insert(\beta, AAAA)$
3. $Insert(\beta, BBBB)$
4. $Check(\beta, AAAA)$
5. $Check(\beta, CCCC)$

¹⁶<https://www.openssl.org/>

6. $Check(\beta, BBBB)$

In this case the hash dimension seemed not to influence the performances of the filter. The computational load was dominated by the generation of the salt string. The performance of the salt string generation presented in Figure 4.10 is the average of the results of five runs of experiments, in which the size of the salt changed from 1 (really easy to brute-force) to 1000 (really hard to brute-force). As shown in the graph, the performance decreased linearly when the salt size increased. This happened because a single random character of the salt must be multiplied by the number of hash functions present in the filter. The $QInsert$ and $Distance$ performances were evaluated using the following querying pattern:

1. $\beta_1 \leftarrow Create(\Gamma, \kappa)$
2. $QInsert(\beta_1, thisismy\text{password}, 2)$
3. $\beta_2 \leftarrow Create(\Gamma, \kappa)$
4. $QInsert(\beta_2, thisismy\text{password}, 2)$
5. $\beta_3 \leftarrow Create(\Gamma, \kappa)$
6. $QInsert(\beta_3, thisismy\text{password}, 2)$
7. $Distance(\beta_1, \beta_2)$
8. $Distance(\beta_1, \beta_3)$

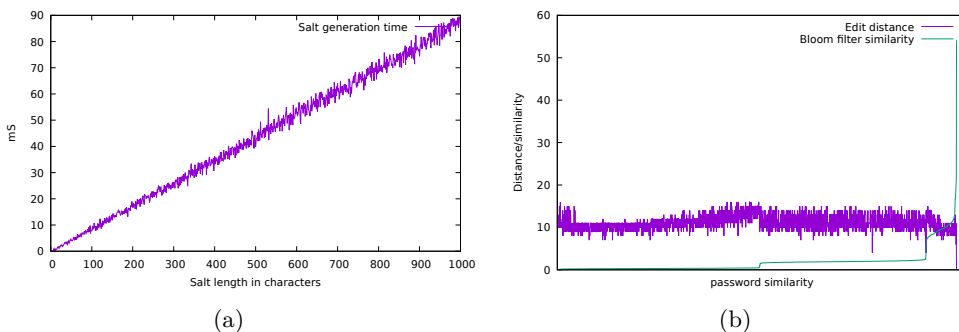


Figure 4.10: Speed of initialization of the filter and performances compared to edit distance. (a) Creation time of the filter changing the size of the salt strings. (b) Performances of the filter compared to the edit distance applied to the password data set.

In this test run, as in the $Insert$ evaluation, the computational load was dominated by the filter generation.

This system was also tested using ca. 5000 credentials from real-world leaked institutional logins. Data were clustered and analyzed using the filter, to observe which values were correctly identified as similar. The graph in Figure 4.10 represents the analysis of the filter's precision. The graph also shows the edit distance between the various strings.

The plots in Figure 4.11a–d are the experimental confirmations of the sizing criteria presented in Section 4.2. These graphs were generated using different credentials from the ones in Figure 4.10a,b, to confirm the general applicability of the filter. *The data set was generated taking the common password database rockyou.txt (The one available in Kali Linux Distribution,¹⁷, under /usr/share/wordlists/rockyou.txt.gz), inserting in the filter 5000 random-chosen credentials, and comparing them with 5000 other credentials randomly generated using a password generator tool such as APG [170].* As clearly highlighted from the graphs, there was not a big performance gap between the two algorithms used, the secure cryptographic hash function SHA512 and the insecure MD5. Furthermore, the behaviour of the system respects the previously described formulae: a bigger filter size reduced the probability of a false positive. The hash functions number had a different impact on the scheme: it could degrade the performances if we chose a non-optimal number of functions. Choosing a low number of functions resulted in collisions by the same function, choosing too many functions resulted in filling the buckets easily, accelerating the occurrence of false positives.

¹⁷<https://www.kali.org/>

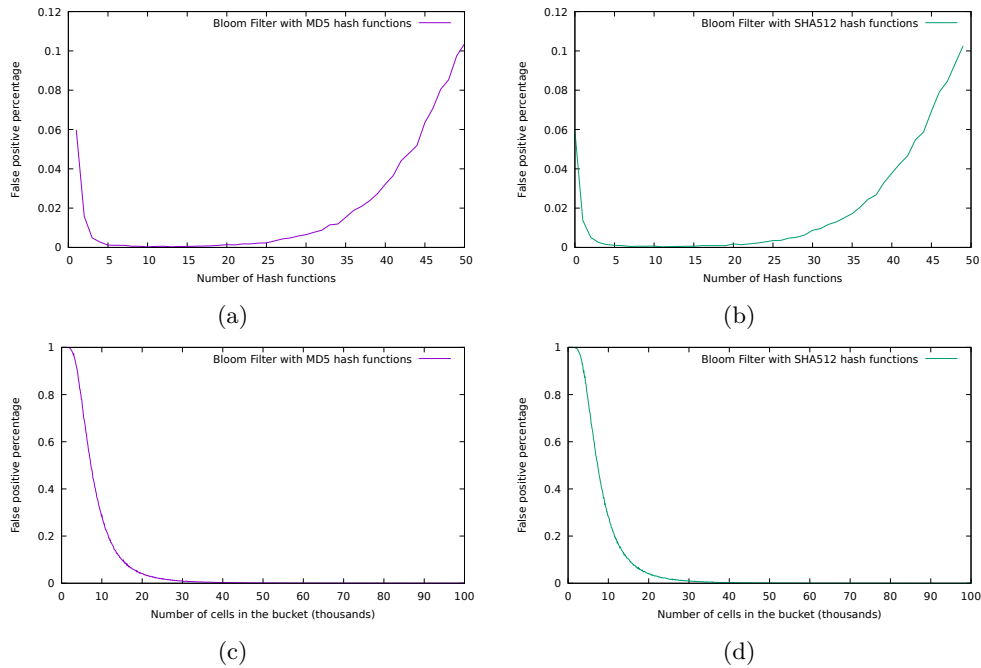


Figure 4.11: *Performance* of the filter in terms of False Positive Percentage with different sizing of Γ (number of hash functions employed) and κ (size of the internal bucket). **(a)** Performance of a Bloom Filter based on MD5 hash functions in respect to number of hash functions used. **(b)** Performance of a Bloom Filter based on SHA512 hash functions in respect to number of hash functions used. **(c)** Performance of a Bloom Filter based on MD5 hash functions in respect to size of the internal bucket. **(d)** Performance of a Bloom Filter based on SHA512 hash functions in respect to size of internal bucket.

Figure 4.12a represents the performances of the system, in terms of time required to execute a *QInsert* of a random string divided in bi-grams or a *Distance* operation. The graph shows that the system had a linear response to the length of the input. We analyzed lengths from 1 to 64 chars, which included the most common lengths of password strings (7, 20) [161]. Furthermore, the plot shows that the SHA512 hashing method was clearly slower than MD5. This speed gap was due to the intrinsic complexity of the algorithm. That introduced a trade off between the security of the scheme in terms of resistance to collisions and the speed of the overall procedure.

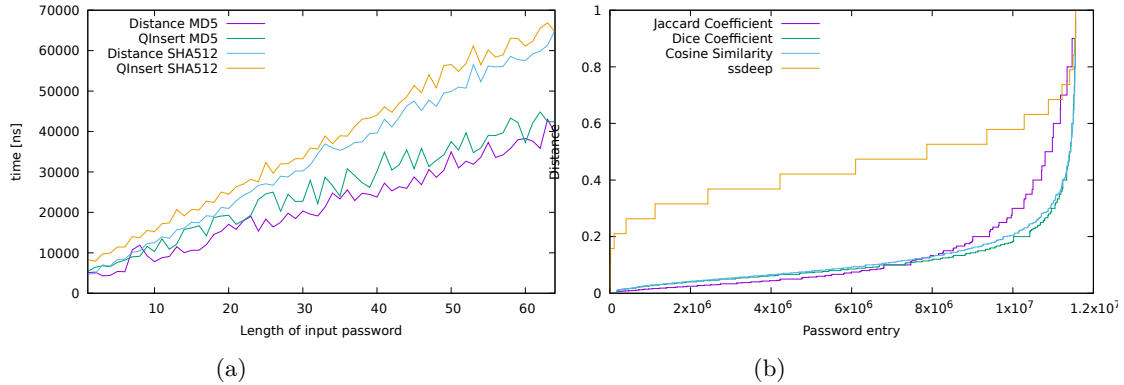


Figure 4.12: *Comparison* of the two hash functions implemented (MD5 and SHA512) and performances of the different distance metrics (Jaccard coefficient, Dice coefficient and Cosine similarity) compared to a reference implementation of CTPH [248] (*ssdeep*). (a) Performance of the system by inserting and checking random password of different lengths. (b) Comparison of different functions to calculate distance. Namely: Jaccard coefficient, Dice coefficient and Cosine similarity.

The last graphical representations in Figure 4.12b describe the difference between the three implemented distance measures presented in Section 4.2. The behaviour of the three functions was comparable across the spectrum of passwords lengths employed to generate the plots in Figure 4.12a. From the plot it is clear that Dice’s method and the Cosine-based one gave more accurate results than the the Jaccard-based solution. The plot also shows the performance difference of our C implementation with a reference implementation of the CTPH scheme: *ssdeep*. Comparisons with the system presented in Table 4.1 would be infeasible due to the difference of application field (RAPPOR), or the unavailability of reference implementations and original data set used in the paper (SSDD and Schnell’s solution). The data visualized in the plot confirm the fact that *ssdeep* was not well suited to process small inputs such as passwords. We claim that the proposed method can be integrated as a modular component in any kind of authentication system, to discourage the use of similar passwords over time and to prevent their use over different domains as pictured in Figure 4.13.

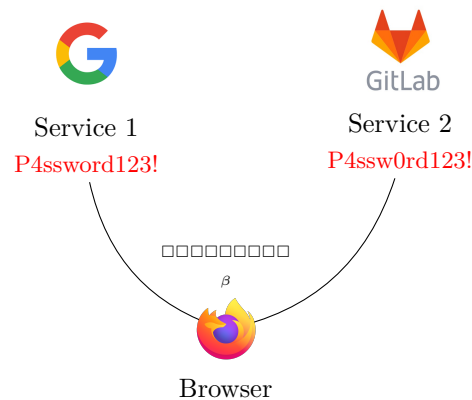


Figure 4.13: Application scenario: The browser can insert the passwords *P4ssword123!* and *P4ssw0rd123!* into the Bloom filter, checking if they are similar enough to throw a warning of password similarity.

The typical application could be a browser’s plug-in that issues a warning when the distance between a new password chosen by the user and the strings saved in the plug-in is below a pre-declared threshold.

The application can be instrumented to report the results of equations in Section 4.2 regarding not only the similarity check result, but also the various parameters characterizing the filter, to evaluate the quality of the classification process.

4.3 Insider Threat Analysis

In this part of thesis, we describe the initial release of an architecture for the automated deployment of a penetration testing environment including insider threat management features. The *Castrum Obsidiis* model is the result of a cooperation between Obsidium, a young start-up and a university research group at the University of Bologna, Italy to devise a system incorporating scientifically sound, state-of-the-art methodologies, at the same time taking into account real-world needs.

For the reasons described in 2.4 we decided to focus mainly on the detection phase, implementing an efficient way to deploy a monitoring infrastructure for suspicious insider activities, based on a customer-oriented insider definition. We argue that this approach is more cost and resource efficient.

What we argue is different in our approach is not the definition of an insider threat behavior but rather how we are able to detect it within a penetration testing scenario.

The architecture we are going to describe has two elements worth highlighting, namely: (i) the inclusion of an easily extensible set of detection tools, and (ii) the automated deployment of said tools and of the complementary data gathering and processing infrastructure.

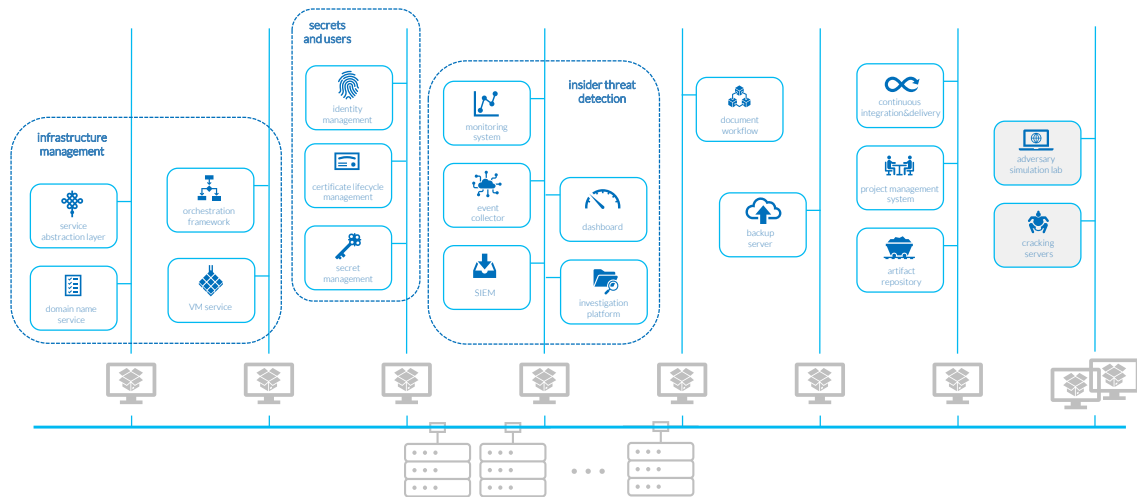


Figure 4.14: Infrastructure layout with the main functions implemented by the insider threat detection platform.

In Figure 4.14, the main elements of the insider threat infrastructure are depicted. Automation is of paramount importance to ensure the consistent deployment and management of all the needed components. Besides technical correctness, it also provides a guarantee that all employees' work will get profiled to detect anomalies in behavioural patterns, acting as a deterrent.

The approach to automatic configuration is based on the Infrastructure as Code (IaC) paradigm. As described in Section 4.3, using such a technology it is possible to configure the entire infrastructure from a single centralized system. The attack surface of the provisioned systems and the process itself is accordingly reduced, since manual code execution is entirely disabled on the configured hosts.

This approach grants the possibility of tailoring security configurations with the most appropriate method for each component, limiting conflicts between subsystems and eliminating the need for manual replication.

During its life-cycle, the infrastructure is going to be continuously updated. To properly manage this process, an automated supply chain that can replicate the production environment and audit the developers changes is needed. To ensure traceability and cope with the incremental development of the infrastructure, our proposal uses a role called *Infrastructure Source Management*. This role is in charge of maintaining the versioned source code and execute analyses on code smells. These automatized analyses and Continuous Development (CD) cycle could be reflected in test and production environment interacting with the provisioning entities, lowering the complexity of

managing a flexible and non-trivial infrastructure.

Although proven to be effective, the CD approach could exhibit many pitfalls regarding secrets management. That is, a secret obviously requires specific manipulation, different from what can be applied on not-sensitive data. For instance, an encrypted enclosure and a correct erasing process must be setup to protect secrets from offline attacks, like forensic analysis on disks.

The *Secret Management* role employs these measures, providing a standardized way to access data from clients, maintaining security as the first principle, and automating as many aspects as possible, to reduce error-prone manual intervention at the bare minimum required by specific processes.

The services of the infrastructure should communicate with the other roles using domains names, not lower layer addresses. That is, a standard way to list services, roles and contact the machines using a communication based on their role, and not on their physical network location, should be employed. This will also enable the distribution of the platform over multiple physical networks/clusters and the development of scripts and tools with the concept of role, not based on communication technologies.

Similarly, users and their roles are managed through a database, conforming to the *Identity Management* role, which is mainly used by the *Secret management* and *Infrastructure source management* roles. Therefore, the *Identity Manager*, will supervise over the access to the other roles, enabling a centralized control over the set of credentials that can be used to access the platform. Exploiting its centralized nature, it can be efficiently audited, looking for new users and for password changes, triggering alarms in case of unexpected or unsolicited credential changes.

The aforementioned components functionally manage their own data-sets, producing a detailed audit trail for the *Log and analysis* role. Upon the reception of logs from the other machines, it will save them in an unalterable vault and analyze them, searching for anomalous patterns. Whenever a log entry or combination thereof raises suspicion of malicious activity, the insider threat managers will be notified, giving them the possibility of further investigating and suggesting a response.

The last part of the infrastructure, that is not explicitly presented in Figure 4.14, are the probes placed on the roles and users machines. These elements retrieve statistics like process owners, the amount of RAM and CPU used, the suspicious packets that are leaving the network, etc.

Infrastructure as Code and environment replication

To set the entire infrastructure up, we used a set of provisioning tools according to the Infrastructure as Code (IaC) paradigm. IaC entails a process based on the concept of maintaining the infrastructure description as source code, enabling the use of techniques such as versioning (e.g., with software like `git`) and code auditing. That is, this approach can be pictured as a series of scripts to play to configure the system.

Most importantly, all these operations are traceable through a Version Control System (VCS) which ensures the portability and replication of such infrastructure. IaC ensures continuity, as all the environments are provisioned and configured automatically, which greatly speeds up and simplifies infrastructure tests.

Seamless updates positively affect security too, since legacy components gone useless and bugged software for which patches exist are immediately taken care of. IaC also brings Continuous Integration (CI) capabilities along. CI and IaC combine to enable rapid provisioning and configuration of the environments where code is developed and tested; new tools and features can be reliably integrated into the main project trunk, following flexible delivery roadmaps, without jeopardizing system performance and stability.

Traffic inspection

One of the core features of our proposed architecture is traffic analysis via deep packet inspection. This kind of monitoring could be quite expensive if centralized on a dedicated machine, which may need to employ hardware offloading or proprietary solutions to accelerate the analysis. A different way to tackle the computational load problems is to distribute traffic analysis on the single hosts, feeding pre-processed results to the Log and analysis machine described in Section 4.3. The drawback of this solution is clear: we shall trust the hosts that analyze the traffic, which are in the hands of potentially malicious insiders, and thus subject to being tampered to not send true logs. On the other hand, the centralized solution exhibits the dedicated inspection machine as a valuable target; it could hog the network sending raw, unprocessed traffic from the probes to the inspector.

Given these preconditions, we chose to implement a distributed approach by default, with the possibility to switch to a centralized DPI platform.

It is important to highlight that the proposed architecture foresees specific organizational roles for each activity. An important role will be that of *insider threat manager*, the person(s) in charge, and the only authorized one(s), to receive notifications of suspicious events and to be able to access the related details. The insider threat managers should also participate in a continuous analysis of

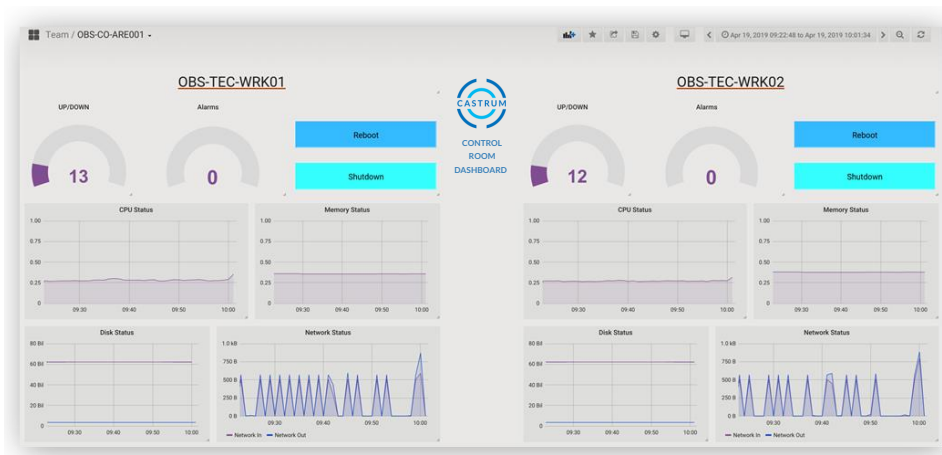


Figure 4.15: Grafana dashboard, the two circles indicates the number of process running in the machines and the number of alarms. At the bottom the information on the status of the machines is presented such as disk space, memory and CPU usages. The two buttons could be used to power off and reboot the machine promptly.

the protocols that should be allowed and subjected to inspection.

In this work, to provide a complete case study, we focus on a specific threat that can come from insiders, i.e., data exfiltration.

To detect sensitive data spill-out, we need to identify it in the flowing traffic. A well-known method to to achieve this goal is using the so-called *HoneyTokens* [364].

This countermeasure is obviously not a silver bullet against exfiltration, as the tokens could be identified and removed from the outgoing stream, but we argue that, in a constantly monitored environment, with a limited time, an attacker unaware of the tokens will trip onto them with high probability.

Relying only on the tokens would be security by obscurity, therefore we see them as an addition to any other analysis that can be made, such as raising warnings for any access to specific file.

This infrastructure has already been tested on several real-world use cases. The goal is not to illustrate the whole process of testing a specific target, but rather to show some of the main interesting results that we are able to proof.

The main threats we have analyzed are the ones we identified as preminent for this scenarios, namely Network exfiltration and External memory exfiltration. Other methods, such as, Physical exfiltration using covert channels, described in [190] or that use creative ways to generate audio

signals like FanSmmitter [189] - are not taken in consideration in this, initial release.

As a demonstration for the possibility introduced by our analysis platform, we have evaluated Network exfiltration via secure channels, such as HTTPS. We argue that this kind of threat is a good candidate for the evaluation of our system, for the measures that every modern browser act to remove the attacks possible by eavesdroppers.

The analysis of HTTPS can occur on different levels. One of the most common and reliable ones is to use a MitM attack on the protocol [87], without introducing weak ciphers or reducing the overall security of the protocol. To implement this probe, we had to intervene on one of the measures introduced to defeat the MitM attacks, the so-called *Certificate Key Pinning*: the browser keeps the certificate of critical sites in its internal environment to ensure that a MitM attack will not invalidate the security of the system.

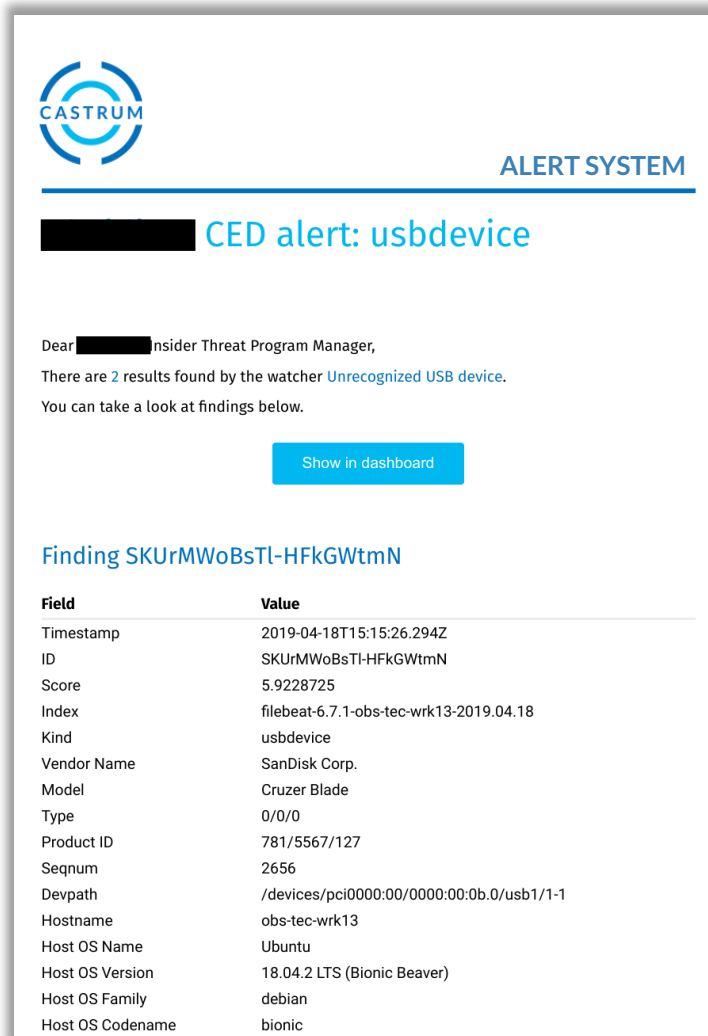
Even without key-pinning, SSL/TLS traffic is not readable by an attacker with only eavesdropping capabilities. This attack is prevented by a technique called Perfect Forward Secrecy.

Disabling this kind of support will reduce the security of the system, therefore, the implementation of the network inspection part of the platform relies on the termination of the connections using SSLsplit and the inspection of the packets on the SSLsplit server. The inspection acted by this server can be split in three parts: the first component redirects all the encrypted connections to the SSLsplit server which can therefore open it using its certificate; then, it analyzes the contents using a classic *Network intrusion detection system*, in our case Zeek; finally, SSLsplit will re-encrypt the traffic, sending it to its original destination.

As stated in Section 4.3, the analysis of traffic can burden a system with excessive load if it is in charge of analyzing the entire list of packets that roam the network. As we consider the monitored machine under control, we offload the analysis of the traffic to the single machines, distributing the load and double checking this traffic behaviour against the statistics sent to the logger machine. To optimize the network load and the work of the analysis machine furthermore, we have designed a filter for the Network intrusion detection module to search for the HoneyTokens.

When encrypted traffic is detected, it will be sent to the analysis platform as a warning. After that, regardless of the previous choice, the detection system will check if a token, pre-injected in databases or in critical files, is present in the traffic. If the token is found, it means that specific data are being exfiltrated.

The External memory exfiltration detection exploits the same concepts seen for network exfiltration. Because the insertion and removal of external devices is a critical operation, it has more



ALERT SYSTEM

██████████ CED alert: usbdevice

Dear ██████████ Insider Threat Program Manager,

There are 2 results found by the watcher **Unrecognized USB device**.

You can take a look at findings below.

[Show in dashboard](#)

Finding SKUrMWoBsTI-HFkGWtmN

Field	Value
Timestamp	2019-04-18T15:15:26.294Z
ID	SKUrMWoBsTI-HFkGWtmN
Score	5.9228725
Index	filebeat-6.7.1-obs-tec-wrk13-2019.04.18
Kind	usbdevice
Vendor Name	SanDisk Corp.
Model	Cruzer Blade
Type	0/0/0
Product ID	781/5567/127
Seqnum	2656
Devpath	/devices/pci0000:00/0000:00:0b.0/usb1/1-1
Hostname	obs-tec-wrk13
Host OS Name	Ubuntu
Host OS Version	18.04.2 LTS (Bionic Beaver)
Host OS Family	debian
Host OS Codename	bionic

Figure 4.16: Alert originated by the insertion of a not allowed USB device. In this case the metadata of the analyzed machine and USB dongle are presented in the alert report.

strict policies implemented by default in our platform.

A difference between network and External memory exfiltration is that the former is directed to a potentially huge number of endpoints that could not be blocked a-priori, e.g., by constructing a block list in advance. The latter is at the opposite side of the spectrum, the employee can declare a list of external devices that he intends to attach to the computer, and a manager will then proceed to evaluate the possibility of registering them in a predefined allow-list.

When a device is inserted or removed, the system will register the event information in a log file. For instance, in case of USB devices, the log will contain the timestamp of the insertion or the unplugging, and information on the device such Vendor and Product IDs. This log will be sent to the analysis machine. If this device is not present in the pre-compiled allow-list, an alarm will be triggered and sent to the responsible team.

We focused mainly on USB devices, but the procedure can be extended to every external device that has some fixed information and issues events in the Linux operating system: for instance, logs in the kernel buffer which are readable using `dmesg`.

Performances and Evaluations

To evaluate the usability of the method, we have used our implementation in a test environment. To get a more balanced picture, we let different users groups perform realistic work; some of them were aware of the monitoring platform, some were not. In detail, the overall effectiveness of the platform was tested over 4 teams using 4 machines each, for a total of 12 monitored Unix machines. We tested the system asking users to insert unregistered USB devices in the running systems and to exfiltrate a sensitive file, (`/etc/shadow`) pre-loaded with HoneyTokens, from the analyzed machines using plain-text (HTTP) and encrypted (HTTPS) connections to public websites. In all these cases, the alerts were promptly reported to the insider threat manager to immediately analyze the anomalies with the help of the Grafana dashboard or the Kibana query platform.

An example of the alert and the associated metadata which are sent to the email address of the insider threat manager is visible in Figure 4.16. In this e-mail, various metadata, useful to identify the target device and machine, are presented to the insider threat manager. A link pointing to the relative Grafana dashboard is available, giving to the user the possibility to investigate more on the alert and take actions as soon as possible. This enables a prompt response of the insider threat manager, that can SSH into the target machine as the administrator and stop the exfiltration.

In Table 4.17 the performance impact of the implementation of traffic inspection described in Section 4.3 are shown.

This test was conducted using a VirtualBox virtual machine with 2GiB of RAM, a single virtual (uncapped) CPU and our custom Kali Linux distribution. The hypervisor was running over an Intel Core i7 CPU 870 with a nominal clock frequency of 2.93GHz. During these runs the machine under test was executing common penetration test tasks, such as network scanning using `nmap`, bruteforce attacks using `hydra`. To test the analysis of unencrypted and encrypted network traffic the machine were running speed tests and file download over HTTP and HTTPS.

Software	Avg. CPU	CPU max	Avg. Mem	Mem max
Zeek IDS	22.45%	100.00%	35.65%	62.81%
SSLsplit	2.83%	12.50%	0.27%	0.34%
Traffic Mirror	1.18%	1.89%	0.15%	0.34%

Figure 4.17: SSL Inspection performances. The test was conducted over one hour of network downloads, brute-forces and network scans.

The results show that the system load is bearable on modern hardware; most of the processing power is required by Zeek, while SSLsplit and the inspection chain are not relevant for the system load.

4.4 TechNETium: Formal network verification using SDN

For the reasons highlighted in 2.2, an important part of the work presented in this thesis will be focused on improving the current state of the art tools to attenuate or remove such limitations. That is, to formally verify policies on the network, we need to have a system which can be used to express, translate and check formal constraints on the network topology. We will see that these constraints can be expressed in terms of Reachability or combined rules.

Our contributions include:

1. **HashTable:** a “reverse” tree is kept in a hash-table for each reachability tree. This table correlates a port and the reachability tree nodes that refer to it, for quicker update of trees.
2. **Ports along the route:** all common ports are kept in the path from s a d , to ease loop detection.
3. **Complementary set of atomic predicates:** if a set of APs, used to represent a predicate, becomes larger than half of $S(true)$, save the complement of the set with respect to $S(true)$ instead of the original set, to save space.
4. **OpenFlow Integration:** is it possible now to feed the BDD with OpenFlow rules taken from the SDN controller (as it is shown in the next section using ONOS).
5. **Multiple Match Key Forwarding Rule:** Thanks to the OpenFlow and ONOS integration we are also able to use three different types of matching keys for the forwarding rules: exact-IP, ternary, and long-prefix match.
6. **Improved domain atomic predicates to integer translation:** We rewrote the algorithm for the domain conversion from to integer tree implementing a cache that keeps track of the last conversions.

In this chapter, we will show TechNETium: our security policy checker tool. It is designed according to Model Driven Development (MDD) [136, 115], an approach that adopts Uniform Modeling Language (UML) as domain-specific modeling language. The goal of MDD is to model entities, relations and behaviors of the elements of the system as high-level abstractions. In the MDD approach, the definition of the specific domain of the problem to be modeled is required. In TechNETium, this involves both security and networking verification tools.

TechNETium exploits a network model-checking based on policies. By policy, we mean a variant aspect of any network (forwarding) mechanism, i.e., a desirable high-level goal expressed in the form of a rule or a configuration. We build a network model to formally check if those policies are met. The basic idea is to create and maintain a model of the network and a set of policies that such model must satisfy. Consequently, the model allows us to continuously verify whether or not the policy set is satisfied on the network. As specified in the previous section, we consider the SDN paradigm as the enabler for this technology, since it natively exhibits the separation between the data plane level and the control plane.

TechNETium aims to be portable and architecture-independent. For this purpose, it is written in Java; however, to perform a first test we choose a specific architecture on which to deploy it: ONOS [60]. which has also been proven as one of the most DoS-tolerant SDN controllers [22].

TechNETium is a tool composed of two main modules that work on two levels of an SDN architecture stack. At the Data Plane level the tool, through the predicates of a typical SDN controller, it generates a representation of the network model using the previously described AP technique. The result of this graph will, therefore, represent a snapshot of all the possible paths of a data traffic divided according to the flow rules of the switches.

The Control Plane level module, on the other hand, will input a “high level” security policy defined and described by the user and / or network administrator, and will perform the verification of this by computing it on the AP graph, in the form of Reachability and ToWayPoint queries.

Atomic Predicates BDD generator

The first part of TechNETium creates the APs trees. Following what has been described in section 2.2 according to the APs theory, we need two distinct sets of data to generate it:

- Physical data related to switches and hosts (links, ports, interfaces);
- All the installed flow rules.

From these two sets, a PolicyGraph is then created. It represents the whole set of possible paths of a packet within the network, according to the currently installed flow rules. As underlined in 2.2 we have been inspired by the AP Verifier tool described in [487], but we introduced many

improvements in terms of performance and efficiency.

The BDD generator has been integrated as a core service of the ONOS Engine. In this way, we can easily query the extended classes of the Topology module of ONOS to query it and obtain the data set of the network devices, links and hosts and the correspondent flow rules, to create the BDD tree of APs.

Policy Checker Application

Once the BDD tree has been created, it is possible to verify the security network policy. The strength of TechNETium is that a policy is simply seen as a property of the network. A property can then be formally verified on its representation as BDD. However, to make the whole process automatic and user-customizable, TechNETium can interpret the high-level policy defined by the user by breaking it down into elementary graph operations. The user is enabled to define a policy as desired properties regarding the routing of network traffic through the various nodes, while the policy verification is efficiently implemented as a sequence of Reachability checks.

The Reachability policy can be verified as a property of the BDD tree. The APs representation allows us to calculate the set of packets that can reach a destination port starting from source port using the following algorithm:

Algorithm 2 Reachability algorithm from port s to port d

Input: $S(F_1), \dots, S(F_j)$: quotient space representation of forwarding predicates among the path s and d (F_1, F_2, \dots, F_j)

Input: $S(A_1), \dots, S(A_k)$: quotient space representation of access control predicates among path between s and d (A_1, A_2, \dots, A_k)

Output: set of packets that can effectively reach d from s

```

1:  $S_F \leftarrow S_F \cap S(F_1) \cap \dots \cap S(F_j)$ 
2: if  $S_F = \emptyset$  then
3:   return false
4: end if
5:  $S_A \leftarrow S_A \cap S(A_1) \cap \dots \cap S(A_k)$ 
6: if  $S_A = \emptyset$  then
7:   return false;
8: end if
9: return  $S_F, S_A$ 

```

The FullReachability and the ToWayPoint policy as a consequence are decomposed as high-level policies to a set of reachability ones. This because when a model checking tool is able to construct the reachability graph of a system, it can in principle answer any reachability question by simply examining this graph [86].

The FullReachability policy is just a simple example of this kind of decomposition. It expresses the

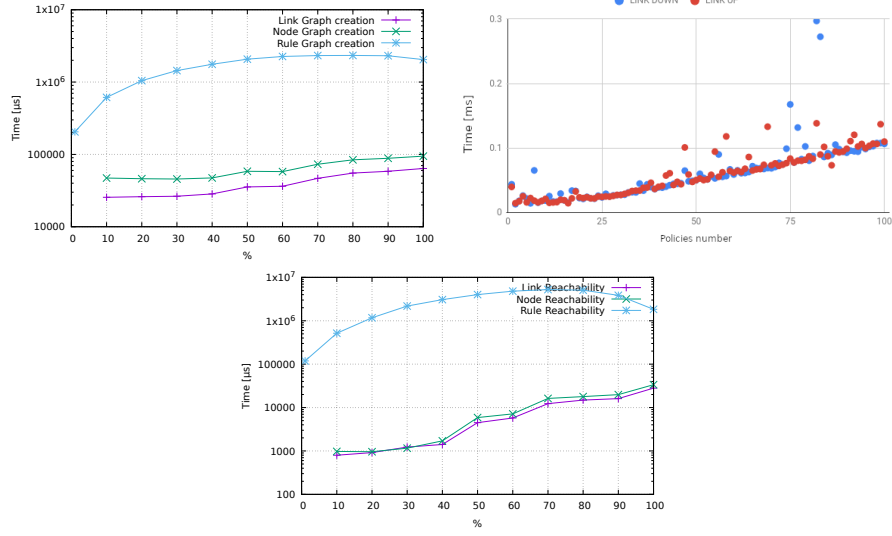


Figure 4.18: Average BDD Creation time / Policy time check after a link up / link down upgrade and node, links and forwarding rules increase.

Reachability from A to B in a duplex way, for this reason, is it simply calculated as:

$$FullReach(A, B) = Reachability(A, B) \wedge Reachability(B, A) \quad (4.2)$$

and no need for an additional reachability graph is necessary.

The ToWayPoint policy is more complex and it calls for a modification of the reachability graph. In order to be expressed in terms of Reachability, ToWayPoint is defined as follows. Considering:

$$\alpha \neq \beta, \omega \neq \alpha, \omega \neq \beta \quad (4.3)$$

$$Reachability(\alpha, \omega) \wedge Reachability(\omega, \beta) \wedge \forall \bar{\omega} \neq \omega | \neg Reachability(\alpha, \bar{\omega}) \vee \neg Reachability(\bar{\omega}, \beta)$$

These policies, however simple, can be considered the first step for a network administrator to define his security policies for network management. In fact, considering the most common network attacks such as distributed / reflection denial of service, network administrators can use this policy in a proactive or reactive mode.

Performance

In this section, we will describe some of the main experiments/tests we performed.

The goals of such tests were mainly 3:

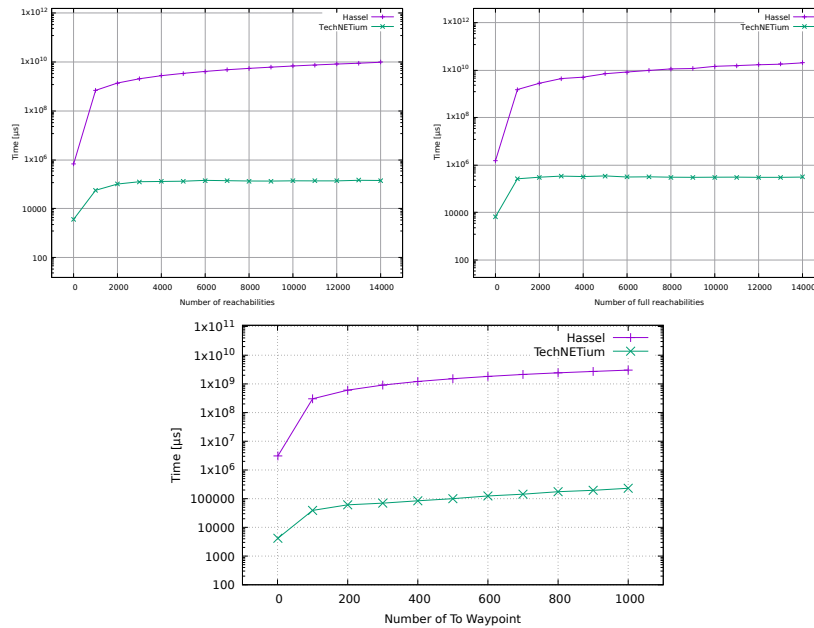


Figure 4.19: Policy time check after a link up / link down upgrade and node, links and forwarding rules increase.

- Showing the performances of the network graph creation.
- Showing the performance difference for reachability verification with related works such as NetPlumber.
- Showing the performance difference for generic policy verification with related works such as NetPlumber.

We decided to compare only our proposal with the NetPlumber solution. The main reason is that NetPlumber appears to be the most cited, documented and tested solution for formal network verification. Other solutions cited in the background section (2.2), instead have several problems. Veriflow and Delta-Net e.g, nowadays do not seems to have open-source code available for private tests, and other works such as Minesweeper [52] have been already tested with NetPlumber [237]. The experiments were performed on several VirtualBox dual-core virtual machine with 4 GB of RAM and Ubuntu Linux 18.04 operating system. The network sample that we used is the i2 Standford Network¹⁸.

This network is composed of 161 Nodes, 11450 Links, and 77451 Forwarding Rules.

¹⁸<https://uit.stanford.edu/service/network/internet2>

The first set of tests has been made to verify the performances on the BDD graph creation. We executed TechNETium increasing the amount of 3 different items: links, nodes and rules. As a result of these numbers, we have therefore measured the creation time of the BDD fee.

On the first graph of figure 4.18, we can see the required time for TechNETium to create the BDD, as we predicted, the growth of creation time is roughly linear in the number of links and nodes. Considering the forwarding rules, instead, the time increases linearly up to 60% of the rules, then it stabilizes. This is due to the fact that we increase the forwarding rules shuffling them at every step but in an equivalent number for each device. For this reason, after a certain threshold the complexity of the network due to the calculation of all possible paths of a packet becomes constant. The second set of tests that we performed aimed to calculate the policy verification time while upgrading the reachability graph. We calculated the time required to verify the same policy changing the network graph as we previously did.

The remaining graphs of figure 4.18 shows the results of such tests. The left graph shows the time to verify an increasing number of Reachability policies when adding or removing one link. These events modify the reachability graph, yet the verification time still increases linearly with the number of policies only.

The graph on the right confirms this result generalizing the test approach. The verification time linearly increases with the cardinalities of the sets of nodes, links, and forwarding rules. There is only an anomalous behavior after the inclusion of 70% of forwarding rules set. In this case, the behavior is due to the fact that, as the forwarding rules increase, many more direct links are configured so that the reachability is much more likely to be successful in less time. Finally, we tested the performance difference between the NetPlumber tool (based on Hassel header space analysis) and TechNETium with atomic predicates. From graphs in figure 4.19 we can, therefore, infer some important data. Policy verification with TechNETium is 3 orders of magnitude more efficient. The growth is also linear for both, therefore the performance difference grows with the number of policies to be verified. Furthermore, linear growth is different. While NetPlumber shows a growth, albeit small, in the verification times with the increase in the number of policies due to the size of the reachability graph this is much less evident on TechNETium, whose growth is much less pronounced.

4.5 P-SCOR: Artificial Intelligence and Programmable Data Plane

To defeat attacks and increase the security of a network, the flexibility of Programmable Data Plane can be combined with constraint programming, in a similar way to the system proposed in Section 4.4. This system can be considered an extension of TechNETium, built using P4 and Constraint Programming (CP) checkers.

The proposed CP orchestration discussed in this part of thesis is an extension of the SCOR system introduced in [260].

SCOR was implemented in Minizinc [331]. In addition to its pre-packaged solvers, e.g., Gecode [399], MiniZinc can utilise other available solvers, such as Jacop [253] and ECLiPSe [418].

SCOR provides a new SDN northbound interface with powerful CP-based abstractions that allows complex routing problems to be expressed in only a few lines of code. More importantly, the problem only needs to be declared, while the solving of the corresponding constraints satisfaction problem is delegated to the powerful, general CP solvers, provided all data are available.

As an example, let us consider *minimum delay routing*, where we aim to find the path with the minimum end-to-end delay between two network nodes. As discussed in [261], this routing problem can easily be expressed and solved in SCOR. However, what is missing is the information about link delays in the network. This information is currently not provided by SDN controllers. This is mainly due to the limitation of OpenFlow, which does not provide the required functionality for data plane instrumentation and monitoring. With P4 we can overcome this limitation, by utilizing its capability for in-band data plane monitoring. By integrating P4 and SCOR, P-SCOR allows combining the benefits of both.

P4 has the ability to monitor critical link information from the data plane, and can provide this information to SCOR, which in turn uses it to very efficiently implement QoS routing applications, such as those discussed in detail in [260]. The focus of this work is neither P4 nor SCOR, but the integration of the two, which is demonstrated in the following sections via two use-case applications.

The P-SCOR architecture is summarized in Fig. 4.20, showing the three components (CP-based orchestrator, SDN controller, and P4-based programmable data plane) and their mutual relationships, focusing on one of the key contributions of this work, i.e., the communication channels between the components.

The key components

Programmable data plane with P4

P4 is not a protocol or device API for run-time control or configuration, i.e., once a P4 program is deployed to a device, P4 does not offer primitives, for example, to add or remove entries in match/action tables, or to read the value of a counter.

To carry out this kind of tasks, the P4-runtime API [493] has been developed to interact with the program. The main purposes of this new standard API are:

- enabling run-time control of P4-defined switches;

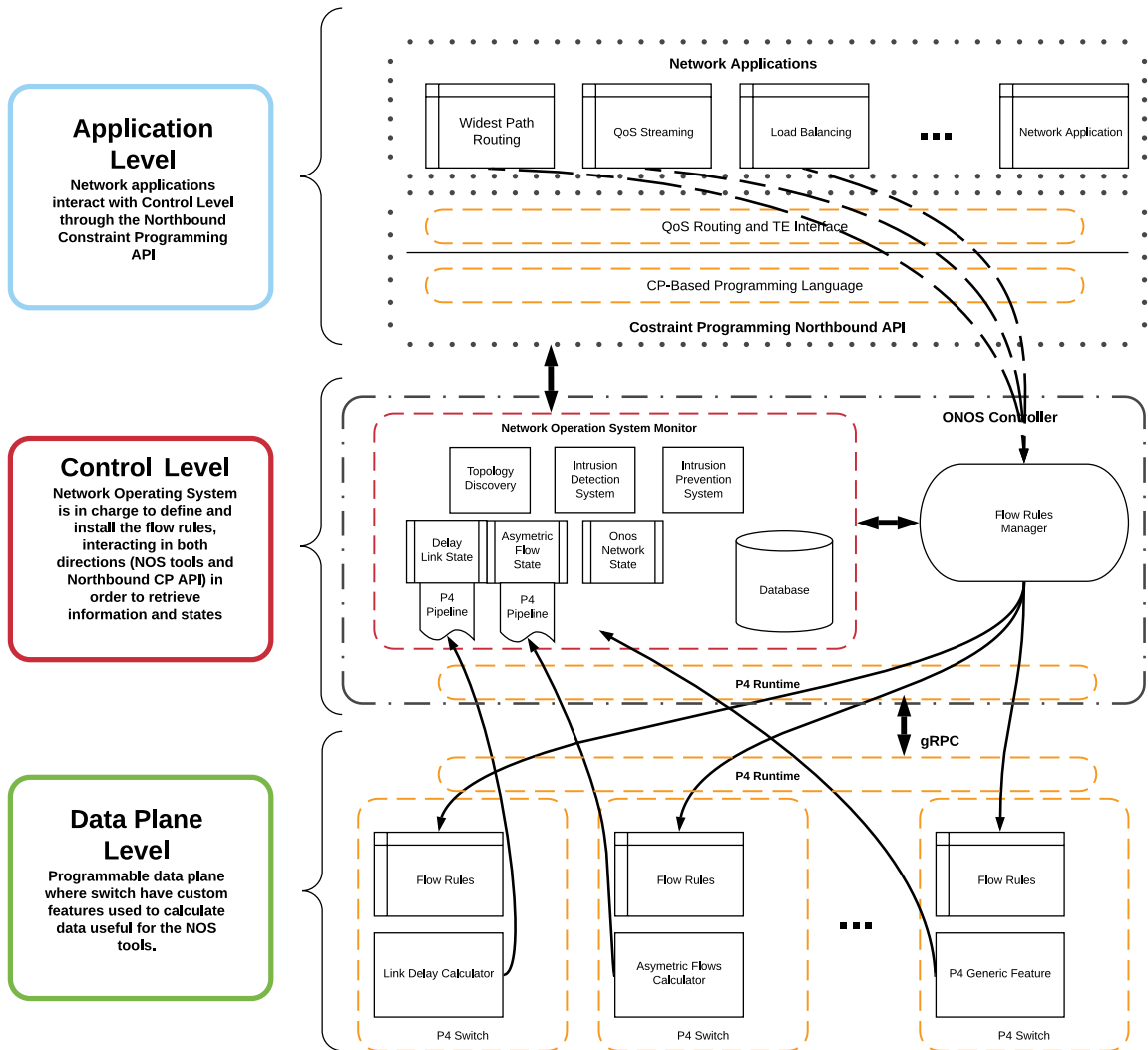


Figure 4.20: P-SCOR Overview of the three main component level

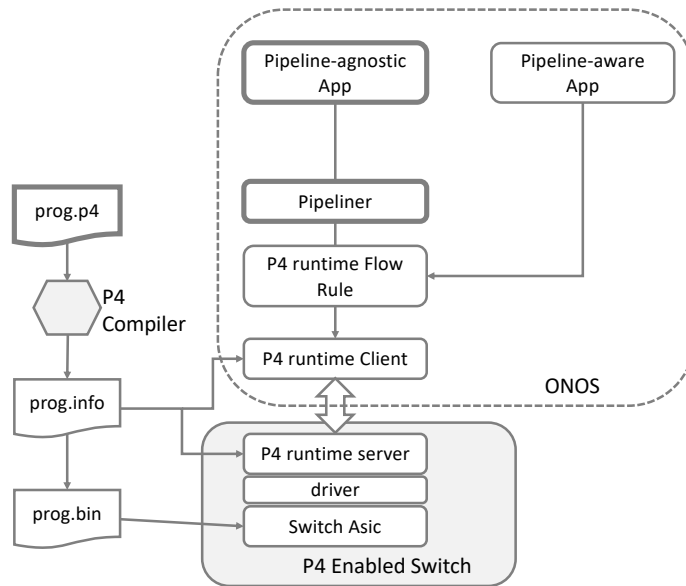


Figure 4.21: A block diagram showing the interaction of the various components with the integration between ONOS and P4 via P4-runtime and the ONOS pipeline. The components that have been originally implemented for this specific work are those highlighted with thicker lines.

- defining program-independent interaction (the API does not change if the P4 program is modified);
- enabling to push a new P4 program without recompiling the switch software stack.

It adheres to a client-server model; the server resides in the data plane, integrated within the switch. A client integrated within a local or remote control plane interacts with the server to load the pipeline/P4 program, write and read pipeline state (e.g., table entries, meters, groups, etc.) and sends/receives packets. P4-runtime uses a gRPC/protobuf-based language to define its own API, called `p4runtime.proto`. What P4-runtime needs in order to work is a set of specifications which are defined in the P4 program and retrieved at compile time. The typical workflow of such process is defined in Fig. 4.21.

A significant part of our work regarded this layer of the architecture, in which we exploited P4 to create an entity able to interact with the upper layer according to the specification required by P4-runtime. This P4 program is then used as a wrapper for the specific data plane functionalities that were implemented and tested, as it will be explained in the following section.

SDN control plane

ONOS was chosen to implement the SDN control plane. It offers a number of features, besides the basic SDN controller capabilities, that qualify it as a real Network Operating System. ONOS was used as:

- the network controller, i.e., a component capable of controlling a whole network composed of several nodes and with a general topology;
- the communication channel between the data plane and the CP orchestrator.

This was achieved by exploiting the work of the ONOS P4 brigade¹⁹ which implemented a component that allows the integration of any P4 program with ONOS.

The idea behind the P4 brigade project is shown in Figure 4.21. The pipeline-agnostic application is any sort of ONOS application that may collect relevant data from the network and apply relevant control plane actions. The *Pipeliner* is a wrapper that gives to the ONOS application the capability to interact with P4-runtime without the need to know the details of the P4 implementation (which would not be case for the pipeline aware application).

In this scenario the deployment of novel software functions can be seen as composed of three steps:

- a new P4 program is deployed; it becomes immediately available thanks to the P4-runtime Standard API but it is not automatically integrated with the controller applications.
- a suitable pipeline is designed and implemented to guarantee a proper communication with the P4 program (via P4-runtime).
- a new ONOS pipeline-aware application can be deployed at any time to access the P4 program functionalities.

In ONOS we can perform this task defining what is called an “ONOS pipeconf”. This component is essentially a regular application that can be loaded in ONOS at run-time and that, once loaded, registers the pipeline, which is the wrapper from the application to southbound API. Once registered, an ONOS application can add this registered pipeline to use a P4-runtime-capable device as shown in figure 4.21. This process gives ONOS the novel ability to create “wrappers” to P4-runtime.

We implemented all the necessary pipelines to allow the SCOR level to talk to the P4 plane. We wrote the pipeconf that tells ONOS the mapping between the table IDs, header fields, and the

¹⁹<https://wiki.onosproject.org/display/acrshort{ONOS}/P4+brigade> visited on May 27, 2020

instructions used in the FlowRule generated by the SCOR apps, and the P4 table names, match fields, and actions as in the P4 program, queried through P4-runtime.

If SCOR interacted only with the SDN control plane, as it happens in the original paper, it could not work on detailed run-time information, because the SDN controller would not have visibility of them. Most commonly, the SDN controller provides only aggregated information on the network behavior and high level information about topology etc.

On the other hand, were SCOR simply integrated with the P4 layer via the P4-runtime interface, it could access run-time information only on a per-device basis, thus missing the overall network view.

By integrating ONOS with P4 we enabled the collection of additional and more detailed information about network run-time properties in the SDN controller, which can pass them to the CP orchestrator implemented with SCOR. As a result, SCOR has access both to the high level network information provided by the SDN control plane and to the low level run-time and node-based information provided by the P4 program.

Consequently, CP becomes applicable to solve problems that could not even be stated otherwise because of the lack of the needed variables and constraints. Moreover, SCOR may instruct the SDN controller to inject in the network specific packets, built by means of the P4 program. The architectural enhancement we achieved, thus, extends its reach to the realm of action on the network flows; it is not limited to a better way of capturing and processing information.

Examples of Applications: Link Delay and Asymmetric Flow Detection

A test-bed of the P-SCOR architecture was implemented. The various components described above were all deployed appropriately, and in the data plane two P4 applications were implemented to test the effectiveness of the proposed approach:

- link delay measurement;
- asymmetric flow detection (possible DoS detection).

The knowledge of the *link delay* is needed by many routing protocols, and in packet networks is indeed one of the main indicators of performance. In a conventional SDN network, analysing packets coming from the data plane is time- and resource-consuming especially if high accuracy is needed [371, 22]. Most SDN controllers do not encompass a built-in application able to do that, and also the original test-bed of SCOR used hard-coded (i.e., emulated, not real time) link delays for the tests.

Asymmetric flow detection can be used as a warning of potentially incorrect network behavior and also to trigger a remediation action (in our test application, raising a warning or rejecting every

packet related to a network node identified as responsible). In SDN, the controller is informed only of the first packet of any new flow; this is enough to decide upon packet forwarding. Counting single packets per flow is not a viable feature of the control plane, as it would entail an unbearable overhead. With P4 we are able to perform such action in the device directly and not at the control plane level, sending only the eventual warning to the CP orchestrator.

Link delay evaluation

The overall latency a packet experiences when traversing a network is due to many different contributions [128]:

- Processing delay – time it takes for routers to process the packet header
- Queuing delay – time the packet spends in routing queues
- Transmission delay – time it takes to push the packet’s bits onto the link
- Propagation delay – time for the packet-bearing signal to reach its destination

Some of these quantities are known and unalterable; they either depend on hardware (e.g., the transmission delay is a feature of the network interface) or are physically constrained (e.g., the propagation delay is a function of the distance between the network nodes). The processing and queuing delays are the main random components and also the ones that can be controlled – and ideally reduced – by enhanced network protocols, scheduling policies, etc.. For example, MultiProtocol Label Switching [120] was introduced to implement dedicated virtual circuit connection (the Label Switched Paths) and reduce the time needed to take the packet routing decisions.

In general, the delay measurement can be [315]:

- **passive**, i.e., non-intrusive and based on capturing packets, in order to store and collect information from various fields within the packet header.
- **active**, i.e., by injecting probe packets, measuring the performance they experience, and taking it as representative of the performance of all the traffic.

The measurement strategy we opted for belongs to the active category. In standard SDN, active measurement is not practical, because the switches act just as forwarders and all the intelligence is in the controllers. Therefore any real life packet measure made at the controller is affected by the switch-controller delay that may impair the measure reliability. P4 shows all its potential in this scenario, because it allows us to perform an active measurement in a simple and automatic way at the switch level.

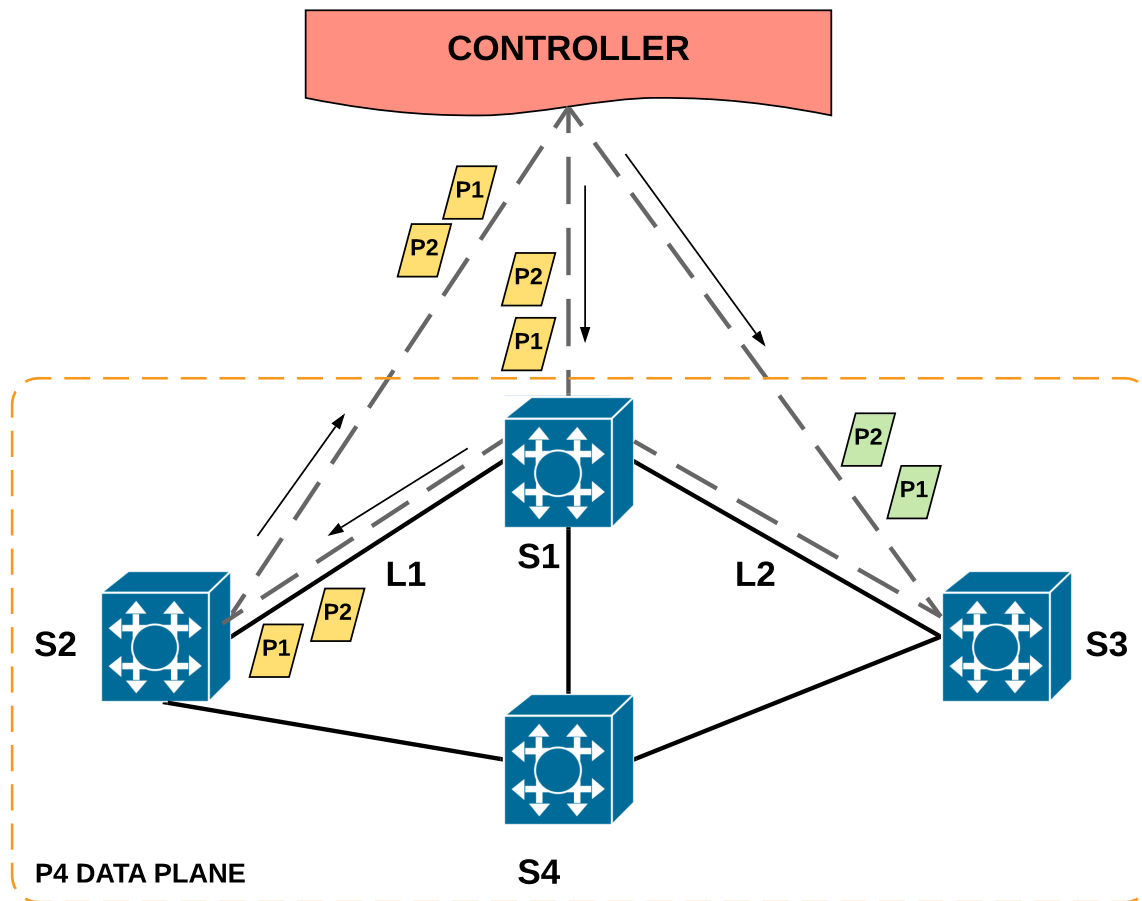


Figure 4.22: Delay Link program work-flow

Since the controller knows the network topology of the data plane, we chose to use a packet-probing technique to estimate the delay of each link of the network, following [372]. The idea is simple:

- the controller targets a link to measure the average delay;
- the controller picks the two switches at the ends of the target link, called $S1$ and $S2$;
- the controller sends to $S1$ two packets, $P1$ and $P2$, to be sent through the target link to $S2$;
- the P4 program installed in $S1$ instruments the packets by adding specific custom headers, that will be used to calculate the delay;

- when the two packets reach $S2$, the P4 program there adds the results of the measurement, again as custom headers, and makes $S2$ send them to the controller as Packet-In;
- the controller gets the information about the delay on the link reading the packets.

The process is summarized in Fig. 4.22.

The custom headers added to the packets are called:

- *WHICH*: a header to identify the packet ordering;
- *WHERE*: a value to specify whether the switch which added this header is the source or the sink of the link;
- *PORT*: the port through which the source switch has to forward the packet to reach the sink of the link;
- *TIME*: the current time-stamp;
- *DELAY*: a time-stamp that is used with a different purpose on P1 and P2, in P1 it is used to calculate the total end-to-end delay of the packet, in P2 it is used to calculate the processing delay to be subtracted from the total delay of P1 to estimate just the link delay.

With reference to Fig. 4.22, assuming the target link is $L1$, an example of measurement can be described as follows:

1. the controller sends two packets to $S1$ – let us call them P1 and P2 – with the information about their destination embedded into the custom header *PORT*: in the form of the port number of $S1$ towards $S2$;
2. when P1 goes to the egress queue, a time-stamp is taken and saved in a register entry of $S1$ for $L1$;
3. P1 is forwarded to $S2$, ;
4. when P2 arrives at the egress queue, $S1$ calculates the difference between the current time-stamp and the time-stamp of P1; in this way we get an estimate of the processing time in $S1$, which is called TP_1 , and which is saved in the correspondent header field *DELAY* of P2;
5. P2 is also forwarded to $S2$;
6. $S2$ behaves in the same way as $S1$, in calculating the processing delay of the packets, but also knows it is the end node of the measurement from the custom header *WHICH*;

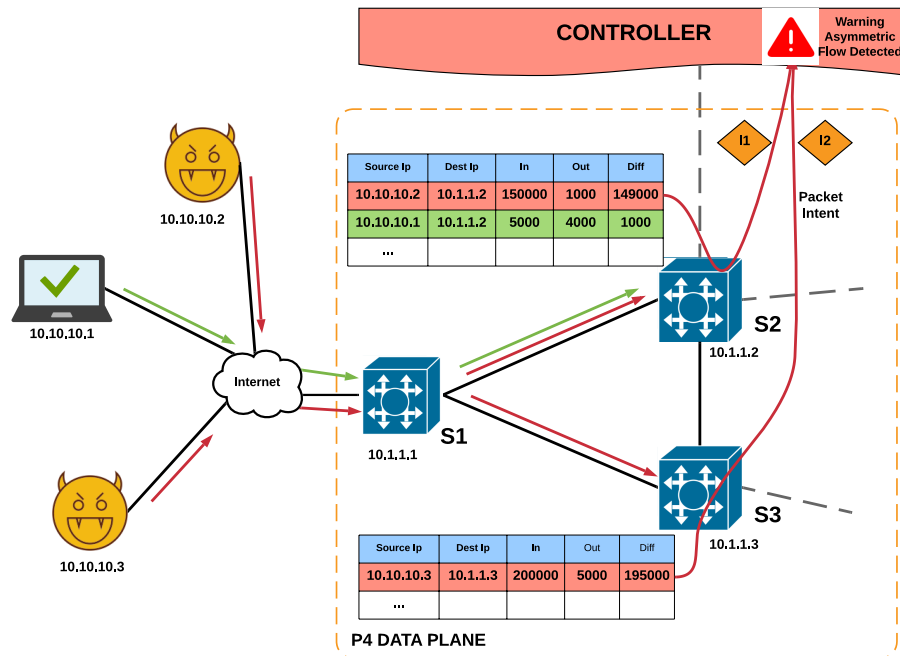


Figure 4.23: Asymmetric flow detection, example of work-flow. Three data flows are active and the P4 application is active in all switches. In S2 and S3, anomalies are detected since some flows show sensible asymmetries between packet numbers observed in the two directions.

7. the processing time in S_2 , called TP_2 is calculated as in S_1 and the total time P_1 and P_2 have been around are also calculated.
8. the delay due to the link is calculated as:

$$Delay = TT(P_1, P_2) - TP(S_1) - TP(S_2) \tag{4.4}$$

where TT is the total time required by the packets to cross S_1, S_2 and the link L_1 , which can be seen as the transmission plus the propagation delay.

9. the calculated value is stored in the custom header $DELAY$ of P_2 before sending the packets to the SDN controller.

In this way we get a measure of the propagation delay on the link plus the queuing delay, if any.

Asymmetric flow detection

The second application we implemented, as an additional example of the effectiveness of the P-SCOR approach, aimed at providing a real time aid to the identification of possible DoS attacks

to the network. This information is passed to the CP orchestrator that will make decisions about possible countermeasures, with an approach similar to the QoS routing already mentioned above.

It is well known that DoS attacks are a serious threat to the availability of networks; even more so in SDN networks where a DoS attack can be mounted against the controller and not just towards the network nodes and terminals.

Typical countermeasures rely on traffic measures to identify anomalies in the traffic profiles, such as for instance [474], [324], [144], [59], [58]. Solutions to safeguard the controller from the risk to be overloaded have also been proposed in the literature [249].

Nonetheless the implementation of DoS detection in the SDN control plane involves data storage and analysis which consumes memory, requires complex computations, and presents the risk of false-positives and false negatives [51].

By programming the data plane, for instance with P4, it is possible to overcome such limitations, implementing a program which is able to produce an aggregated result of a possible DoS attack warning. In this work, we implemented as an example a P4 program for asymmetric flow detection. It calculates the ratio between the amount of incoming and outgoing traffic for a specific IP entity. The basic idea is that a very large asymmetry is an indication of a possible DoS attack.

Figure 4.23 shows a schematic of how this works. The P4 program on the switches maintains a registry entry for each IP address or IP class of interest. The program works on a pre-defined time scale, storing packets flowing in both directions between two sets of destinations. If a severe difference is detected between packets flowing one way and packet flowing the opposite way, this is considered a possible anomaly, causing the switch to send a P4-runtime packet-in to the controller to raise a warning of a possible DoS attack. As an example, in the figure, three connections are monitored and only one of them, the one involving 10.10.10.1, is not malicious.

The threshold triggering the warning is implemented in a dynamic way, so that the controller can progressively track the level of the asymmetry and decide which action to perform. The threshold starts at an initial value, and every time the flow reaches it, the value is increased by a factor of 2. In Listing 4.1, a small block code of the P4 application is shown, performing the calculations and the checks for the threshold.

Listing 4.1: Threshold computation and checks using P4.

```
apply
  if (hdr.ipv4.isValid()) {
    ipv4_lpm.apply();
    ...
    window.read(last_time, flow);
    threshold.read(currentThreshold, flow);
```

```

// first time initialize
...
intertime = standard_metadata.ingress_
global_timestamp - last_time;
window.write((bit<32>)flow,
    standard_metadata.
    ingress_global_timestamp);
// check window
if(intertime > WINDOW){
    restore_flow(flow,flow_opp);
}
last_seen.read(last_pkt_cnt,flow);
last_seen.read(last_pkt_cnt_opp,flow_opp);
tmp = last_pkt_cnt - last_pkt_cnt_opp + 1;

if(tmp < (bit<48>)currentThreshold) {
    get_inter_packet_gap(last_pkt_cnt,flow);
}
else{
    // threshold is reached, drop it
    // (send packetin to the controller)
    if(currentThreshold > 1000){
        drop();
    }
    // else i increase your threshold,
    // and restore the flow
    else {
        threshold.write(flow,
            currentThreshold+200);
        // increase your threshold,
        // restore the flow
        restore_flow(flow,flow_opp);
    }
}
}
}

```

4.6 Test Bed and Experimental Results

P-SCOR was implemented in a virtualized environment on a server with Ubuntu Linux 18.04, 8 GB of RAM and dual core CPU.

The components used were:

- ONOS controller, version 1.14, the most recent version supported by the ONOS brigade community;
- P4 version 16;
- Bmv2²⁰ as switches supporting P4;
- mininet²¹ to implement virtual network topologies;
- some well known tools for traffic generation, such as MiniCPS [35], iperf²², hping²³, and the Python scapy library²⁴ to forge custom packets.

With this test bed it was possible to run and test P-SCOR with the aim to:

- verify the correct functionalities of the P4 program implementation and of the architecture as a whole;
- verify the overhead introduced by our solution;
- compare the P-SCOR solution with competing, existing ones.

The tests were performed on a simple ring network topology with three switches, unless otherwise specified.

At first we compared the performance of the switches with the P4 programs used by P-SCOR with conventional switches and with OpenFlow controlled switches. The goal is to check how much overhead (if any) is introduced by the P-SCOR components.

Following [64] we performed two different types of tests. We measured the average Round Trip Time (RTT) to transmit a set of ICMP packets of size 512 and 8192 Bytes. Each of these tests were performed 10 times for each set of packets, and the results are the average of the 10 rounds. The aim of such test is to measure the forwarding-behaviour performances of each switch. Is important to mention that for tests on the OpenFlow switches we had to limit the bandwidth to 18 Mbit/s.

²⁰<https://github.com/p4lang/behavioral-model>

²¹<http://mininet.org/>

²²<https://iperf.fr/iperf-doc.php>

²³<http://www.hping.org/>

²⁴<https://scapy.net/>

This because it was the maximum amount of stable bandwidth that we were able to obtain for the Bmv2 P4 switch and we needed a consistent environment for the evaluation tests.

These results are shown in Fig. 4.24, where the confidence interval of the measure is also shown as error bar on top of the histograms bars. The confidence interval is very small, meaning that the measures are very consistent. They show the rather obvious fact that forwarding long packets takes longer than forwarding short ones, but they also show that the switch exhibiting the worst performance, albeit for just a few ms, is the OpenFlow-based one.

This was an expected result, since the basic P4 switch is the lighter one; it implements only the essential forwarding reactive behavior and it does not have all the tables and features that a standard OpenFlow switch has. Otherwise our modified version of the P4 delay link switch includes only the new feature to process special crafted packets, and the performances are slightly worse than those of the basic P4 switch but a bit better than those of the OpenFlow one.

This is confirmed by the histogram in Fig. 4.25, where we measured the Total Transmission Time of bursts of packet of increasing length, from 20 to 200, both for packet size 512 and 8192 Bytes and in the same conditions as before. Again, the performances of the three variants of the switches are very similar; the times needed to send the train of packets are similar as well, since the transmission delay becomes negligible, when compared to the sum of the propagation delay of the all the packets in the burst.

The same comparison has been performed on the asymmetric flow detection switch. The results very closely matched the ones we just illustrated for the delay case, so we deemed not necessary to include an additional graph.

These first experiments therefore allow us to conclude that the performance of our switch is in line with the existing state of the art, despite the fact that we have introduced some important changes.

Table 4.2: The link delay calculated by the P4 program.

Number of Measurements	100
Interval Measurements	1000ms
Min value	6
Max value	110
Average delay	37.42ms
Variance	5676.79

The second set of tests was used to validate the correct integration of the three layers by means of the two applications implemented in P4. The link delay was supposed to be a key input even in the original implementation of SCOR, with the goal to perform QoS-based routing strategies to minimize the overall packet latency, but it was not implemented with real time measurement. The

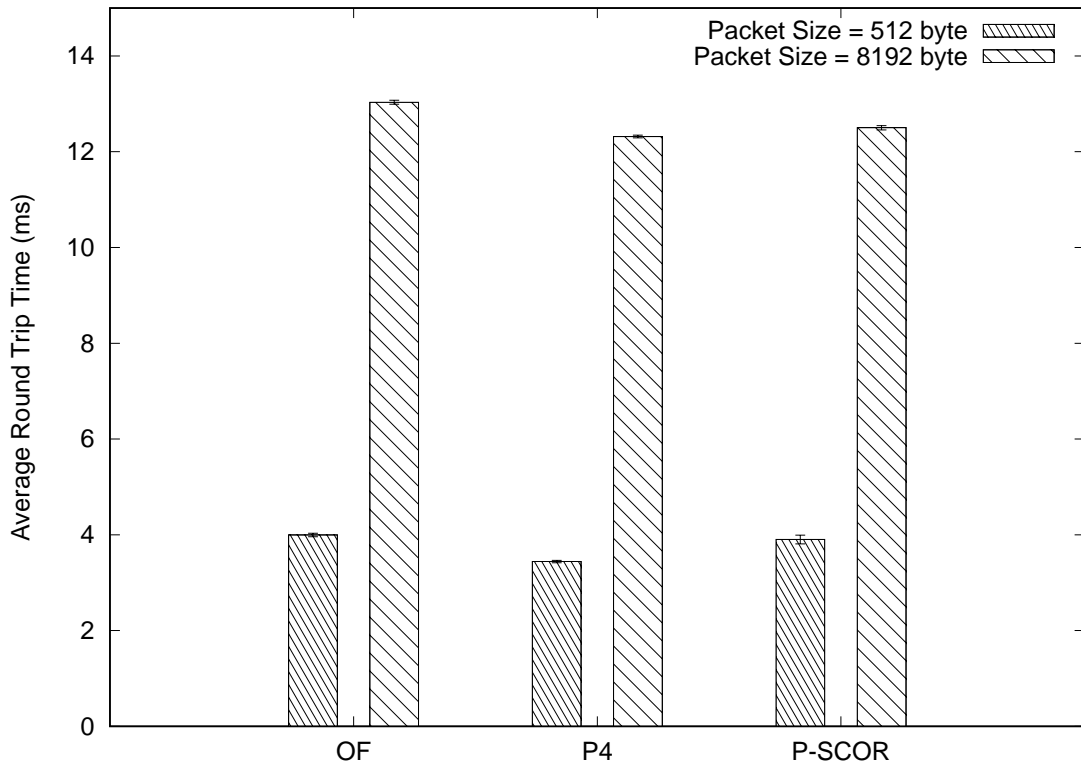


Figure 4.24: Comparison of the forwarding performance of an OpenFlow switch, a P4 enabled switch and a P4 enabled switch running the P-SCOR related programs.

asymmetric flow detection is used as an input to trigger remediation actions, in this case with packet drop as explained later. The CP orchestration programming will not be discussed here because it follows what presented in [260].

Table 4.2 summarizes the first set of tests on link delay measurement. We sent a set of four consecutive pairs of packet probes every second, and we measured the delay of the link. In this case we expected a small delay just due to the packet transfer in the virtualized environment. The average measured delay was 37.42 ms and is taken as a reference for the subsequent tests, in which a delay was introduced on the link, increasing from 1 second to 20 seconds in 1-second steps. The goal was to check that the application could keep measuring the correct delay. The results are shown in Fig. 4.26.

Here every point of the graph represents the average of 5 tests, therefore the 95% confidence interval of the measure is also shown, confirming that the measure is rather accurate.

The tests on asymmetric flow detection were also run in a similar way. In this case, providing

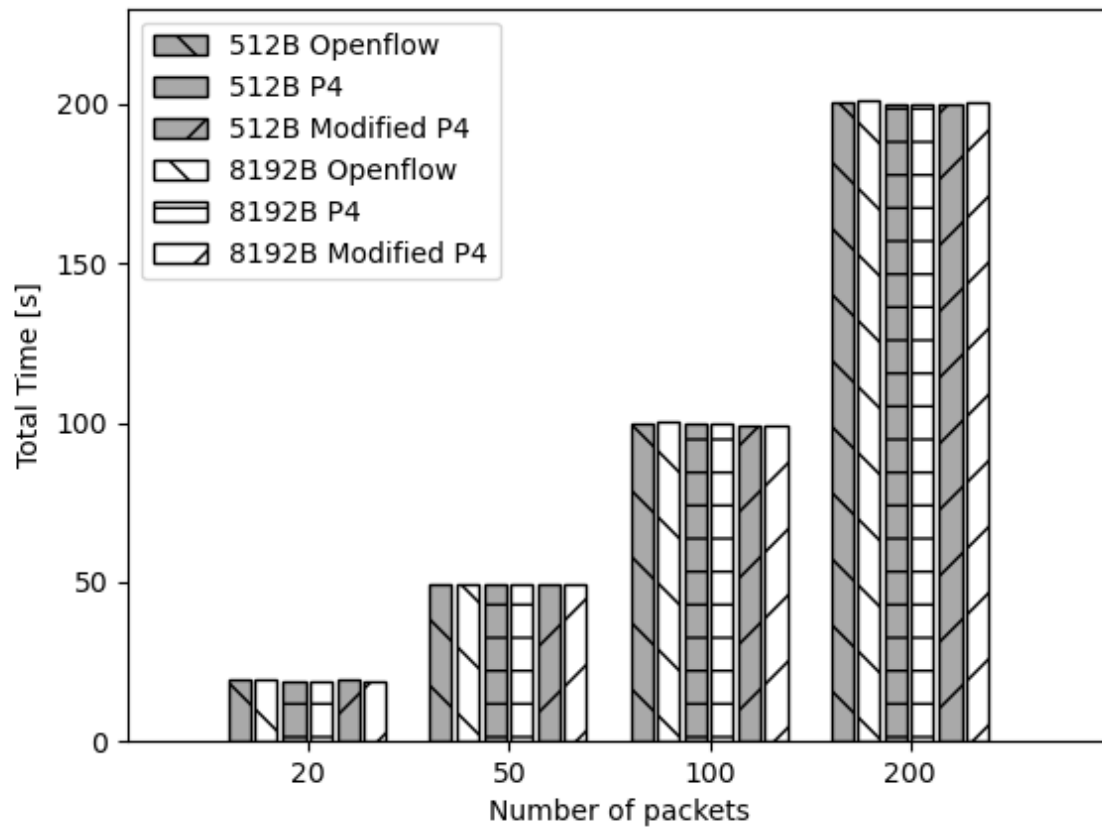


Figure 4.25: Comparison of the total time transmission time of a burst of packet (with burst size from 20 to 200)

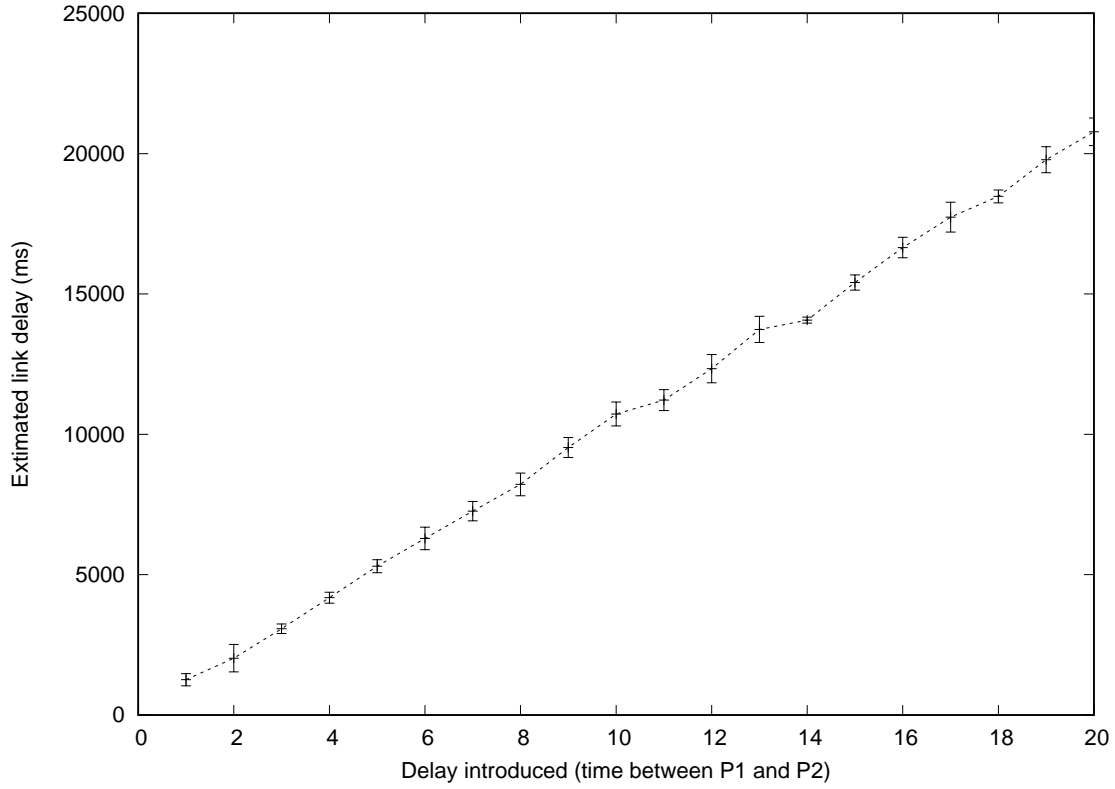


Figure 4.26: The link delay calculated by the P4 program, with an increased delay on the link from 1 to 20 s. The 95% confidence interval is plotted with the average. This confidence interval is quite good and was achieved with 5 experiments per point.

evidence of the effectiveness of the operations of P-SCOR is slightly more complicate. We set up the P4 application performing the tests on the flow asymmetry with a threshold T . A time window of $W = 15$ seconds was set. The time in the example is measured as a multiple of W , therefore $t = 1W$ means $T = 15$ seconds. Every W we use iperf3 to send $N(t)$ packets, a number increasing with t . These packets emulate the asymmetry of the bidirectional connection, i.e., the difference between the number of packets flowing in one direction and the number of packets flowing in the other direction. A warning of asymmetric flow detection is sent, as specified in Section 4.5, every time the asymmetry of the flow hits the intermediate threshold.

The threshold T starts at 300 packets per W ; it is increased by 100 packets at every window in which no warning occurs, up to a value of $T_M = 600$ packets. The CP orchestrator, upon receiving a warning, simply asks the control plane to drop the packets of the flow; the rule is then programmed in the switches.

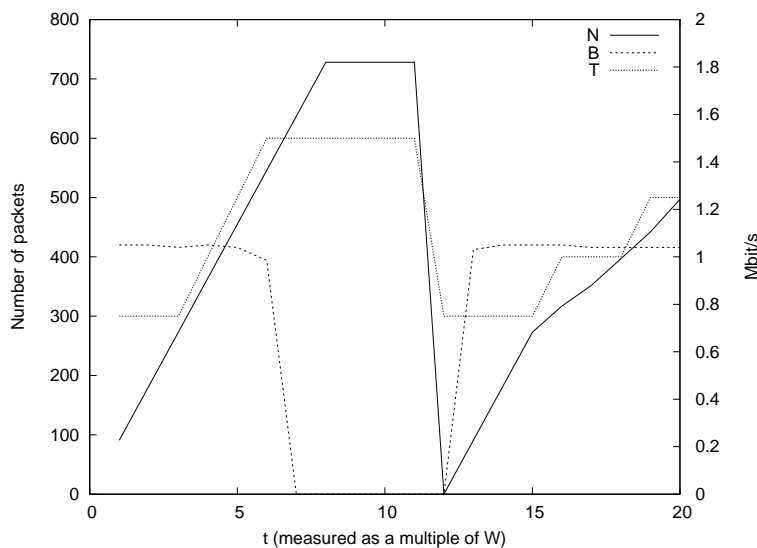


Figure 4.27: A graphical example of the application for asymmetric flow detection. The adopted policy is to drop until the number of packet in the overloaded direction decreases. The figure plots the link bandwidth used by the source, the number of sent packets and the threshold. When the threshold is reached and overcome the packets are dropped (used bandwidth goes to 0).

When the number of packets in the flow decreases, the threshold T is brought back to the starting value and packets are allowed to cross the network again.

This behavior is shown in the example of Fig. 4.27, where we plotted three curves as a function of time. The threshold T is the dotted curve that increases, reaches its maximum and then decreases again when the traffic falls back within the pre-set limits. The number of packets N sent per W is the continuous line, and goes up from 100 to more than 600. Then it drops to 0 and stays at 0 for a minute ($4W$), which is the flat section of the continuous curve. Then iperf3 starts sending packets again as before. After a minute also T is reset as shown by the dotted curve. The dashed curve in the figure is the bandwidth B used by the flow as measured by iperf3. As it can be seen, when the number of packets sent overcomes the threshold T , then B drops to 0, meaning that the packets are dropped and iperf3 cannot see any capacity available for the traffic flow.

Otherwise, when the network behaves normally, B is constant and equal to the link capacity.

The specific values of these quantities are just an example, to show how the P4 application works and interacts with the CP orchestrator; in the same way, dropping packets is just one of the possible countermeasures to be taken.

The last set of tests aimed at comparing the proposed solution with similar ones from the literature. The approach presented in [14] is the closest to ours: there, a ratio between packet

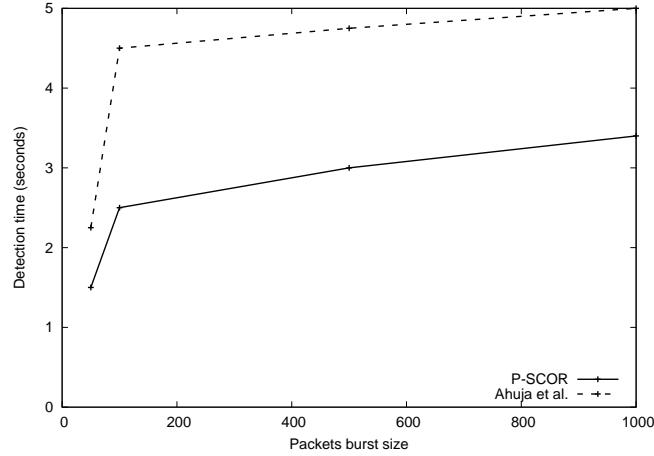


Figure 4.28: Detection time of traffic flows very asymmetric of P-SCOR, compared with the detection time reported in [14]. The threshold of detection is set to 100 packets per window.

entering and leaving the switch is computed, in order to look for an asymmetry of the traffic flow, but the authors had to implement the calculation at the controller level. To the best of our knowledge, there is no solution like the one we proposed, in which the asymmetric flow calculation is performed directly and dynamically at the data plane level.

Graph 4.28 presents a comparison between our work and the results reported in [14]. We simulated the same attack traffic with the same ratio threshold of 100 packets per second. The network topology for this test is the same used for the previous ones, i.e., three switches connected in a ring.

What we compare is the time taken to detect the traffic as malicious, before and after reaching the threshold. The graph shows that when the attack rate reaches the threshold, the detection time stabilizes. Comparing the P-SCOR solution with that proposed in [14], we can see the clear advantage coming from not having to send packets to the controller level for ratio computations.

5

Attack and testing

In this section, we focus on the offensive part of the thesis. We will describe some attacks to the aforementioned systems, and how to emulate various testbeds to demonstrate our attacks. We will mainly focus on industrial networks, introducing attacks to these kind of networks, in specific scenarios such as time synchronization between devices.

The goal of this chapter is to answer to RQ2: *RQ2: Are emerging network technologies introducing new classes of threats?*

While being appealing for the enhanced performances, new network paradigms can introduce security threats. This question was investigated by discovering and implementing attacks on technologies, an answer can be found in Chapter 5. RQ3: *Are emerging network technologies re-introducing well-known problematics that should be included in a Threat Modelling methodology?*

and RQ4: *Can test-bed, resembling the real systems to some degrees, be built to make the process of Security testing in critical scenarios more feasible?*

These answers are analyzed through the discovery of new attacks and the applications of well known vulnerabilities to new domains. The testbed part is therefore used to replicate the attacks over realistic scenarios to highlight and measure the outcomes.

5.1 Anagram Attack to password similarity systems

The system described in section 4.2 is vulnerable to an attack which exploits the order of the discovered n-grams. This attack aims to reconstruct the password as the anagram of the various n-grams. The attack is composed of four steps:

1. Generate all the hashes for a specific n-gram;
2. Hash the n-grams into a Bloom Filter;
3. Analyze the Bloom Filter and get the position of bits set to the true value;

4. Compose the various n-grams to create a password.

This scenario can be disruptive and can lead to the full disclosure of hashed password in no time. Additionally, this kind of attack can be enhanced with the help of a word dictionary similar to the one used in classical password attacks: the search tree can be pruned by excluding the words which do not contain the n-gram. For instance, let us imagine that a user inserts the password “password!!” in the filter and splitting the input in a list of bi-grams. In this work, we use a general definition of ν -grams functions, a ν -gram function will split the input in its ν -grams, where ν is the number of characters in each substring. In the case of bigrams, obviously $\nu = 2$. Accordingly, the attacker will generate all the possible bi-grams. This, using an alphabet Δ will result in a generation of $|\Delta|^2$ bi-grams which, for the ASCII case, is $(127 - 32)^2 = 9025$ bi-grams, an operation which requires at most a couple of milliseconds on any modern CPU. After this step, the attacker will hash the bi-grams inserting them into the filter, which requires $\Theta(n)$ insertions with n as the number of bi-grams. Subsequently, the attacker can create all the possible combinations in the search space generated by the pruned alphabet of bi-grams Δ_{II} . The research can be conducted by using an incremental number of repetitions. Therefore, the number of combinations which can be generated using the corresponding Bloom Filter are:

$$\binom{\Delta_\nu}{\frac{n}{\nu}} \quad (5.1)$$

with n as the length of the searched string. In the case of a common password of 8 ASCII characters and a Bloom Filter constructed with bi-grams the formula will result to:

$$\binom{9025}{\frac{8}{2}} \approx 2.76 \times 10^{14} \quad (5.2)$$

combinations. If we consider the worst case with repetitions the formula becomes

$$\binom{\Delta_\nu + \frac{n}{\nu} - 1}{\frac{n}{\nu}} \quad (5.3)$$

which, in this example, will result to $\binom{9025+4-1}{4}$ which is almost the same value as seen for the non-repetition case. Passwords can be longer than eight characters to provide enough security against brute-force or dictionary attacks. For a password varying from n characters to N the number of combinations are:

$$\sum_{i=n}^N \binom{\Delta_\nu + \frac{i}{\nu} - 1}{\frac{i}{\nu}} \quad (5.4)$$

In our data-set the average password size was 11.56, therefore, limits between 8 and 14 can be evaluated resulting in:

$$\sum_{i=8}^{14} \binom{9025 + \frac{i}{2} - 1}{\frac{i}{2}} \approx 9.96 \times 10^{23} \quad (5.5)$$

The crypto-analysis of the attack should include the details of the filter like the size of the bucket or the number of hash functions used similar to the analysis present in section 4.2.

5.2 Our Attacks on PTP

In this section, we present the main research questions and challenges, and present an high-level description of our attacks. In general, we leverage the lack of authentication of TLV frames, that allows persistent compromise of PTP infrastructure, without requiring consistent traffic manipulation.

Research Question and Challenges

In this part of work, we address the following main research question: *Does the PTP standard define sufficient security features to ensure that standard-compliant implementations are secure against manipulation?*

Answering this question is non-trivial due to the complexity of the standard, and the need to investigate practical implementations.

The main drawback of the PTP protocol is the lack of authentication between nodes, that opens the door to many kind of attacks, like altering the Master election protocol (BMCS) or the network management TLVs, as will be described.

This is in general a known issue and, as a consequence, PTPv2.1 (IEEE-1588-2019 [212]) introduces new specifications to overcome it. Two different way to secure the protocol are introduced: internal security and external security [405]. Nonetheless the proposed remediations are too specific and shallow for many cases. For instance the TLV attacks that will be presented in Section 5.2 are not directly addressed by this remediation. TLVs could be authenticated but this is not common practice because the focus is mostly placed on the information exchange directly focusing on the synchronization process.

Moreover focusing on PTPv2.0 (IEEE-1588 2008) the security of the TLVs and management part of the protocol is fully implementation-dependant; i.e., different implementation may have different vulnerability degrees (for instance *LinuxPTP* implements stricter security methods than *PTPd* by exploiting UNIX sockets).

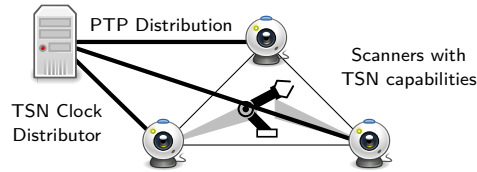


Figure 5.1: TSN with safety scanners. In this case the field of view of the scanners is synchronized using TSN/clock synchronization protocols to cover complex view (e.g., 3d) areas.

To the best of our knowledge, existing works mostly focus on attack scenarios on the synchronization part of the protocol (i.e., briefly reviewed in Section 2.3). As a consequence, countermeasures have been studied and proposed mostly in this perspective [405].

To date, vulnerabilities stemming from the TLVs have mostly been neglected. In the following, we show that not securing the TLV PDUs leads to security flaws that can easily have effects on the synchronization similar to the ones described in Section 2.3. Therefore the answer to the research question is "no", unless proper attention is given to all the components of the protocol, included management messages.

System Model

The target network implements PTP version 2.0 (IEEE-1588-2008). Using two PTP enabled devices, we created a minimal setup. These devices are interconnected by a single boundary device. The configuration in Figure 5.3 is similar to the one we are referring on. The figure presents two active clocks, an interconnection device and an attacker. Therefore we experimented our attacks in an unsecured scenario, where correctly placed network segmentation – e.g., VLANs – is bypassable or not present. In this scenario we focus on the security of TLVs, analyzing the possibility of various exploitations, with different outcomes.

There are two ways to communicate with PTP nodes: using UDP communication or Ethernet packets. We do not focus strictly on a specific communication technology. If PTP is used as a protocol at Layer 7 of the ISO/OSI model, the system could be synchronized on the internet using a network route. The downside of this approach is the conspicuous network latency compared to the Layer 2 solution. While a UDP-based solution is easier to setup, the attacker could bypass firewalls using IP spoofing attacks. In Ethernet communication, every PTP-capable node must be connected to the same LAN. In our attacker design we consider it directly connected to the PTP network. Such network can be protected by segmentation or cryptographic technologies such as VLANs or MACsec. How to bypass these defensive technologies is outside of the scope of this document. Tools and techniques to perform IP spoofing or VLAN hopping attacks are present in

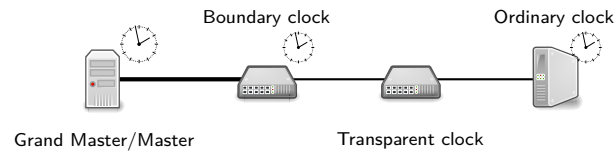


Figure 5.2: PTP hierarchy. The clock symbol specifies which nodes are synchronized each other using PTP.

literature [200, 26]. Having a direct foothold in the network can lead to worse scenarios than the one presented in this work such as total denial of service of the traffic or data breaches. These attacks usually require a lot of bandwidth [259] or vulnerable applications. We argue that these scenarios are outside of the scope of application of PTP, which is usually tied to very field-specific machines, with very little processing power and network bandwidth.

Attacker Model

Our attacker workflow is modeled after the cyber-security kill-chain, as described by Yadav et al. in [483]. Acting as described by the kill-chain model, the attacker will try to get information on the network topology and configuration; get an initial foothold in the network and alter parameters; then he/she will try to maintain the access and keep the network under control, cleaning up any trace of the attack. This model describes a very broad scenario, applicable to almost every attack; to narrow the goals of an attacker, we have identified three possible outcomes.

Disruption of safety constraints. Safety Integrity Levels (SIL) [56] are requirements classes which specify threshold values to obtain context-specific safety. For instance, medical devices have specific parameters to ensure safety. This kind of requirements can be disrupted if the time synchronization between the devices is altered. In general, devices implementing these requirements must have a procedure that handles failures. This kind of procedures could include the complete stop of the system. For instance, in the scenario introduced in Section 2.3, the robotic arm could be stopped indefinitely if the scanners are not synchronized. The attacker can therefore subvert these safety measures, transforming them into a Denial of Service attack.

Downgrade of endpoint confidentiality and authentication schemes. Public key infrastructure technologies are heavily dependent on the timing aspect of the various clocks. While the time requirements of these technologies is sensibly different from the application target of PTP, time de-synchronization can occur continuously to alter the wall-time clock of the endpoint. This will enable attacks on the validity of the certificate, enabling expired ones to be validated by the endpoint. Other protocols rely on a more strict synchronization than the one used by PKI; a typical example is the *Kerberos* protocol. Since this protocol uses timestamps extensively, if the clock is

not precisely synchronized between machines, it can be tricked into authenticating an entity using an old authenticator request. This replay attack, described by Bellare et al. in [57], can be eased if de-synchronizations of the clocks are possible.

De-synchronization of 5G edge applications. As MEC-API 011 specifications states [6], a MEC application can request time synchronization to the MEC platform. This can be made using PTP and enables the attacks to the time synchronization PDUs as well as attack to management frames. If the PTP daemon of the MEC platform is not secured, it can suffer from time de-synchronization by communicating with the application using TLVs. That is, an application can alter the time reference of the clock by interacting with the platform. If not segmented correctly, the application can spoof a valid master, altering the synchronization of other application. This alteration will make the downgrade presented in the previous point, feasible.

Implementation

As anticipated in the Introduction, the attacks here presented exploit the TLV frames included in the PTP protocol. The scope of TLVs is to enable massive, remote and automated management of the PTP network, an action which is usually not required in small and closed contexts. On the other hand, these features is key to the deployment of larger scale implementations, as those foreseen in the use cases presented in Section 2.3, as well as to more complex operations like topology discovery, etc [38].

Therefore we assume a deployment scenario where TLVs are enabled and used for the scopes mentioned above. At the best of our knowledge when this happens TLVs are not authenticated and can be exploited as an attack vector.

Along this line here we present the core idea for three different attacks and introduce the related TLVs and how they map to the (abstract) attacks. We present the PTP testbed we implemented, demonstrate the effectiveness of the proposed attacks and finally compare their efficiency with respect to prior work.

We conclude with further attack validation results, obtained from an alternative second (physical) testbed, demonstrating very similar behavior with respect to the simulated one.

Exploiting TLVs in PTP Implementations

We propose three attacks, which all exploit TLVs (Reconnaissance, Clock Disable Port, and Clock Accuracy Attacks). The TLVs of PTP have three different verbs, `GET`, `SET` and `COMMAND`. While the first two are used to change or retrieve a value of the running daemon, the third can issue commands in the configuration (e.g., enabling or disabling a network port using the `DISABLE_PORT` command

TLV	Security Problem
DISABLE_PORT	Exploitation: Denial of service
DELAY_MECHANISM	Exploitation: Denial of service
CLOCK_DESCRIPTION	Reconnaissance: Topology map
USER_DESCRIPTION	Reconnaissance: Topology map
PRIORITY1	Exploitation: BMC tampering
PRIORITY2	Exploitation: BMC tampering
CLOCK_ACCURACY	Exploitation: BMC tampering

Table 5.1: Exploited TLVs, details on attacks possible with these TLVs are presented in Section 5.2.

TLV). Our proposed attacks target those TLVs, as they are most relevant for the security of the network.

The specific commands/TLVs we are attacking are summarized in Table 5.1. We provide additional details on attack implementation and results in Section 2.3.

- *Reconnaissance Attacks* leverage commands such as `CLOCK_DESCRIPTION` and `USER_DESCRIPTION` for reconnaissance attacks, aiming to map the topology of the PTP enabled network. The `CLOCK_DESCRIPTION` TLV can be the first reconnaissance available to the attacker, if not correctly limited to specific users (such as the tactic used by LinuxPTP) or not correctly confined in the network;
- *Clock Disable Port Attacks* disrupt synchronization between nodes using enable or disable network ports with `ENABLE_PORT` and `DISABLE_PORT` commands;
- *Clock Accuracy Attacks* change the parameters used by the Best Master Clock Selection algorithm using `PRIORITY` related TLVs.

In addition to the TLVs in Table 5.1, there are others TLV which can be exploited for security advantages: these were not used in this work but were analysed and a taxonomy is presented in Appendix B.

The relevance and the possible risks of a mis-use of some TLVs is not a new issue, since it is also highlighted in the notes of the PTP Standard documents. For instance, an extract from the IEEE-1588-2008 document [211] specifies a note on the `CLOCK_ACCURACY` TLV:

The accuracy and the time in the grandmaster clock is normally determined by interacting with a primary or application-specific time source, e.g., GPS, by means outside the scope of this standard. If the time is set in the grandmaster by means of the `TIME` TLV, then the accuracy should also be set. Since the `clockAccuracy` attribute is considered in

the operation of the BMC algorithm, the setting of the `clockAccuracy` attribute in any clock by means of this TLV can result in a change of grandmaster the next time the BMC algorithm is performed.

Nonetheless this critical role of such TLVs is not directly associated with a security threat, in spite of the fact that the quoted TLV can be used to control the BMC algorithm outcome, resulting in election of different grandmaster and master clocks.

Virtualized Testbed Design and Implementation

To demonstrate the attacks proposed in Section 5.2, a practical testbed is required. We now summarize the testbed which we designed and implemented for this purpose. We note that, in addition to demonstrating our new attacks, the testbed is also able to reproduce attacks from prior work, such as byzantine master. After acceptance of this work, we plan to release the testbed framework as open source for others to replicate.

In general, the virtualized testbed leverages virtualization that allows us to run actual PTP programs on guests connected by virtual networks. This setup allows for a portable and self-contained specification of attacks, and allows to easily replicate results. We also tested our attacks in a second physical testbed, which is introduced in Section 5.2.

The architecture of our virtualized testbed can be seen in Figure 5.3 and Figure 5.6. The configuration relies on LinuxPTP, PTPD, and PTPv2 software time-stamping. The testbed can show the results of clock alteration on the target system, accelerating or slowing the clock difference between nodes. We configured the PTP hosts to enable configuration via TLVs, to replicate a setting in which TLVs are in general used for remote PTP management. Initially, TLVs are correctly configured (e.g., `priority1` and `priority2` in LinuxPTP), but can be reconfigured remotely.

The testbed is implemented as a virtualized infrastructure using Vagrant¹ and Ansible² to automate the tests. All the nodes of the infrastructures run the same operating system: Ubuntu-18.04.

The nodes are virtualized as VirtualBox guests equipped with E1000 GigE virtual NICs. In our setup, the system was tested using an Intel Core i7-8700 CPU, with a nominal clock frequency of 3.20GHz. The Hypervisor does not offer support for PTP or TSN-enabled virtual NICs. To overcome this problem we configured software time stamping in the PTP daemons. Therefore, we connected the virtual machines with the default hypervisor network support (NatNetwork, in the terminology of VirtualBox). We implemented PTP synchronization in virtual testbed with

¹A tool to build portable development environment in form of virtual machines or containers: <https://vagrantup.com>

²A configuration toolkit to build and configure environments using SSH: <https://ansible.com>

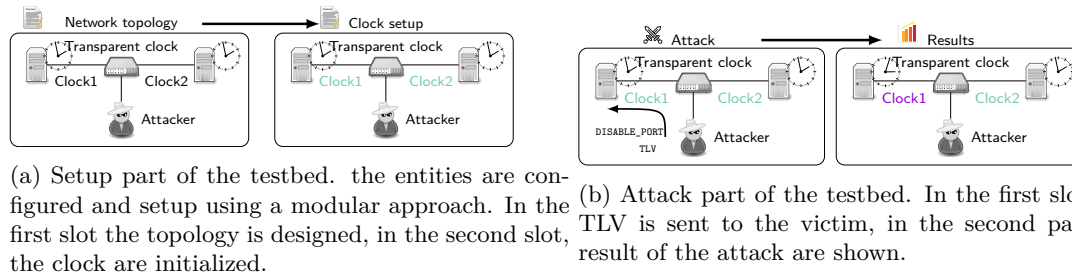


Figure 5.3: Testbed design, all entities in the figure are optional and configurable. The system is built with IaC technologies and it can be reconfigured to include many clocks, attacker or different topologies. In black, entities that are not part of the time synchronization platform; in green entity mutually synchronized; in red de-synchronized entities.

software timestamps using UDP, therefore it is completely transparent to layer 2 topology and protocols employed.

The workflow of using the virtualized testbed is as follows:

1. we define a topology using configuration files, in this case a topology with two legitimate entities, a network switch and an attacker;
2. we chose the nodes on which to instantiate a PTP clock, in this case a PTPd daemon in the two servers, which became Ordinary Clocks,
3. the other entities are left as passive elements and the switch became a Transparent Clock and not a Boundary Clock;
4. the attacker is issuing the TLVs to execute the attack;
5. the various nodes are observed to highlight clocks behavior.

Software Tools

To support the attacks operations three different software tools were developed a modular software tool that we are about to publish as open-source. The tool is named *PEF* that stands for PTP Exploitation Framework.

The main modules of PEF are:

1. *TLV forging module* baseline to forge and send TLV frames;
2. *reconnaissance module* used to investigate the topology and active functionality of the PTP network;

3. *indirect attack module* used to modify the PTP network behavior by means of indirect misconfiguration of nodes, meaning that a node (the attacker) will modify the behavior another node by sending a specific TLV to a third node;
4. *direct attack module* used to directly interact and modify the PTP network behavior by means of specific TLVs, sent to the target by another node (the attacker).

The PEF tools was implemented for GNU/Linux systems in Python 3 and interfacing it with C APIs and reference tools such as `ptp41`.

Attack Implementation and Evaluation

In this section we discuss the implementation of the three attack strategies, as explained in Section 5.2, and quantitatively demonstrate their effectiveness.

Reconnaissance We implemented the Reconnaissance attack as exploitation module for PEF. The module starts by analysing the network to identify hosts that are the running PTP daemons, leveraging TLV-based queries. The queries allow to scan the network for PTP hosts and retrieve the version of the running daemon.

In particular, our queries use `CLOCK_DESCRIPTION` management TLVs. Sending this TLV to all the hosts (e.g. using a multicast or broadcast address) will cause all TLV-capable hosts to reply to the scanning hosts. The results are then aggregated by PEF, and presented to the user as a list of all PTP-capable nodes, and meta-data for that host such as the name of the network clock, the specific PTP daemon used, and its version number. In addition, the information on each host contains details on the BMC algorithm properties of a node, therefore the attacker can get a clear view of the topology and capabilities of the network. This behaviour is pictured in Figure 5.4

After this phase, we analyzed the list of running daemons. If a daemon is known to be vulnerable, it can be exploited with some attacks, leading to the problems described in Section 5.2.

Clock Disable Port Attack This attack exploits the *direct attack module* of PEF. The attacker will misconfigure the target by sending directly a TLV that will then result in a synchronization failure.

The general steps of the attacks are described in Figure 5.3. The attack starts after the systems is running for thirty minutes of wall-clock synchronization using PTP – visible in purple at the bottom left corner.

```

[root@haigha pef]# ./pef.py
pef> interface enp0s25 type 2
pef> recon
Found Host: BridgeNode#1
clockType 32768
physicalLayerProtocol IEEE802.3
physicalAddress 6c:b3:11:1c:92:a9
protocolAddress 3 6c:b3:11:1c:92:a9
manufacturerId 00:00:00
productDescription PEFLinuxPTP;2.0;1
revisionData 2.0;2.0;2.0
userDescription BridgeNode#1
profileId 00:1b:19:00:01:00

Found Host: LeafNode#1
clockType 32768
physicalLayerProtocol IEEE802.3
physicalAddress 6c:b3:11:1d:b9:23
protocolAddress 3 6c:b3:11:1d:b9:23
manufacturerId 00:00:00
productDescription PEFLinuxPTP;2.0;1
revisionData 2.0;2.0;2.0
userDescription LeafNode#1
profileId 00:1b:19:00:01:00

```

Figure 5.4: PTP Exploitation Framework demonstration, the reconnaissance module displays the information got from the running PTP daemons.

- The first two steps (Figure 5.3a) show the topology and the start up of the scenario with the two synchronized hosts and the attacker. The latter does not require to be synchronized with the hosts.
- A `DISABLE_PORT` TLV is sent to the victim, and the PTP port is disabled for two minutes (Figure 5.3b).
- In this short time window, pictured as the two red bars in the figure, the daemon port is disabled, without any synchronization capability.

After this time frame we stopped the attack and the drift is measured³, with results summarized in Figure 5.5. The PTP leaves continues to drift from the reference master. If the drift reaches values higher than the maximum correction values specified in the configuration of the daemons, the clocks could continue to drift indefinitely even after a `ENABLE_PORT` command is issued on the victim, without any re-synchronization from the daemons. This is shown by the green curve in the rightmost part of the figure.

³We disabled the time synchronization by re-instantiating the daemon in dry-run mode to measure the drift between the clocks.

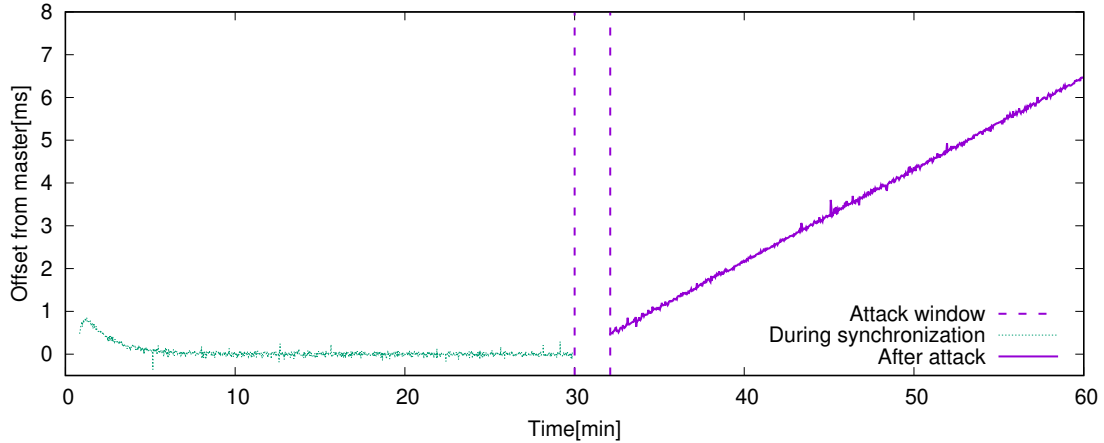


Figure 5.5: Results of a clock de-synchronization attack in the virtualized testbed. In this attack we have exploited a `DISABLE_PORT` TLV. The clock then proceed to drift from the reference master, during the attack we keep the port disabled and then proceed to re-enable it after one minute.

Our attack successfully introduced a significant clock drift after few minutes, continuing and eventually reaching a value of 8ms after thirty minutes. As a typical PTP synchronization is precise within $10ns$ to $100ns$ [308], we argue that the drift of $\approx 4\mu s/s$ introduced by our attack is significant.

Clock Accuracy Attack This attack exploits the *indirect attack module* of PEF. The attacker will send a TLV that cause the target to choose a wrong reference clock and therefore lock on a wrong synchronization.

The concept of this attack is show in Figure 5.6.

- In a first phase, referred as the *learning* phase, we synchronized the attacker node with the leaf nodes. This phase is required because the synchronization of PTP leaves can have some constraints like the maximum amount of μs which can be corrected. To speed up the synchronization of the victims we synchronized with the legitimate master. As, for now, there is no authorization mechanism over the synchronization of new leaves, this operation is completely transparent to the clock master.
- After the learning phase, we spoofed and taken over the master-clock role. This can be made in various ways, in our proposal by a TLV announcing a clock with more priority.
 - We send the `CLOCK_ACCURACY` TLV.

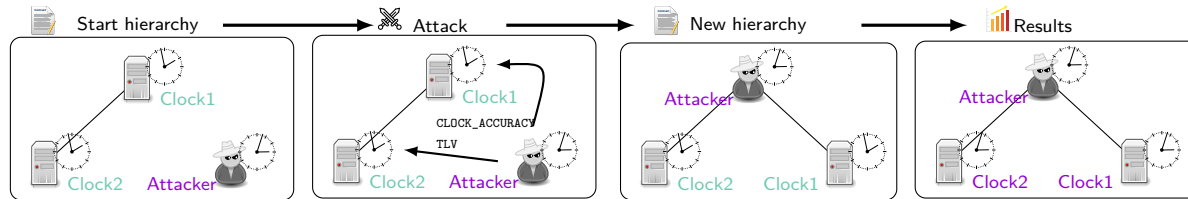


Figure 5.6: Hierarchy-related attack: In the testbed the attacker issue a `CLOCK_ACCURACY TLV`. This TLV will poison the hierarchy management algorithm, when an election occurs, the attacker will become the Master / Grand-Master of the system. The synchronized clocks not controlled by the attacker are represented in green, in red the ones controlled by the attacker in every step.

- We change the declared `ACCURACY` of the clocks in order to alter the outcome of the next Best Master Clock Election (BMC). Also, we set our node exposed accuracy to a value higher than the one set into the victims.
- The new hierarchy is then pictured. When a BMC protocol run is triggered (after a time configured in the various daemons or after a loss of connectivity between clock1 and clock2), we will win this election.

- After this spoofing phase, we had completely control of the Ordinary clocks' reference.

The final result is shown in Figure 5.7 where the clock is synchronized with the attacker once and then it is continuously altered by bringing the system clock back by a second every ten seconds. In terms of accuracy, this attack is even more efficient than the previous one. The clocks of the system are kept back from flowing normally.

Comparison to Prior Work Attacks

To implement each of our attacks, the attacker just needs to learn the network topology and send few management frames (as low as one frame for the Disable Port Attack). As a consequence, the bandwidth requirements for our attacks are near to zero, enabling this attack even from embedded network platforms such as low powered micro controllers⁴.

Compared to attack strategies to PTP already presented in the literature, such as for instance [452, 321], our attacks obtain similar effects at less bandwidth cost. As mentioned in the introduction these attacks require a persistency in the attack phase, either based on denial of service for bandwidth exhaustion or on network addressing poisoning, that is not needed in the cases presented in this work.

⁴TLV size is, on average, under a few hundreds of bytes. For instance, `DISABLE_PORT` TLVs have 44 bytes of PDU header, 4 Bytes of Type and Length fields and an empty Value.

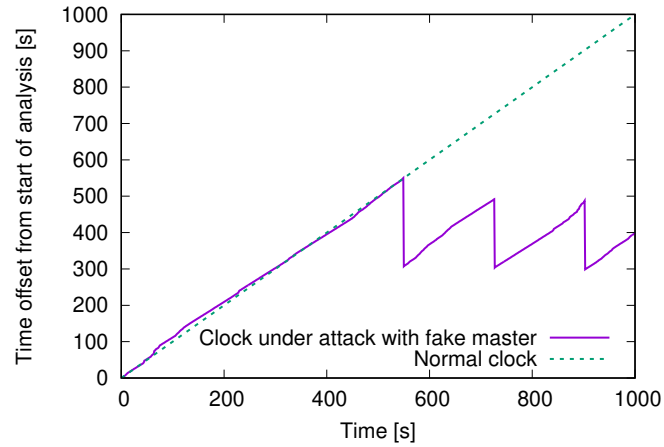


Figure 5.7: When the attacker is elected as the master, the other clocks in the system are completely controllable. This representation of the clock is a query of the internal clock of a victim and compared with the internal clock value of a previously synchronized external observer.

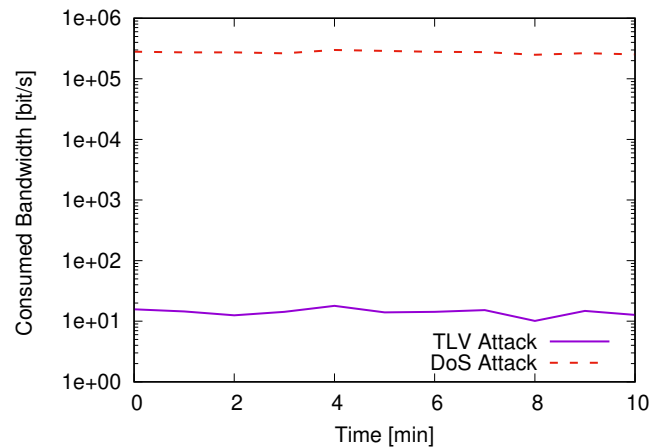


Figure 5.8: Comparison between Bandwidth-requirements of a Denial of Service (DOS) attack and normal PTP operations (PTP). The data are a result of an average between five runs of 10 minutes and represented using logarithmic scale. The Denial of Service attack was done using hping3.

This is confirmed by the Plot 5.8, a comparison with a Bandwidth Denial of Service and Normal PTP operations. A bandwidth attack would be extremely more detectable by Bandwidth-related alarms, to which our attack would be totally transparent.

Alternative Physical Testbed

The results presented in the previous sections were obtained, as explained, into a laboratory environment based on a virtualized infrastructure. A further question we wanted to answer as part of this work is whether our virtual environment can be considered a realistic implementation of real life PTP network based on physical devices. To answer this question we implemented a second testbed with the same topology but using physical hardware.

The physical testbed is based on a network constructed with three GNU/Linux physical devices. These hardware devices mounts NICs capable of time-stamping conforming to IEEE-1588-2008 standard. We equipped two leaf devices with Intel 82579LM GigE controllers. We interconnected these devices by a third node, equipped with a double-port Intel 82576 GigE NIC, that can act as boundary/transparent clock. We used LinuxPTP to test the vulnerability. While in the virtual case the LinuxPTP daemon is directly connected to the system clock, in the physical case the system clock is synchronized using the Linux kernel physical hardware clock (PHC) API.

Aruba's switches A proprietary implementation of PTP can be found on Aruba Switches. We have tested the implementation of IEEE-1588-2008 of an Aruba switch, model 2930M ⁵. Unfortunately, this switch series implements only Transparent Clocks. Therefore, the switches, only correct the in-transit PDUs, not analyzing them in depth. This behaviour is totally transparent to our attacks.

Open source and free software daemons

NetLeap ⁶ and RELYUM (RELY-TSN-BRIDGE) can be used to create TSN-enabled switches and network nodes, then used by companies to develop and integrate TSN in industrial products.

These products mainly use hybrid systems, employing Microprocessors running Linux kernel and FPGA to manage the communication. While the synchronization PDUs are managed by FPGA, in dedicated IPcores, the TLV and configuration part is done by an userspace daemon which is usually PTPd2 or LinuxPTP. That is very similar to what the Intel NICs do, on which our physical testbed is based on. Furthermore, this pattern can be found in other vendors such as Xilinx [3] or Altera. Quoting Altera 1588 Reference Design [1]:

⁵<https://www.arubanetworks.com/products/switches/access/2930m-series/>

⁶<https://novtech.com/products/netleap.html>

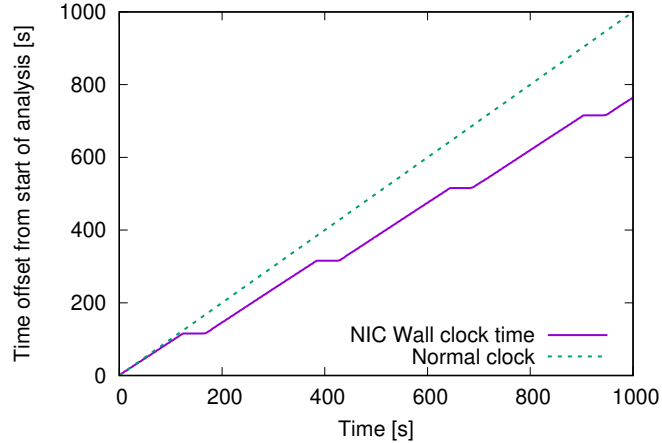


Figure 5.9: Clock accuracy attack in the physical testbed. In this scenario, a slightly modified version of the previously presented attack is pictured. Over a time span of thirty minutes, the clock is taken back by the spoofed master by 10 seconds every 60 seconds.

The objectives of the reference design include the following: 6.99ns timestamp accuracy using Altera 1588 hardware IP and Linux PTP software stack for 10Gbps speed.

At the moment of writing, no stable implementation of IEEE-1588-2019 is present on the market. For these reasons we implemented the main Free and Open Source Software (FOSS) solutions using LinuxPTP and PTPd (version 1 and 2), interchangeably.

Results In these infrastructures we executed the same attacks as already explained in the previous section with almost identical results. Just as an example Figure 5.9 shows what happens when a clock accuracy attack is executed in the physical testbed. This attack is a little less intrusive than the one against virtual infrastructure, due to physical constraints and implementation of interface. As evinctable from the plot, the wall clock time of the interfaces is left flowing for sixty seconds, and then a jump of 10 seconds occurs. In this time lapse, the victim’s clock is kept stall and the reference to the UNIX epoch is frozen. This will give us an advantage on the target clock, gaining a relative control of the target wall clock time. We concluded that the results obtained in the software implementation of the test-bed are realistic and basically executable also in physical implementations.

Part III

Conclusions, Appendices, and References



6

Conclusions

In this Thesis, we described the security of emerging networks and infrastructures.

We designed the Thesis and the research focusing on three aspects that are fundamental for security: Analysis, Attacks, and Defenses. These three aspects were analyzed for modern technologies such as Software Defined Networking (SDN), Network Function Virtualization (NFV), Industrial Networks with clock distribution, and Microservices.

The various cores of this Thesis focus on the analysis of the literature attacks and defenses and their distribution (in the Analysis chapter), the showcase of novel attack techniques (Attacks), and the analysis of how the aforementioned technologies can improve the overall protection of the network or of the systems (Defenses).

This approach answered to the following research questions presented in Chapter 1.

RQ1 *Which are the emerging network technologies that can increase the protection part of the Security of a system in proactive or reactive ways?*

As demonstrated by TecNetium and P-SCOR, network programmability can increase the overall security of the system, introducing new monitoring and hijacking capabilities. This approach can add more flexibility to the system, therefore giving to the defenders more time to react to the attacks.

RQ2 *Are emerging network technologies introducing new classes of threats?*

To answer this question, the approach of this Thesis is to present new attacks to new protocols. In Chapter 5, new attacks to novel systems and to well-known protocols are presented with their relative impact factors.

RQ3 *Are emerging network technologies re-introducing well-known problematics that should be included in a Threat Modelling methodology?*

As RQ2, the answer to this question is presented in Chapter 5, old attacks to PTP are presented and described. Also, in Chapter 3, the reintroduction of spoofing and denial-of-service is addressed in the field of Microservices.

RQ4 *Can test-bed, resembling the real systems to some degrees, be built to make the process of Security testing in critical scenarios more feasible?*

Testbed can be employed to emulate and isolate the vulnerable component of the infrastructure. This approach can be used as demonstrated for PTP in Chapter 5, where the protocol is emulated and the critical part can be easily pinpointed.

We believe that analysis, attack and defenses fields should ideally be completely balanced. While this thesis is slightly unbalanced on the defense side, we believe that the attacks we described can be expanded to various field bus or network technologies. That holds also for the defense and analysis part of the thesis. We believe that these research directions can be expanded to cover a plethora of novel possibilities in research as well as in industrial applications.

Appendices



A

TSN protocols

The IEEE specification for time sensitive networking is based on IEEE Tag VLAN (802.1Q). This make Time-Sensitive Networking (TSN) an extension of this protocol, adding time-based enhancements to this one. IEEE TSN is composed of the following protocols:

- 802.1Q - VLAN tagging; This protocol assigns to every L2 packet a tag which will be used to treat the traffic as flowing through a different LAN.
- 802.1AS - Timing and synchronization; which specifies the Generalized Precision Time Protocol (gPTP).
- 802.1CB - Frame replication; which specifies Parallel Redundancy Protocol (PRP) for the network. The traffic can be replicated to have a strong availability of the network and, at the same time, discover network link problems.
- 802.1Qat - Stream Reservation protocol; nodes can announce and retrieve network capacity. This can be used to reserve paths for a specific bandwidth.
- 802.1Qbu & 802.1Qbv - Preemption and Scheduled Traffic. This standards declare the inner scheduler workflow, declaring how traffic in TSN can coexist with standard traffic.
- 802.1Qca - Path control; path in TSN can be chosen using different algorithms, IS-IS, Constraint based routing or explicit routing.
- 802.1Qcc - TSN Configuration; the configuration of the TSN networks can be made using a single Centralized system configuration which will use configuration protocols like NETCONF to configure the network equipments.
- 802.1Qch - Cyclic queuing and forwarding. This standard specifies cyclical buffer with coordinate clocks.

- 802.1Qci - Per stream filtering. This standard specifies the security policies to apply to TSN switches. This can be viewed as a firewall on the TSN networking.
- 802.1Qcr - Asynchronous shaping; introduces a non-global time based scheduler algorithm based on slice allocation providing determinism.

B Taxonomy of PTP TLVs for security

TLVs can have different level of security problematics, in this appendix the TLVs are grouped by the attack which them can lead to. We tested different implementations of PTP daemons, namely `LinuxPTP` and `PTPd`, that seems to be the main implementation of the protocol under Linux and BSD systems. The implementation that was used in the security testbed is present in every table in column *PTP implementation*. The score of the attack is calculated using the CVSS scoring system¹. In this system we classify the different score based on the following table:

AC Complexity of the attack **H** → high, **M** → medium, **L** → low.

C Confidentiality is harmed: **N** → none, **P** → partial, **C** → complete.

I Integrity is harmed: **N** → none, **P** → partial, **C** → complete.

A Availability is harmed: **N** → none, **P** → partial, **C** → complete.

TLVs that seem unexploitable.

We have not found any security problem from the use the TLVs presented in Table B.1.

The attacker model will follow different phases, nominally:

1. Reconnaissance;
2. Intrusion and delivery of the attack;
3. Exploitation of the vulnerability;
4. Persistence of the attack;
5. Cleanup of the traces.

¹<https://nvd.nist.gov/vuln-metrics/cvss> visited 2020-06-04

TLV name	Scope	Simulated with PTP testbed	PTP Implementation	Score
NULL_MANAGEMENT	-	✓	LinuxPTP, PTPd	-
ALTERNATE_TIME_OFFSET_ENABLE	-			-
ALTERNATE_TIME_OFFSET_NAME	-			-
ALTERNATE_TIME_OFFSET_MAX_KEY	-			-
ALTERNATE_TIME_OFFSET_PROPERTIES	-			-
TRANSPARENT_CLOCK_DEFAULT_DATA_SET	-			-
TRANSPARENT_CLOCK_PORT_DATA_SET	-			-
PRIMARY_DOMAIN	-			-
LOG_MIN_PDELAY_REQ_INTERVAL	-			-
TIME_PROPERTIES_DATA_SET	-			-
TRACEABILITY_PROPERTIES	-			-
UNICAST_MASTER_MAX_TABLE_SIZE	-			-
ACCEPTABLE_MASTER_MAX_TABLE_SIZE	-			-
ENABLE_PORT	-	✓		-

Table B.1: PTP IEEE-1588 2008 TLVs that seems to be unexploitable.

Mimicking this model, we can separate the various unsecure TLVs in the following clusters. These TLVs can imply security in different scopes, these scopes include:

Network General PTP network domain;

Time Time synchronization;

BMCS Best Master Clock Selection (BMCS) algorithm;

Log Log management;

Extension PTP-2008 standard extensions.

TLVs that can be used for Reconnaissance.

The reconnaissance phase is the first foothold in the network, it analyzes the network to search for the exploitable vulnerabilities.

In Table B.2, is presented a list of TLVs that can be used to scan the network or gain sensible information that can be used in subsequent phases.

TLVs that can be used to exploit the network.

In the exploitation phase, the attacker will exploit the discovered vulnerabilities and try to maximize

TLV name	Scope	Simulated with PTP testbed	PTP implementation used	Score
CLOCK_DESCRIPTION	Network	✓	LinuxPTP, PTPd	AC:L/C:N/I:N/A:N
USER_DESCRIPTION	Network	✓	LinuxPTP, PTPd	AC:L/C:N/I:N/A:N
FAULT_LOG	Log			AC:M/C:N/I:N/A:N
DEFAULT_DATA_SET	BMCS	✓	PTPd	AC:L/C:N/I:N/A:N
CURRENT_DATA_SET	BMCS	✓	PTPd	AC:L/C:N/I:N/A:N
PARENT_DATA_SET	BMCS	✓	PTPd	AC:L/C:N/I:N/A:N
PORT_DATA_SET	BMCS	✓	PTPd	AC:L/C:N/I:N/A:N
PATH_TRACE_LIST	Extension			AC:L/C:N/I:N/A:N
PATH_TRACE_ENABLE	Extension			AC:L/C:N/I:P/A:N

Table B.2: PTP IEEE-1588 2008 TLVs that can be used in the Reconnaissance phase.

the damage of his attack. The TLVs in Table B.3 can be employed to do so:

TLVs that can be used to tamper the network.

After the exploitation phase, the attacker can try to maintain the access to the system. To do so a set of TLVs, listed in Table B.4, can be employed.

TLVs that can be used to cleanup traces of attack.

TLVs can also be employed to cleanup the trace of an attack, e.g., to hide the presence of a trojan horse in the network. This can be the first step in the track covering, making difficult for an incident response team to reconstruct the dynamics of the attack. In Table B.5, is presented a set of TLVs of the PTP standard that can be used to do so.

TLV name	Scope	Simulated with PTP testbed	PTP Implementation	Score
INITIALIZE	BMCS			AC:M/C:N/I:N/A:C
UTC_PROPERTIES	Network,Time	✓		AC:L/C:N/I:N/A:C
VERSION_NUMBER	Network,Time	✓		AC:L/C:N/I:N/A:C
LOG_SYNC_INTERVAL	Network,Time			AC:L/C:N/I:P/A:C
ANNOUNCE_RECEIPT_TIMEOUT	Network,Time			AC:L/C:N/I:N/A:P
LOG_ANNOUNCE_INTERVAL	Network,Time			AC:L/C:N/I:N/A:P
DOMAIN	Network,Time	✓		AC:L/C:N/I:N/A:P
SLAVE_ONLY	Network,Time			AC:L/C:N/I:N/A:P
DISABLE_PORT	Network	✓		AC:L/C:N/I:N/A:C
TIME	Time			AC:L/C:N/I:N/A:C
CLOCK_ACCURACY	BMCS	✓		AC:L/C:N/I:P/A:P
PRIORITY1	BMCS	✓		AC:L/C:N/I:P/A:P
PRIORITY2	BMCS	✓		AC:L/C:N/I:P/A:P
DELAY_MECHANISM	Network	✓		AC:L/C:N/I:P/A:C
TIMESCALE_PROPERTIES	Time			AC:L/C:N/I:P/A:P
UNICAST_NEGOTIATION_ENABLE	Network			AC:L/C:N/I:P/A:P
GRANDMASTER_CLUSTER_TABLE	Network			AC:L/C:N/I:C/A:P
UNICAST_MASTER_TABLE	Network			AC:L/C:N/I:C/A:P
ACCEPTABLE_MASTER_TABLE	BMCS			AC:L/C:N/I:P/A:P
ACCEPTABLE_MASTER_TABLE_ENABLE	BMCS			AC:L/C:N/I:P/A:P
ALTERNATE_MASTER	Network			AC:L/C:N/I:P/A:P

Table B.3: PTP IEEE-1588 2008 TLVs that can be used to exploit the network.

TLV name	Scope	Simulated with PTP testbed	PTP Implementation	Score
SAVE_IN_NON_VOLATILE_STORAGE	BMCS,Time			AC:M/C:N/I:P/A:N
RESET_NON_VOLATILE_STORAGE	BMCS,Time			AC:M/C:N/I:P/A:N

Table B.4: PTP IEEE-1588 2008 TLVs that can be used to tamper the network.

TLV name	Scope	Simulated with PTP testbed	PTP Implementation	Score
FAULT_LOG_RESET	Log			AC:L/C:N/I:P/A:N

Table B.5: PTP IEEE-1588 2008 TLVs that can be used to cover tracks of a PTP attack.

C MicroServices Dataset and R.Q.

We partition the dataset into four tables, each representing the categorisation described in subsection “Qualitative results” of section “Results of the survey” section in the paper — i) Theoretical, ii) Applicative, and iii) Theoretical and Applicative publications and iv) Survey. For each table we have 5 columns. The first 4 columns from the left (after the column containing the reference (“Ref.”) to the publication from the publications dataset) and grouped under the column group “Group” report the 4 Research Questions Groups as defined in the “Research questions” section of the paper. The value shown indicates the amount of questions of each group the publications answered. The last column labeled “Q.Num.” presents the number of questions having a positive answer.

Survey Publications

Ref.	Group				Q. Num.
	G1	G2	G3	G4	
[424]	3	2	2	1	2,4,5,8,11,13,14,16
[95]	0	0	1	0	13
[471]	0	1	1	1	11,15,16
[47]	0	0	2	0	13,14
[13]	0	1	1	0	8,13
[137]	0	0	1	1	13,16
[55]	0	0	2	0	13,14
[299]	2	0	2	0	2,4,13,14
[368]	1	0	0	0	2
[296]	0	2	1	1	8,11,13,17
[290]	2	0	1	0	3,4,13
[355]	1	0	0	0	2
[93]	0	0	1	1	13,16
[413]	1	1	2	3	4,8,13,14,16-18
[24]	1	0	1	0	2,13
[494]	0	1	1	0	11,13
[446]	0	0	1	0	13
[8]	1	1	2	0	3,8,13,14
[276]	0	1	1	0	11,13
[316]	0	1	1	1	11,13,17
[343]	0	0	1	0	13
[375]	0	1	1	0	11,13
[488]	1	1	1	0	3,11,13
[495]	1	1	2	1	2,8,13,14,16
[94]	2	1	1	0	2,5,11,13
[361]	0	0	0	1	17
[135]	2	0	0	2	4,5,16,17
[215]	3	1	0	0	2,4,5,8
[454]	2	2	0	0	2,5,9,11
[55]	0	0	2	0	13,14
[299]	2	0	2	0	2,4,13,14
[368]	1	0	0	0	2
[296]	0	2	1	1	8,11,13,17
[290]	2	0	1	0	3,4,13
[355]	1	0	0	0	2
[93]	0	0	1	1	13,16
[413]	1	1	2	3	4,8,13,14,16,17,18
[24]	1	0	1	0	2,13
[494]	0	1	1	0	11,13
[424]	3	2	2	1	2,4,5,8,11,13,14,16
[13]	0	1	1	0	8,13
[446]	0	0	1	0	13
[95]	0	0	1	0	13
[12]	2	7	3	3	2,3,6-18
[8]	1	1	2	0	3,8,13,14
[276]	0	1	1	0	11,13
[316]	0	1	1	1	11,13,17
[334]	0	0	0	0	
[343]	0	0	1	0	13
[375]	0	1	1	0	11,13
[488]	1	1	1	0	3,11,13
[495]	1	1	2	1	2,8,13,14,16
[94]	2	1	1	0	2,5,11,13
[361]	0	0	0	1	17
[135]	2	0	0	2	4,5,16,17
[471]	0	1	1	1	11,15,16
[215]	3	1	0	0	2,4,5,8
[454]	2	2	0	0	2,5,9,11
[277]	2	1	0	3	3,4,11,16,17,20
[23]	4	0	0	0	2,3,4,5
[124]	2	1	1	2	2,3,11,13,16,19
[7]	2	0	0	0	2,5
[129]	3	0	0	0	2,4,5
[313]	2	0	0	1	2,4,18
[468]	2	1	2	4	2,4,11,13,14,16-19
[310]	1	1	1	2	2,11,13,16,17
[336]	1	1	0	0	2,11
[320]	2	0	0	0	2,4
[121]	4	2	0	0	2,3,4,5,6,7
[313]	2	0	0	1	2,4,18
[378]	2	1	3	4	2,3,11,13-19
[476]	1	2	0	0	2,6,11

Applicative Publications

Ref.	Group				Q. Num.
	G1	G2	G3	G4	
[171]	2	0	1	0	2,5,13
[438]	1	1	1	0	5,8,13
[107]	0	1	1	1	11,13,17
[319]	2	2	1	0	3,5,8,11,13
[157]	2	3	2	1	2,3,6-8,13,14,16
[227]	1	0	2	0	2,13,14
[359]	0	0	1	1	13,17
[351]	1	0	2	0	3,13,14
[408]	0	0	1	0	13
[146]	1	1	1	0	3,8,13
[231]	1	1	2	1	2,8,13,14,16
[328]	1	1	1	0	2,8,13
[338]	0	2	1	0	8,11,13
[391]	0	0	2	1	13,14,16
[416]	3	1	0	0	2,3,5,7
[319]	2	2	1	0	3,5,8,11,13
[157]	2	3	2	1	2,3,6,7,8,13,14,16
[227]	1	0	2	0	2,13,14
[359]	0	0	1	1	13,17
[351]	1	0	2	0	3,13,14
[408]	0	0	1	0	13
[95]	0	0	1	0	13
[146]	1	1	1	0	3,8,13
[231]	1	1	2	1	2,8,13,14,16
[328]	1	1	1	0	2,8,13
[338]	0	2	1	0	8,11,13
[391]	0	0	2	1	13,14,16
[416]	3	1	0	0	2,3,5,7
[356]	1	1	1	0	2,11,13
[479]	0	1	1	1	11,13,16
[79]	1	2	1	0	2,6,11,13
[292]	1	1	0	1	3,7,16
[344]	1	1	0	1	3,7,16
[102]	2	1	0	0	2,3,11
[508]	2	1	1	0	2,3,11,13
[286]	1	3	1	2	2,6,7,11,13,16,19
[175]	3	1	0	0	2,3,4,11
[173]	1	1	0	0	2,11
[501]	1	1	0	0	2,11
[196]	3	1	0	0	2,3,4,11
[164]	3	3	1	0	2,3,4,6,7,11,13
[421]	2	0	0	1	2,4,18
[199]	1	0	1	0	2,13
[230]	2	1	1	0	2,4,11,13
[30]	2	1	0	0	2,4,11
[384]	3	1	1	0	2,3,4,11,13
[78]	2	4	1	2	2,3,6-8,11,13,16,18
[223]	2	1	1	0	2,4,11,13

Theoretical Publications (1/3)

Ref.	Group				Q. Num.
	G1	G2	G3	G4	
[407]	3	1	1	0	2,3,4,11,13
[142]	4	1	1	0	2,3,4,5,11,13
[159]	3	3	1	0	2,3,4,6,7,11,13
[160]	3	3	0	0	2,3,4,6,7,11
[69]	2	0	0	0	2,4
[118]	3	1	0	0	2,3,4,11
[214]	3	1	1	0	2,3,4,11,13
[132]	3	1	1	0	2,3,4,11,13
[82]	1	0	0	0	2
[176]	2	0	0	2	2,4,16,17
[266]	2	1	1	2	2,4,11,13,16,17
[294]	3	1	0	1	2,3,4,9,16
[114]	0	1	1	3	11,13,16,17,18
[155]	1	0	0	2	2,18,19
[377]	1	0	0	0	2
[432]	0	1	1	0	11,13
[389]	2	1	0	0	2,4,11
[125]	0	1	1	1	11,13,16
[27]	0	1	1	3	11,13,16,17,19
[235]	0	0	1	2	13,16,19
[379]	2	1	0	0	2,3,11
[119]	2	1	0	0	2,3,11
[255]	3	1	0	0	2,3,4,11
[242]	1	1	1	3	2,11,13,16,17,18
[131]	1	1	1	2	2,11,13,16,17
[61]	0	1	1	1	11,13,16
[138]	2	1	0	1	2,4,11,16
[318]	2	1	1	2	2,4,11,13,16,17
[274]	2	1	1	0	2,3,11,13

Theoretical and Applicative Publications					
Ref.	Group				Q. Num.
	G1	G2	G3	G4	
[12]	2	7	3	3	2,3,6-18
[164]	0	0	2	0	13,14
[139]	2	1	1	0	4,5,11,13
[194]	3	1	1	0	2,3,5,9,13
[354]	2	4	2	1	2,4,6,8,9,12,13,14,17
[422]	2	2	1	0	2,5,8,12,13
[32]	3	2	2	0	2,3,4,8,11,13,14
[32]	3	2	2	0	2,3,4,8,11,13,14
[269]	4	5	3	2	2-9,11,13-15,18,19
[17]	1	3	2	0	3,6,7,9,13,14
[339]	1	1	2	0	5,8,13,14
[325]	4	1	1	0	2-6,13
[467]	2	0	0	1	2,3,16
[50]	1	1	1	0	2,9,13
[298]	2	0	2	0	2,3,13,14
[130]	3	0	0	0	2,3,5
[353]	1	0	2	1	5,13,14,20
[233]	2	1	2	1	3,4,8,13,14,17
[347]	0	0	0	1	17
[480]	2	1	1	1	2,3,11,13,18
[116]	2	1	0	0	2,4,9
[225]	3	1	0	0	2,3,4,12
[470]	2	2	0	0	3,4,8,12
[90]	3	1	2	0	2,4,5,8,13,14
[219]	2	1	1	2	2,3,11,13,16,20
[220]	2	1	1	1	2,3,11,13,20
[428]	3	1	1	1	3-5,10,13,20
[203]	4	2	3	1	2,3,4,5,8,11,13-15,16
[376]	4	1	2	2	2-5,8,13,14,16,17
[348]	2	3	2	1	2,3,6,7,8,13,14,17
[492]	1	3	2	0	3,6,7,8,13,14
[287]	0	1	3	3	8,13-18
[91]	2	1	3	4	2,3,8,13-19
[342]	3	3	3	1	2,3,5,8,9,11,13-15,17
[54]	0	1	2	0	9,13,14
[105]	2	0	1	0	3,5,13
[218]	1	3	1	0	5,6,7,8,14
[305]	1	1	2	0	3,12,13,14
[357]	2	0	2	0	2,3,13,14
[365]	0	0	1	0	13
[208]	3	0	1	2	2,3,4,13,16,17
[374]	2	0	1	0	2,3,13
[194]	3	1	1	0	2,3,5,9,13
[354]	2	4	2	1	2,4,6,8,9,12-14,17
[422]	2	2	1	0	2,5,8,12,13
[52]	3	2	2	0	2,3,4,8,11,13,14
[269]	4	5	3	2	2-9,11,13-15,18,19
[17]	1	3	2	0	3,6,7,9,13,14
[339]	1	1	2	0	5,8,13,14
[325]	0	0	0	0	
[467]	2	0	0	1	2,3,16
[50]	1	1	1	0	2,9,13
[298]	2	0	2	0	2,3,13,14
[130]	3	0	0	0	2,3,5
[353]	1	0	2	1	5,13,14,20
[233]	2	1	2	1	3,4,8,13,14,17
[347]	0	0	0	1	17
[480]	2	1	1	1	2,3,11,13,18
[116]	2	1	0	0	2,4,9
[225]	3	1	0	0	2,3,4,12
[470]	2	2	0	0	3,4,8,12
[5]	3	1	2	0	3,4,5,8,13,14
[90]	3	1	2	0	2,4,5,8,13,14
[438]	1	1	1	0	5,8,13
[219]	2	1	1	2	2,3,11,13,16,20
[220]	2	1	1	1	2,3,11,13,20
[428]	3	1	1	1	3,4,5,10,13,20
[107]	0	1	1	1	11,13,17
[139]	2	1	1	0	4,5,11,13
[203]	4	2	3	1	2-5,8,11,13-16
[376]	4	1	2	2	2-5,8,13,14,16,17
[348]	2	3	2	1	2,3,6,7,8,13,14,17
[492]	1	3	2	0	3,6,7,8,13,14
[287]	0	1	3	3	8,13,14,15,16,17,18
[91]	2	1	3	4	2,3,8,13,14,15,16,17,18,19
[12]	2	7	3	3	2,3,6-18
[342]	3	3	3	1	2,3,5,8,9,11,13-15,17
[54]	0	1	2	0	9,13,14
[105]	2	0	1	0	3,5,13
[218]	1	3	1	0	5,6,7,8,14
[305]	1	1	2	0	3,12,13,14
[357]	2	0	2	0	2,3,13,14
[365]	0	0	1	0	13
[208]	3	0	1	2	2,3,4,13,16,17
[374]	2	0	1	0	2,3,13
[373]	2	1	1	0	2,3,11,13
[145]	2	1	2	0	2,3,11,13,15
[198]	4	1	1	2	2,3,4,5,11,13,16,17
[40]	3	4	1	4	2-4,6,7,9,11,13,16-19
[20]	3	1	0	0	2,3,4,11
[156]	1	1	0	0	2,6
[449]	2	1	1	1	2,4,11,13,16
[337]	3	1	2	0	2,3,4,11,13,14
[254]	2	4	2	3	2,4,6,7,8,11,13,14,16,17,18
[221]	3	1	1	1	2,3,4,11,13,20
[192]	3	4	1	1	2,3,4,8,9,10,11,13,16
[104]	2	0	0	0	2,3
[257]	0	1	1	0	11,13
[222]	2	0	1	0	3,4,13
[283]	4	4	0	0	2,3,4,5,6,7,10,11
[293]	4	0	0	0	2,3,4,5
[285]	2	1	0	2	2,3,11,18,19
[113]	3	1	1	1	2,4,5,11,13,20
[973]	2	1	1	0	2,3,11,13

Bibliography

- [1] Altera 1588 reference design. <https://www.intel.com/content/www/us/en/programmable/documentation/kly1423707073680.html>. Accessed: 2020-12-30.
- [2] Facebook buys black market passwords to keep your account safe. <https://www.cnet.com/news/facebook-chief-security-officer-alex-stamos-web-summit-lisbon-hackers/>. Accessed: 2020-15-12.
- [3] Xilinx Solarflare enhanced ptp user guide. <https://support-nic.xilinx.com/wp/drivers?sd=SF-109110-CD-8&pe=165>. Accessed: 2020-12-30.
- [4] Openflow switch specification, version 1.5.1 (protocol version 0x06), March 2015.
- [5] Sarra Abidi, Mehrez Essafi, Chirine Ghedira Guegan, Myriam Fakhri, Hamad Witt, and Henda Hjjami Ben Ghezala. A web service security governance approach based on dedicated micro-services. *Procedia Computer Science*, 159:372–386, 2019.
- [6] Multi access Edge Computing(MEC) ETSI Industry Specification Group. Mec; edge platform application enablement. 2019.
- [7] A Adam et al. The fog cloud of things: a survey on concepts, architecture, standards, tools, and applications. *internet things* 9 (2020), 2020.
- [8] Oluwasegun Adedugbe, Elhadj Benkhelifa, Russell Champion, Feras Al-Obeidat, Anoud Bani Hani, and Jayawickrama Uchitha. Leveraging cloud computing for the semantic web: review and trends. *Soft Computing*, pages 1–16, 2019.
- [9] Satyam Agarwal, Francesco Malandrino, Carla Fabiana Chiasserini, and Swades De. Vnf placement and resource allocation for the support of vertical services in 5g networks. *IEEE/ACM Transactions on Networking*, 27(1):433–446, 2019.
- [10] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov. Security in software defined networks: A survey. *IEEE Communications Surveys Tutorials*, 17(4):2317–2346, Fourthquarter 2015.

- [11] Mohsen Ahmadvand and Amjad Ibrahim. Requirements reconciliation for scalable and secure microservice (de) composition. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 68–73. IEEE, 2016.
- [12] Mohsen Ahmadvand, Alexander Pretschner, Keith Ball, and Daniel Eyring. Integrity protection against insiders in microservice-based infrastructures: From threats to a security framework. In *Federation of International Conferences on Software Technologies: Applications and Foundations*, pages 573–588. Springer, 2018.
- [13] Abdelmuttlib Ibrahim Abdalla Ahmed, Abdullah Gani, Siti Hafizah Ab Hamid, Abdelzahir Abdelmaboud, Hassan Jamil Syed, Riyaz Ahamed Ariyaluran Habeeb Mohamed, and Ihsan Ali. Service management for iot: Requirements, taxonomy, recent advances and open research challenges. *IEEE Access*, 7:155472–155488, 2019.
- [14] N. Ahuja and G. Singal. Ddos attack detection & prevention in sdn using openflow statistics. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, pages 147–152. IEEE, 2019.
- [15] S. B. Akers. Binary decision diagrams. *IEEE Trans. Comput.*, 27(6):509–516, June 1978.
- [16] Adnan Akhuzada, Abdullah Gani, Nor Badrul Anuar, Ahmed Abdelaziz, Muhammad Khuram Khan, Amir Hayat, and Samee U. Khan. Secure and dependable software defined networks. *J. Netw. Comput. Appl.*, 61(C):199–221, February 2016.
- [17] Sven Akkermans, Bruno Crispo, Wouter Joosen, and Danny Hughes. Polyglot cerberos: Resource security, interoperability and multi-tenancy for iot services on a multilingual platform. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 59–68, 2018.
- [18] Aya Al-Sakran, Mahmoud H Qutqut, Fadi Almasalha, Hossam S Hassanein, and Mohammad Hijjawi. An overview of the internet of things closed source operating systems. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 291–297. IEEE, 2018.
- [19] M. Alaggan, Gambs S, and A. Kermarrec. Blip: non-interactive differentially-private similarity computation on bloom filters. In *Symposium on Self-Stabilizing Systems*, pages 202–216. Springer, 2012.
- [20] Max Alaluna, Luís Ferrolho, José Rui Figueira, Nuno Neves, and Fernando MV Ramos. Secure multi-cloud virtual network embedding. *Computer Communications*, 155:252–265, 2020.

- [21] I. Alam, K. Sharif, F. Li, Z. Latif, M. Karim, S. Biswas, B. Nour, and Y. Wang. A survey of network virtualization techniques for internet of things using sdn and nfv. *ACM Computing Surveys (CSUR)*, 53(2):1–40, 2020.
- [22] T. Alharbi, S. Layeghy, and M. Portmann. Experimental evaluation of the impact of dos attacks in sdn. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6, 2017.
- [23] Isra Mohamed Ali, Maurantonio Caprolu, and Roberto Di Pietro. Foundations, properties, and security applications of puzzles: A survey. *ACM Computing Surveys (CSUR)*, 53(4):1–38, 2020.
- [24] Washington Henrique Carvalho Almeida, Luciano de Aguiar Monteiro, Raphael Rodrigues Hazin, Anderson Cavalcanti de Lima, and Felipe Silva Ferraz. Survey on microservice architecture-security, privacy and standardization on cloud computing environment. *ICSEA 2017*, pages 199–205, 2017.
- [25] N. Alshuqayran, N. Ali, and R. Evans. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51, 2016.
- [26] Hayriye Altunbasak, Sven Krasser, Henry L Owen, Jochen Grimminger, Hans-Peter Huth, and Joachim Sokol. Securing layer 2 in local area networks. In *International conference on networking*, pages 699–706. Springer, 2005.
- [27] Darwin Alulema, Javier Criado, Luis Iribarne, Antonio Jesús Fernández-García, and Rosa Ayala. A model-driven engineering approach for the service integration of iot systems. *Cluster Computing*, 23(3):1937–1954, 2020.
- [28] Rafael Álvarez-Sánchez, Alicia Andrade-Bazurto, Ivan Santos-González, and Antonio Zamora-Gómez. Aes-ctr as a password-hashing function. In *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain, September 6–8, 2017, Proceeding*, pages 610–617. Springer, 2017.
- [29] J. Amelot, J. Fletcher, D. Anand, C. Vasseur, Y. Li-Baboud, and J. Moyne. An ieee 1588 time synchronization testbed for assessing power distribution requirements. In *2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pages 13–18, 2010.
- [30] Sepehr Amir-Mohammadian and Chadi Kari. Correct audit logging in concurrent systems. *Electronic Notes in Theoretical Computer Science*, 351:115–141, 2020.

- [31] Michael P. Andersen, John Kolb, Kaifei Chen, David E. Culler, and Randy H. Katz. Old democratizing authority in the built environment. In Kamin Whitehouse, Prabal Dutta, and Hae Young Noh, editors, *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments, BuildSys 2017, Delft, The Netherlands, November 08-09, 2017*, pages 23:1–23:10. ACM, 2017.
- [32] Michael P Andersen, John Kolb, Kaifei Chen, Gabe Fierro, David E Culler, and Randy Katz. Democratizing authority in the built environment. *ACM Transactions on Sensor Networks (TOSN)*, 14(3-4):1–26, 2018.
- [33] Marco Anisetti, Claudio A Ardagna, Filippo Gaudenzi, and Ernesto Damiani. A continuous certification methodology for devops. In *Proceedings of the 11th International Conference on Management of Digital EcoSystems*, pages 205–212, 2019.
- [34] Robert Annessi, Joachim Fabini, Felix Iglesias, and Tanja Zseby. Encryption is futile: Delay attacks on high-precision clock synchronization. *arXiv preprint arXiv:1811.08569*, 2018.
- [35] D. Antonioli and N. Tippenhauer. Minicps: A toolkit for security research on cps networks. In *Proceedings of the First ACM workshop on cyber-physical systems-security and/or privacy*, pages 91–100, 2015.
- [36] F. M. Anwar, A. Alanwar, and M. B. Srivastava. Openclock: A testbed for clock synchronization research. In *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 1–6, 2018.
- [37] A. Appleby. Murmurhash 2.0, 2008.
- [38] Douglas Arnold. TLVs in PTP Messages. <https://blog.meinbergglobal.com/2019/12/06/tlvs-in-ntp-messages/>, 2019. [Online; accessed 22-Feb-2021].
- [39] Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Francesco Silvestri. Distance-sensitive hashing. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 89–104, 2018.
- [40] Alberto Avritzer, Vincenzo Ferme, Andrea Janes, Barbara Russo, André van Hoorn, Henning Schulz, Daniel Menasché, and Vilc Rufino. Scalability assessment of microservice architecture deployment configurations: A domain-based approach leveraging operational profiles and load tests. *Journal of Systems and Software*, page 110564, 2020.
- [41] Ataollah Fatahi Baarzi, George Kesidis, Dan Fleck, and Angelos Stavrou. Microservices made attack-resilient using unsupervised service fissioning. In *Proceedings of the 13th European workshop on Systems Security*, pages 31–36, 2020.

- [42] Mihai Baboi, Adrian Iftene, and Daniela Gifu. Dynamic microservices to create scalable and fault tolerance architecture. *Procedia Computer Science*, 159:1035–1044, 2019.
- [43] Claudio Badii, Pierfrancesco Bellini, Angelo Difino, Paolo Nesi, Gianni Pantaleo, and Michela Paolucci. Microservices suite for smart city applications. *Sensors*, 19(21):4798, 2019.
- [44] Oras Baker and Quy Nguyen. A novel approach to secure microservice architecture from owasp vulnerabilities. In *CITRENTZ Conference (2019)*, 2019.
- [45] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Microservices architecture enables devops: Migration to a cloud-native architecture. *Ieee Software*, 33(3):42–52, 2016.
- [46] Anna Bánáti, Eszter Kail, Krisztián Karóczkai, and Miklos Kozlovsky. Authentication and authorization orchestrator for microservice-based software architectures. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1180–1184. IEEE, 2018.
- [47] Alan Bandeira, Carlos Alberto Medeiros, Matheus Paixao, and Paulo Henrique Maia. We need to talk about microservices: an analysis from the discussions on stackoverflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 255–259. IEEE, 2019.
- [48] J. Barkman et al. Phytosociology and ecology of cryptogamic epiphytes (including a taxonomic survey and description of their vegetation units in europe). *Phytosociology and ecology of cryptogamic epiphytes (including a taxonomic survey and description of their vegetation units in Europe)*., 1969.
- [49] R. Bartak. Constraint programming: In pursuit of the holy grail. In *In Proceedings of the Week of Doctoral Students (WDS99 -invited lecture)*, volume Part IV, pages 555–564, Prague, Poland, 1999. MatFyzPress.
- [50] Tania Basso, Regina Moraes, Nuno Antunes, Marco Vieira, Walter Santos, and Wagner Meira. Privaaas: privacy approach for a distributed cloud-based data analytics platforms. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CC-GRID)*, pages 1108–1116. IEEE, 2017.
- [51] N. Bawany, J. Shamsi, and K. Salah. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. *Arabian Journal for Science and Engineering*, 42(2):425–441, 2017.

- [52] Ryan Beckett, Aarti Gupta, Ratul Mahajan, and David Walker. A general approach to network configuration verification. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 155–168. ACM, 2017.
- [53] Jethro G Beekman and Donald E Porter. Challenges for scaling applications across enclaves. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, pages 1–2, 2017.
- [54] Amin Beheshti, Boualem Benatallah, Alireza Tabebordbar, Hamid Reza Motahari-Nezhad, Moshe Chai Barukh, and Reza Nouri. Datasynapse: a social data curation foundry. *Distributed and Parallel Databases*, 37(3):351–384, 2019.
- [55] Maxime Bélaïr, Sylvie Laniece, and Jean-Marc Menaud. Leveraging kernel security mechanisms to improve container security: a survey. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–6, 2019.
- [56] Ron Bell. Introduction to iec 61508. In *ACM International Conference Proceeding Series*, volume 162, pages 3–12, 2006.
- [57] Steven M Bellovin and Michael Merritt. Limitations of the kerberos authentication system. *ACM SIGCOMM Computer Communication Review*, 20(5):119–132, 1990.
- [58] D. Berardi, F. Callegati, A. Melis, and M. Prandini. Security network policy enforcement through a sdn framework. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–4, Nov 2018.
- [59] D. Berardi, F. Callegati, A. Melis, and M. Prandini. Technetium: Atomic predicates and model driven development to verify security network policies. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, Jan 2020.
- [60] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O’Connor, Pavlin Radoslavov, William Snow, and Guru Parulkar. Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN ’14*, pages 1–6, 2014.
- [61] Antonia Bertolino, Guglielmo De Angelis, Antonio Guerriero, Breno Miranda, Roberto Pietrantuono, and Stefano Russo. Devopret: Continuous reliability testing in devops. *Journal of Software: Evolution and Process*, page e2298, 2020.
- [62] Ratnadeep Bhattacharya. Smart proxying for microservices. In *Proceedings of the 20th International Middleware Conference Doctoral Symposium*, pages 31–33, 2019.

- [63] Kaushal Bhavsar and Bhushan H Trivedi. An insider cyber threat prediction mechanism based on behavioral analysis. In *Proceedings of International Conference on ICT for Sustainable Development*, pages 345–353. Springer, 2016.
- [64] I. Bholebawa and U. Dalal. Design and performance analysis of openflow-enabled network topologies using mininet. *International Journal of Computer and Communication Engineering*, 5(6):419, 2016.
- [65] G. Bianchi, L. Bracciale, and Loreti Pierpaolo. ” better than nothing” privacy with bloom filters: To what extent? In *International Conference on Privacy in Statistical Databases*, pages 348–363. Springer, 2012.
- [66] Matt Bishop and Carrie Gates. Defining the insider threat. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, pages 1–3, 2008.
- [67] Nikolaj Bjørner, Garvit Juniwal, Ratul Mahajan, Sanjit A. Seshia, and George Varghese. ddnf: An efficient data structure for header spaces. In Roderick Bloem and Eli Arbel, editors, *Hardware and Software: Verification and Testing*, pages 49–64, Cham, 2016. Springer International Publishing.
- [68] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
- [69] Jasmin Bogatinovski, Sasho Nedelkoski, Jorge Cardoso, and Odej Kao. Self-supervised anomaly detection from distributed traces. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 342–347. IEEE, 2020.
- [70] Justus Bogner, Jonas Fritzsich, Stefan Wagner, and Alfred Zimmermann. Microservices in industry: insights into technologies, characteristics, and software quality. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 187–195. IEEE, 2019.
- [71] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.
- [72] Karoly Bozan, Kalle Lyytinen, and Gregory M Rose. How to transition incrementally to microservice architecture. *Communications of the ACM*, 64(1):79–85, 2020.

- [73] Marco Brambilla, Eric Umuhoza, and Roberto Acerbis. Model-driven development of user interfaces for iot systems via domain-specific components and patterns. *Journal of Internet Services and Applications*, 8(1):14, 2017.
- [74] Stefan Brenner, Tobias Hundt, Giovanni Mazzeo, and Rüdiger Kapitza. Secure cloud micro services using intel sgx. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 177–191. Springer, 2017.
- [75] T. Bridi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini. A constraint programming scheduler for heterogeneous high-performance computing machines. *IEEE transactions on parallel and distributed systems*, 27(10):2781–2794, 2016.
- [76] Andrey Brito, Christof Fetzer, Stefan Köpsell, Peter Pietzuch, Marcelo Pasin, Pascal Felber, Keiko Fonseca, Marcelo Rosa, Luiz Gomes, Rodrigo Riella, et al. Secure end-to-end processing of smart metering data. *Journal of Cloud Computing*, 8(1):1–13, 2019.
- [77] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.
- [78] Yérom-David Bromberg and Louison Gitzinger. Droidautoml: A microservice architecture to automate the evaluation of android machine learning detection systems. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 148–165. Springer, 2020.
- [79] Rolando Brondolin and Marco D Santambrogio. A black-box monitoring approach to measure microservices runtime performance. *ACM Transactions on Architecture and Code Optimization (TACO)*, 17(4):1–26, 2020.
- [80] A. Brown, S. Randall, J. Boyd, and A. Ferrante. Evaluation of approximate comparison methods on bloom filters for probabilistic linkage. *International Journal of Population Data Science*, 4(1):1–16, 2019.
- [81] Achim D Brucker, Bo Zhou, Francesco Malmignati, Qi Shi, and Madjid Merabti. Modelling, validating, and ranking of secure service compositions. *Software: Practice and Experience*, 47(12):1923–1943, 2017.
- [82] Daniel Bumblauskas, Arti Mann, Brett Dugan, and Jacy Rittmer. A blockchain use case in food distribution: Do you know where your food has been? *International Journal of Information Management*, 52:102008, 2020.

- [83] B. Butler. What p4 programming is and why it's such a big deal for software defined networking, Jan 2017.
- [84] Brandon. Butler. What p4 programming is and why it's such a big deal for software defined networking., 2016.
- [85] Alina Buzachis and Massimo Villari. Basic principles of osmotic computing: Secure and dependable microelements (mels) orchestration leveraging blockchain facilities. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 47–52. IEEE, 2018.
- [86] Béatrice Bérard, M Bidoit, Alain Finkel, F Laroussinie, A Petit, Laure Petrucci, and Ph Schnoebelen. Systems and software verification: Model-checking techniques and tools. 01 2001.
- [87] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-middle attack to the https protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.
- [88] Franco Callegati, Saverio Giallorenzo, Maurizio Gabbrielli, Andrea Melis, and Marco Prandini. Federated platooning: Insider threats and mitigations. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.
- [89] Franco Callegati, Saverio Giallorenzo, Andrea Melis, and Marco Prandini. Data security issues in maas-enabling platforms. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–5. IEEE, 2016.
- [90] Franco Callegati, Saverio Giallorenzo, Andrea Melis, and Marco Prandini. Cloud-of-things meets mobility-as-a-service: An insider threat perspective. *Computers & Security*, 74:277–295, 2018.
- [91] Matteo Camilli, Carlo Bellettini, Lorenzo Capra, and Mattia Monga. A formal framework for specifying and verifying microservices based process flows. In *International Conference on Software Engineering and Formal Methods*, pages 187–202. Springer, 2017.
- [92] G Casale, M Artač, W-J van den Heuvel, A Van Hoorn, P Jakovits, F Leymann, M Long, V Papanikolaou, D Presenza, A Russo, et al. Radon: rational decomposition and orchestration for serverless computing. *SICS Software-Intensive Cyber-Physical Systems*, pages 1–11, 2019.
- [93] Giuliano Casale, Cristina Chesta, Peter Deussen, Elisabetta Di Nitto, Panagiotis Gouvas, Sotiris Koussouris, Vlado Stankovski, Andreas Symeonidis, Vlassis Vlassiou, Anastasios

- Zafeiropoulos, et al. Current and future challenges of software engineering for services and applications. In *Cloud Forward*, pages 34–42, 2016.
- [94] Emiliano Casalicchio and Stefano Iannucci. The state-of-the-art in container technologies: Application, orchestration and security. *Concurrency and Computation: Practice and Experience*, page e5668, 2020.
- [95] Tomas Cerny and Michael J Donahoo. Survey on concern separation in service integration. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 518–531. Springer, 2016.
- [96] Tomas Cerny, Filip Sedlisky, and Michael J Donahoo. On isolation-driven automated module decomposition. In *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, pages 302–307, 2018.
- [97] Tomas Cerny, Jan Svacina, Dipta Das, Vincent Bushong, Miroslav Bures, Pavel Tisnovsky, Karel Frajtak, Dongwan Shin, and Jun Huang. On code analysis opportunities and challenges for enterprise systems and microservices. *IEEE Access*, 8:159449–159470, 2020.
- [98] Ramaswamy Chandramouli. Microservices-based application systems. *NIST Special Publication*, 800:204, 2019.
- [99] B. Charles. Automating orchestration in the cloud with ubuntu juju. Philadelphia, PA, June 2014. USENIX Association.
- [100] Virginia T Chavira. *Insider Threats and the Effects of Operant Conditioning*. PhD thesis, Utica College, 2017.
- [101] Chien-An Chen. With great abstraction comes great responsibility: Sealing the microservices attack surface. In *2019 IEEE Cybersecurity Development (SecDev)*, pages 144–144. IEEE, 2019.
- [102] Hongyang Chen, Pengfei Chen, and Guangba Yu. A framework of virtual war room and matrix sketch-based streaming anomaly detection for microservice systems. *IEEE Access*, 8:43413–43426, 2020.
- [103] Jiyu Chen, Heqing Huang, and Hao Chen. Informer: irregular traffic detection for containerized microservices rpc in the real world. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 389–394, 2019.

- [104] Sunil Cheruvu, Anil Kumar, Ned Smith, and David M Wheeler. *Demystifying internet of things security: successful iot device/edge and platform security deployment*. Springer Nature, 2020.
- [105] Nithya Chidambaram, Pethuru Raj, K Thenmozhi, Sundararaman Rajagopalan, and Ren-garajan Amirtharajan. A cloud compatible dna coded security solution for multimedia file sharing & storage. *Multimedia Tools and Applications*, 78(23):33837–33863, 2019.
- [106] Nacha Chondamrongkul, Jing Sun, and Ian Warren. Automated security analysis for microservice architecture. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 79–82. IEEE, 2020.
- [107] Michele Ciavotta, Marino Alge, Silvia Menato, Diego Rovere, and Paolo Pedrazzoli. A microservice-based middleware for the digital factory. *Procedia Manufacturing*, 11:931–938, 2017.
- [108] T Charles Clancy, Robert W McGwier, and Lidong Chen. Post-quantum cryptography and 5g security: tutorial. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 285–285, 2019.
- [109] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.
- [110] Sean B Cleveland, Anagha Jamthe, Smruti Padhy, Joe Stubbs, Michale Packard, Julia Looney, Steve Terry, Richard Cardone, Maytal Dahan, and Gwen A Jacobs. Tapis api development with python: best practices in scientific rest api implementation: experience implementing a distributed stream api. In *Practice and Experience in Advanced Research Computing*, pages 181–187. 2020.
- [111] Richard Cochran et al. The linux ptp project, 2015.
- [112] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.
- [113] Sebastian Copei, Manuel Wickert, and Albert Zündorf. Certification as a service. In Maria Paasivaara and Philippe Kruchten, editors, *Agile Processes in Software Engineering and Extreme Programming – Workshops*, pages 203–210, Cham, 2020. Springer International Publishing.
- [114] Bruno Costa, Paulo F Pires, and Flávia C Delicato. Towards the adoption of omg standards in the development of soa-based iot systems. *Journal of Systems and Software*, 169:110720, 2020.

- [115] Alberto Rodrigues Da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155, 2015.
- [116] Matheus Sthefano Leite da Silva, Fábio Fernando de Oliveira Silva, and Andrey Brito. Squad: A secure, simple storage service for sgx-based microservices. In *2019 9th Latin-American Symposium on Dependable Computing (LADC)*, pages 1–9. IEEE, 2019.
- [117] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich. Security challenges and opportunities of software-defined networking. *IEEE Security Privacy*, 15(2):96–100, March 2017.
- [118] Haitham Abu Damis, Dina Shehada, Claude Fachkha, Amjad Gawanmeh, and Jamal N Al-Karaki. A microservices architecture for ads-b data security using blockchain. In *2020 3rd International Conference on Signal Processing and Information Security (ICSPIS)*, pages 1–4. IEEE, 2020.
- [119] Pandit Byomakesha Dash, Janmenjoy Nayak, Bighnaraj Naik, Etuari Oram, and SK Hafizul Islam. Model based iot security framework using multiclass adaptive boosting with smote. *Security and Privacy*, 3(5):e112, 2020.
- [120] B. Davie and Y. Rekhter. *MPLS: technology and applications*. Morgan Kaufmann Publishers Inc., 2000.
- [121] Angelita Rettore de Araujo Zanella, Eduardo da Silva, and Luiz Carlos Pessoa Albini. Security challenges to smart agriculture: Current state, key issues, and future directions. *Array*, page 100048, 2020.
- [122] Michele De Donno, Alberto Giaretta, Nicola Dragoni, Antonio Bucchiarone, and Manuel Mazzara. Cyber-storms come from clouds: Security of cloud computing in the iot era. *Future Internet*, 11(6):127, 2019.
- [123] Thatiane de Oliveira Rosa, João Francisco Lino Daniel, Eduardo Martins Guerra, and Alfredo Goldman. A method for architectural trade-off analysis based on patterns: Evaluating microservices structural attributes. In *Proceedings of the European Conference on Pattern Languages of Programs 2020*, pages 1–8, 2020.
- [124] Pamela Soares de Sousa, Nataniel Parente Nogueira, Rayane Celestino dos Santos, Paulo Henrique M Maia, and Jerffeson Teixeira de Souza. Building a prototype based on microservices and blockchain technologies for notary’s office: An academic experience report. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 122–129. IEEE, 2020.

- [125] Saulo S de Toledo, Antonio Martini, and Dag IK Sjøberg. Improving agility by managing shared libraries in microservices. In *International Conference on Agile Software Development*, pages 195–202. Springer, 2020.
- [126] Darren Death. *Information security handbook: develop a threat model and incident response strategy to build a strong information security framework*. Packt Publishing Ltd, 2017.
- [127] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert. Impact of cyberattacks on precision time protocol. *IEEE Transactions on Instrumentation and Measurement*, 69(5):2172–2181, 2020.
- [128] M. Deep and R. Karthik. Chapter 4 - network flow models. In Deep Medhi and Karthik Ramasamy, editors, *Network Routing (Second Edition)*, The Morgan Kaufmann Series in Networking, pages 114 – 157. Morgan Kaufmann, Boston, second edition edition, 2018.
- [129] Flavia C Delicato, Adnan Al-Anbuky, I Kevin, and Kai Wang. Smart cyber–physical systems: Toward pervasive intelligence systems, 2020.
- [130] Henri Maxime Demoulin, Tavish Vaidya, Isaac Pedisich, Bob DiMaiolo, Jingyu Qian, Chirag Shah, Yuankai Zhang, Ang Chen, Andreas Haeberlen, Boon Thau Loo, et al. Dedos: Defusing dos with dispersion oriented software. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 712–722, 2018.
- [131] James DesLauriers, Tamas Kiss, Resmi C Ariyattu, Hai-Van Dang, Amjad Ullah, James Bowden, Dagmar Krefting, Gabriele Pierantoni, and Gabor Terstyanszky. Cloud apps to-go: Cloud portability with toasca and micado. *Concurrency and Computation: Practice and Experience*, page e6093, 2020.
- [132] Favian Dewanta. Secure microservices deployment for fog computing services in a remote office. In *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*, pages 425–430. IEEE, 2020.
- [133] Claudio Di Ciccio, Alessio Cecconi, Marlon Dumas, Luciano García-Bañuelos, Orlenys López-Pintado, Qinghua Lu, Jan Mendling, Alexander Ponomarev, An Binh Tran, and Ingo Weber. Blockchain support for collaborative business processes. *Informatik Spektrum*, 42(3):182–190, 2019.
- [134] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, 150:77 – 97, 2019.

- [135] Paolo Di Francesco, Ivano Malavolta, and Patricia Lago. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 21–30. IEEE, 2017.
- [136] Elisabetta Di Nitto, Peter Matthews, Dana Petcu, and Arnor Solberg. *Model-driven Development and Operation of Multi-cloud Applications: The ModacLOUDS Approach*. Springer, 2017.
- [137] Amleto Di Salle, Francesco Gallo, and Claudio Pompilio. Composition of advanced (μ) services for the next generation of the internet of things. In *Federation of International Conferences on Software Technologies: Applications and Foundations*, pages 436–444. Springer, 2016.
- [138] Pierangelo Di Sanzo, Dimiter R Avresky, and Alessandro Pellegrini. Autonomic rejuvenation of cloud applications as a countermeasure to software anomalies. *Software: Practice and Experience*, 51(1):46–71, 2021.
- [139] Daniel Díaz-Sánchez, Andrés Marín-Lopez, Florina Almenárez Mendoza, and Patricia Arias Cabarcos. Dns/dane collision-based distributed and dynamic authentication for microservices in iot. *Sensors*, 19(15):3292, 2019.
- [140] L. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.
- [141] Cornelius Diekmann, Johannes Naab, Andreas Korsten, and Georg Carle. Agile network access control in the container age. *IEEE Transactions on Network and Service Management*, 16(1):41–55, 2018.
- [142] Dulaj Dilshan, Supimi Piumika, Chameera Rupasinghe, Indika Perera, and Prabath Siriwardena. Mschain: Blockchain based decentralized certificate transparency for microservices. In *2020 Moratuwa Engineering Research Conference (MERCOn)*, pages 1–6. IEEE, 2020.
- [143] 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.
- [144] L. Dridi and M. Zhani. Sdn-guard: Dos attacks mitigation in sdn networks. In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, pages 212–217. IEEE, 2016.
- [145] Dong Du, Tianyi Yu, Yubin Xia, Binyu Zang, Guanglu Yan, Chenggang Qin, Qixuan Wu, and Haibo Chen. Catalyzer: Sub-millisecond startup for serverless computing with initialization-less booting. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 467–481, 2020.

- [146] Qingfeng Du, Tiandi Xie, and Yu He. Anomaly detection and diagnosis for container-based microservices with performance monitoring. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 560–572. Springer, 2018.
- [147] C. Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.
- [148] I Hilmi Elifoglu, Ivan Abel, and Özlem Taşseven. Minimizing insider threat risk with behavioral monitoring. *Review of business*, 38(2):61–73, 2018.
- [149] Marwa Elsayed and Mohammad Zulkernine. Offering security diagnosis as a service for cloud saas applications. *Journal of information security and applications*, 44:32–48, 2019.
- [150] Úl. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.
- [151] Silvia Esparrachiari, Tanya Reilly, and Ashleigh Rentz. Tracking and controlling microservice dependencies. *Queue*, 16(4):44–65, 2018.
- [152] Christian Esposito, Aniello Castiglione, Constantin-Alexandru Tudorica, and Florin Pop. Security and privacy for cloud-based data management in the health network service chain: a microservice approach. *IEEE Communications Magazine*, 55(9):102–108, 2017.
- [153] Network Functions Virtualisation ETSI. Network functions virtualisation (nfv). *Management and Orchestration*, 1:V1, 2014.
- [154] OSM ETSI. Open source mano. *OSM home page*, 2016.
- [155] Mahdi Fahmideh and Didar Zowghi. An exploration of iot platform development. *Information Systems*, 87:101409, 2020.
- [156] Muhammad Fajrul Falah, Sritrusta Sukaridhoto, Muhammad Udin Harun Al Rasyid, and Hendro Wicaksono. Design of virtual engineering and digital twin platform as implementation of cyber-physical systems. *Procedia Manufacturing*, 52:331–336, 2020.
- [157] Christof Fetzter, Giovanni Mazzeo, John Oliver, Luigi Romano, and Martijn Verburg. Integrating reactive cloud applications in sereca. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–8, 2017.
- [158] Norman Finn. Introduction to time-sensitive networking. *IEEE Communications Standards Magazine*, 2(2):22–28, 2018.

- [159] José Flora. Improving the security of microservice systems by detecting and tolerating intrusions. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 131–134. IEEE, 2020.
- [160] José Flora, Paulo Gonçalves, and Nuno Antunes. Using attack injection to evaluate intrusion detection effectiveness in container-based systems. In *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 60–69. IEEE, 2020.
- [161] D. Florencio and C. Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666, 2007.
- [162] Lori Flynn, Greg Porter, and Chas DiFatta. Cloud service provider methods for managing insider threats: Analysis phase 2, expanded analysis and recommendations. Technical report, Pittsburgh University, 2014.
- [163] S. Forman and B. Samanthula. Secure similar document detection: Optimized computation using the jaccard coefficient. 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 1–4. IEEE, 2018.
- [164] Stefano Forti, Gian-Luigi Ferrari, and Antonio Brogi. Secure cloud-edge deployments, with trust. *Future Generation Computer Systems*, 102:775–788, 2020.
- [165] E.C. Freuder. In pursuit of the holy grail. *Constraints*, 2(1):57–61, 1997.
- [166] Saurabh Ganeriwal, Srdjan Čapkun, Chih-Chieh Han, and Mani B Srivastava. Secure time synchronization service for sensor networks. In *Proceedings of the 4th ACM workshop on Wireless security*, pages 97–106, 2005.
- [167] Saurabh Ganeriwal, Christina Pöpper, Srdjan Čapkun, and Mani B Srivastava. Secure time synchronization in sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 11(4):1–35, 2008.
- [168] Somya Garg and Satvik Garg. Automated cloud infrastructure, continuous integration and continuous delivery using docker with robust container security. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 467–470. IEEE, 2019.
- [169] Martin Garriga. Towards a taxonomy of microservices architectures. In *International Conference on Software Engineering and Formal Methods*, pages 203–218. Springer, 2017.

- [170] M. Gasser. A random word generator for pronounceable passwords. Technical report, MITRE CORP BEDFORD MA, 1975.
- [171] Vathalloor Merin George and Qusay H Mahmoud. Claimsware: A claims-based middleware for securing iot services. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 649–654. IEEE, 2017.
- [172] Christopher Gerking and David Schubert. Component-based refinement and verification of information-flow security policies for cyber-physical microservice architectures. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 61–70. IEEE, 2019.
- [173] Ilias Gerostathopoulos, Tomas Bures, et al. A toolbox for realtime timeseries anomaly detection. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 278–281. IEEE, 2020.
- [174] Shahbaz Ahmed Khan Ghayyur, Abdul Razzaq, Saeed Ullah, and Salman Ahmed. Matrix clustering based migration of system application to microservices architecture. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 9(1):284–296, 2018.
- [175] Suyash S Ghuge, Nishant Kumar, S Savitha, and V Suraj. Multilayer technique to secure data transfer in private cloud for saas applications. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 646–651. IEEE, 2020.
- [176] Federico Giaimo, Hugo Andrade, and Christian Berger. Continuous experimentation and the cyber-physical systems challenge: An overview of the literature and the industrial perspective. *Journal of Systems and Software*, 170:110781, 2020.
- [177] H. Gilbert and H. Handschuh. Security analysis of sha-256 and sisters. In *International workshop on selected areas in cryptography*, pages 175–193. Springer, 2003.
- [178] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.
- [179] Deeksha Gorige, Eyhab Al-Masri, Sergey Kanzhelev, and Hossam Fattah. Privacy-risk detection in microservices composition using distributed tracing. In *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pages 250–253. IEEE, 2020.
- [180] Paul A Grassi, M. Garcia E, and J. Fenton. Draft nist special publication 800-63-3 digital identity guidelines. *National Institute of Standards and Technology, Los Altos, CA*, 2017.

- [181] F. L. Greitzer, J. Purl, Y. M. Leong, and P. J. Sticha. Positioning your organization to respond to insider threats. *IEEE Engineering Management Review*, 47(2):75–83, 2019.
- [182] Frank Greitzer, Justin Purl, DE Becker, Paul Sticha, and Yung Mei Leong. Modeling expert judgments of insider threat using ontology structure: Effects of individual indicator threat value and class membership. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.
- [183] Frank L. Greitzer. Insider threats: It’s the human, stupid! In *Proceedings of the Northwest Cybersecurity Symposium, NCS ’19*, New York, NY, USA, 2019. Association for Computing Machinery.
- [184] L. Gremillion. Designing a bloom filter for differential file access. *Communications of the ACM*, 25(9):600–604, 1982.
- [185] S. Gueron, S. Johnson, and J. Walker Jesse. Sha-512/256. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 354–358. IEEE, 2011.
- [186] Daniel Guija and Muhammad Shuaib Siddiqui. Identity and access control for micro-services based 5g nfv platforms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10, 2018.
- [187] Yu Guo, Cong Wang, and Xiaohua Jia. Enabling secure and dynamic deep packet inspection in outsourced middleboxes. In *Proceedings of the 6th International Workshop on Security in Cloud Computing*, pages 49–55, 2018.
- [188] Rajeev Kumar Gupta, Mekanathan Venkatachalapathy, and Ferose Khan Jeberla. Challenges in adopting continuous delivery and devops in a globally distributed product team: a case study of a healthcare organization. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, pages 30–34. IEEE, 2019.
- [189] Mordechai Guri, Yosef Solewicz, Andrey Daidakulov, and Yuval Elovici. Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers. *arXiv preprint arXiv:1606.05915*, 2016.
- [190] Mordechai Guri, Boris Zadov, Dima Bykhovsky, and Yuval Elovici. Powerhammer: Exfiltrating data from air-gapped computers through power lines. *arXiv preprint arXiv:1804.04014*, 2018.
- [191] Z. Gutterman, B. Pinkas, and T. Reinman. Analysis of the linux random number generator. In *2006 IEEE Symposium on Security and Privacy (S&P’06)*, pages 15–pp. IEEE, 2006.

- [192] Dalton A Hahn, Drew Davidson, and Alexandru G Bardas. Mismatch: Security issues and challenges in service meshes. In *International Conference on Security and Privacy in Communication Systems*, pages 140–151. Springer, 2020.
- [193] Jeremy Hajek, Mudassir Rashid, Mert Sevil, Ali Cinar, Pablo Angel Alvarez Fernandez, and Dhiraj Jain. The necessity of interdisciplinary software development for building viable research platforms: Case study in automated drug delivery in diabetes. In *Proceedings of the 21st Annual Conference on Information Technology Education*, pages 390–396, 2020.
- [194] Juhyeng Han, Seongmin Kim, Taesoo Kim, and Dongsu Han. Toward scaling hardware security module for emerging cloud services. In *Proceedings of the 4th Workshop on System Software for Trusted Execution*, pages 1–6, 2019.
- [195] Mingyu Han and Peter Crossley. Vulnerability of iee 1588 under time synchronization attacks. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2019.
- [196] Lei Hang, Israr Ullah, and Do-Hyeun Kim. A secure fish farm platform based on blockchain for agriculture data integrity. *Computers and Electronics in Agriculture*, 170:105251, 2020.
- [197] Abdelhakim Hannousse and Salima Yahiouche. Securing microservices and microservice architectures: A systematic mapping study, 2020.
- [198] Mubin Ul Haque, Leonardo Horn Iwaya, and M Ali Babar. Challenges in docker development: A large-scale study using stack overflow. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2020.
- [199] Mahmud Hasan and Binil Starly. Decentralized cloud manufacturing-as-a-service (cmaas) platform architecture with configurable digital assets. *Journal of Manufacturing Systems*, 56:157–174, 2020.
- [200] Nelson E Hastings and Paul A McLean. Tcp/ip spoofing fundamentals. In *Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, pages 218–224. IEEE, 1996.
- [201] Xiuyu He and Xudong Yang. Authentication and authorization of end user in microservice architecture. In *Journal of Physics: Conference Series*, volume 910, page 012060. IOP Publishing, 2017.
- [202] Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. Serverless computation with openlambda. In *8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

- [203] Jrgen Kjell Hole. *Anti-fragile ICT systems*. Springer-Verlag GmbH, 2016.
- [204] Ivan Homoliak, Flavio Toffalini, Juan Guarnizo, Yuval Elovici, and Martn Ochoa. Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Comput. Surv.*, 52(2), April 2019.
- [205] A. Horn, A. Kheradmand, and M. Prasad. Delta-net: Real-time network verification using atoms. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 735–749, 2017.
- [206] Tony Hsiang-Chih Hsu. *Hands-On Security in DevOps: Ensure continuous security, deployment, and delivery with DevSecOps*. Packt Publishing Ltd, 2018.
- [207] Fei Hu, Qi Hao, and Ke Bao. A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys & Tutorials*, 16(4):2181–2206, 2014.
- [208] Amjad Ibrahim, Stevica Bozhinoski, and Alexander Pretschner. Attack graph generation for microservice architecture. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1235–1242, 2019.
- [209] P. Indyk and Motwani Rajeev. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.
- [210] IEEE Instrumentation and Measurement Society. Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. 2002.
- [211] IEEE Instrumentation and Measurement Society. Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. 2008.
- [212] IEEE Instrumentation and Measurement Society. Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. 2019.
- [213] IEEE Instrumentation and Measurement Society. Ieee standard for local and metropolitan area networks—timing and synchronization for time-sensitive applications. 2020.
- [214] Omar Iraqi and Hanan El Bakkali. Immunizer: A scalable loosely-coupled self-protecting software framework using adaptive microagents and parallelized microservices. In *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 24–27. IEEE, 2020.

- [215] Tariqul Islam, D Manivannan, and Sherali Zeadally. A classification and characterization of security threats in cloud computing. *Int. J. Next-Gener. Comput*, 7(1), 2016.
- [216] B. Ives, K. Walsh, and H. Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4):75–78, 2004.
- [217] P. Jaccard. *Le coefficient generique et le coefficient de communaute dans la flore marocaine*. Impr. Commerciale, 1926.
- [218] Sadeeq Jan, Annibale Panichella, Andrea Arcuri, and Lionel Briand. Search-based multi-vulnerability testing of xml injections in web applications. *Empirical Software Engineering*, 24(6):3696–3729, 2019.
- [219] Kai Jander, Lars Braubach, and Alexander Pokahr. Defense-in-depth and role authentication for microservice systems. *Procedia computer science*, 130:456–463, 2018.
- [220] Kai Jander, Lars Braubach, and Alexander Pokahr. Practical defense-in-depth solution for microservice systems. *Journal of Ubiquitous Systems & Pervasive Networks*, 11(1):17–25, 2019.
- [221] Kanwal Janjua, Munam Ali Shah, Ahmad Almogren, Hasan Ali Khattak, Carsten Maple, and Ikram Ud Din. Proactive forensics in iot: privacy-aware log-preservation architecture in fog-enabled-cloud using holochain and containerization technologies. *Electronics*, 9(7):1172, 2020.
- [222] Asad Javed, Jérémy Robert, Keijo Heljanko, and Kary Främling. Iotef: A federated edge-cloud architecture for fault-tolerant iot applications. *Journal of Grid Computing*, pages 1–24, 2020.
- [223] Janusz Jaworski, Waldemar Karwowski, and Marian Rusek. Microservice-based cloud application ported to unikernels: Performance comparison of different technologies. In *International Conference on Information Systems Architecture and Technology*, pages 255–264. Springer, 2019.
- [224] S. Ji, S. Yang, A. Das, X. Hu, and R. Beyah. Password correlation: Quantification, evaluation and application. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.
- [225] Hai Jin, Zhi Li, Deqing Zou, and Bin Yuan. Dseom: A framework for dynamic security evaluation and optimization of mtd in container-based cloud. *IEEE Transactions on Dependable and Secure Computing*, 2019.

- [226] Mingxu Jin, Aoran Lv, Yuanpeng Zhu, Zijiang Wen, Yubin Zhong, Zexin Zhao, Jiang Wu, Hejie Li, Hanheng He, and Fengyi Chen. An anomaly detection algorithm for microservice architecture based on robust principal component analysis. *IEEE Access*, 2020.
- [227] Hlengekile Jita and Vreda Pieterse. A framework to apply the internet of things for medical care in a home environment. In *Proceedings of the 2018 International Conference on Cloud Computing and Internet of Things*, pages 45–54, 2018.
- [228] Christina Terese Joseph and K Chandrasekaran. Straddling the crevasse: A review of microservice software architecture foundations and recent advancements. *Software: Practice and Experience*, 49(10):1448–1484, 2019.
- [229] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic. A survey on data plane flexibility and programmability in software-defined networking. *IEEE Access*, 7:47804–47840, 2019.
- [230] Dimitrios Kallergis, Zacharenia Garofalaki, Georgios Katsikogiannis, and Christos Douligeris. Capodaz: A containerised authorisation and policy-driven architecture using microservices. *Ad Hoc Networks*, 104:102153, 2020.
- [231] Miika Kalske, Niko Mäkitalo, and Tommi Mikkonen. Challenges when moving from monolith to microservice architecture. In *International Conference on Web Engineering*, pages 32–47. Springer, 2017.
- [232] Keshav Govind Kamble and Amitabh Sinha. Service level agreements and application defined security policies for application and data security registration, dec 2016. US Patent App. 15/191,420.
- [233] Moonjoong Kang, Jun-Sik Shin, and JongWon Kim. Protected coordination of service mesh for container-based 3-tier service traffic. In *2019 International Conference on Information Networking (ICOIN)*, pages 427–429. IEEE, 2019.
- [234] Rui Kang, Zhenyu Zhou, Jiahua Liu, Zhongran Zhou, and Shunwang Xu. Distributed monitoring system for microservices-based iot middleware system. In *International Conference on Cloud Computing and Security*, pages 467–477. Springer, 2018.
- [235] Stefan Kapferer and Olaf Zimmermann. Domain-driven service design. In *Symposium and Summer School on Service-Oriented Computing*, pages 189–208. Springer, 2020.
- [236] Pradeeban Kathiravelu, Peter Van Roy, and Luís Veiga. Sd-cps: software-defined cyber-physical systems. taming the challenges of cps with workflows at the edge. *Cluster Computing*, 22(3):661–677, 2019.

- [237] Peyman Kazemian, Michael Chang, Hongyi Zeng, George Varghese, Nick McKeown, and Scott Whyte. Real time network policy checking using header space analysis. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 99–111, 2013.
- [238] Peyman Kazemian, George Varghese, and Nick McKeown. Header space analysis: Static checking for networks. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 9–9, 2012.
- [239] Haoren Ke, Hao Wu, and Dongmei Yang. Towards evolving security requirements of industrial internet: A layered security architecture solution based on data transfer techniques. In *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, pages 504–511, 2020.
- [240] 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*, 28:1–28, 2018.
- [241] Florian Kelbert, Franz Gregor, Rafael Pires, Stefan Köpsell, Marcelo Pasin, Aurélien Havet, Valerio Schiavoni, Pascal Felber, Christof Fetzter, and Peter Pietzuch. Securecloud: Secure big data processing in untrusted clouds. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 282–285. IEEE, 2017.
- [242] Arif Ali Khan and Mohammad Shameem. Multicriteria decision-making taxonomy for devops challenging factors using analytical hierarchy process. *Journal of Software: Evolution and Process*, 32(10):e2263, 2020.
- [243] Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. Veriflow: Verifying network-wide invariants in real time. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 15–27, 2013.
- [244] J. Kim, A. Biryukov, B. Preneel, and S. Hong. On the security of hmac and nmac based on haval, md4, md5, sha-0 and sha-1. In *International Conference on Security and Cryptography for Networks*, pages 242–256. Springer, 2006.
- [245] A. Kirsch and M. Mitzenmacher. Distance-sensitive bloom filters. In *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 41–50. SIAM, 2006.

- [246] Petar Kochovski, Sandi Gec, Vlado Stankovski, Marko Bajec, and Pavel D Drobintsev. Trust management in a blockchain based fog computing platform with trustless smart oracles. *Future Generation Computer Systems*, 101:747–759, 2019.
- [247] Loren Kohnfelder and Praerit Garg. The threats to our products. *Microsoft Interface, Microsoft Corporation*, 33, 1999.
- [248] J. Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3:91 – 97, 2006. The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06).
- [249] D. Kotani and Y. Okabe. A packet-in message filtering mechanism for protection of control plane in openflow networks. In *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 29–40, 2014.
- [250] Michel Krämer, Sven Frese, and Arjan Kuijper. Implementing secure applications in smart city clouds using microservices. *Future Generation Computer Systems*, 99:308–320, 2019.
- [251] Diego Kreutz, Fernando M.V. Ramos, and Paulo Verissimo. Towards secure and dependable software-defined networks. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13*, pages 55–60, 2013.
- [252] Prabhakar Krishnan, Subhasri Duttagupta, and Krishnashree Achuthan. Sdn/nfv security framework for fog-to-things computing infrastructure. *Software: Practice and Experience*, 2019.
- [253] K. Kuchcinski and R. Szymanek. Jacop-java constraint programming solver. In *CP Solvers: Modeling, Applications, Integration, and Standardization, co-located with the 19th International Conference on Principles and Practice of Constraint Programming*, 2013.
- [254] Rakesh Kumar and Rinkaj Goyal. Modeling continuous security: A conceptual model for automated devsecops using open-source software over cloud (adoc). *Computers & Security*, 97:101967, 2020.
- [255] Soonhong Kwon, Sang-Jin Son, Yangseo Choi, and Jong-Hyouk Lee. Protocol fuzzing to find security vulnerabilities of rabbitmq. *Concurrency and Computation: Practice and Experience*, page e6012, 2020.
- [256] C. Laissaoui, N. Idboufker, R. Ellassali, and K. El Baamrani. A measurement of the response times of various openflow/sdn controllers with cbench. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–2. IEEE, 2015.

- [257] Abdullah Lakhani and Xiaoping Li. Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks. *Computing*, 102(1):105–139, 2020.
- [258] Sebastian Łaskawiec, Michał Choraś, and Rafał Kozik. New solutions for exposing clustered applications deployed in the cloud. *Cluster Computing*, 22(3):829–838, 2019.
- [259] Felix Lau, Stuart H Rubin, Michael H Smith, and Ljiljana Trajkovic. Distributed denial of service attacks. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. cybernetics evolving to systems, humans, organizations, and their complex interactions* (cat. no. 0, volume 3, pages 2275–2280. IEEE, 2000.
- [260] S. Layeghy, F. Pakzad, and M. Portmann. Scor: Constraint programming-based northbound interface for sdn. In *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 83–88, Dec 2016.
- [261] S. Layeghy, F. Pakzad, M. Portmann, et al. A new qos routing northbound interface for sdn. *Journal of Telecommunications and the Digital Economy*, 5(1):92, 2017.
- [262] Duc C Le and Nur Zincir-Heywood. Exploring anomalous behaviour detection and classification for insider threat identification. *International Journal of Network Management*, 31(4):e2109, 2021.
- [263] Alessandro Ferreira Leite, Vander Alves, Genáina Nunes Rodrigues, Claude Tadonki, Christine Eisenbeis, and Alba Cristina Magalhaes Alves de Melo. Dohko: an autonomic system for provision, configuration, and management of inter-cloud environments based on a software product line engineering method. *Cluster Computing*, 20(3):1951–1976, 2017.
- [264] Leonardo Leite, Fabio Kon, Gustavo Pinto, and Paulo Meirelles. Platform teams: An organizational structure for continuous delivery. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 505–511, 2020.
- [265] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. A survey of devops concepts and challenges. *ACM Computing Surveys (CSUR)*, 52(6):1–35, 2019.
- [266] Valentina Lenarduzzi, Francesco Lomio, Nyyti Saarimäki, and Davide Taibi. Does migrating a monolithic system to microservices decrease the technical debt? *Journal of Systems and Software*, page 110710, 2020.
- [267] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

- [268] Nancy G Leveson. Software safety in embedded computer systems. *Communications of the ACM*, 34(2):34–46, 1991.
- [269] Hongda Li, Hongxin Hu, Guofei Gu, Gail-Joon Ahn, and Fuqiang Zhang. vnids: Towards elastic security with safe and efficient virtualization of network intrusion detection systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–34, 2018.
- [270] Jie Li, Jinshu Su, Rongmao Chen, Xiaofeng Wang, and Shuhui Chen. Practical privacy-preserving deep packet inspection outsourcing. *Concurrency and Computation: Practice and Experience*, 31(22):e4435, 2019.
- [271] Shanshan Li, Qianwen Xu, Peiyu Hou, Xiudi Chen, Yanze Wang, He Zhang, and Guoping Rong. Exploring the challenges of developing and operating consortium blockchains: A case study. In *Proceedings of the Evaluation and Assessment in Software Engineering*, pages 398–404. 2020.
- [272] Wubin Li, Yves Lemieux, Jing Gao, Zhuofeng Zhao, and Yanbo Han. Service mesh: Challenges, state of the art, and future research opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 122–1225. IEEE, 2019.
- [273] Y. Li, S. Sundaramurthy Chandran, A. Bardas G, X. Ou, D. Caragea, X. Hu, and J. Jang. Experimental study of fuzzy hashing in malware clustering analysis. In *8th Workshop on Cyber Security Experimentation and Test ({CSET} 15)*, 2015.
- [274] Zhi Li, Hai Jin, Deqing Zou, and Bin Yuan. Exploring new opportunities to defeat low-rate ddos attack in container-based cloud environment. *IEEE Transactions on Parallel and Distributed Systems*, 2019.
- [275] Xiubo Liang and Qian Zhao. On the design of a blockchain-based student quality assessment system. In *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, pages 1–7. IEEE, 2020.
- [276] Robin Lichtenthäler, Mike Precht, Christoph Schwill, Tobias Schwartz, Pascal Cezanne, and Guido Wirtz. Requirements for a model-driven cloud-native migration of monolithic web-based applications. *SICS Software-Intensive Cyber-Physical Systems*, pages 1–12, 2019.
- [277] Martin Forsberg Lie, Mary Sánchez-Gordón, and Ricardo Colomo-Palacios. Devops in an iso 13485 regulated environment: A multivocal literature review. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2020.

- [278] M. Line, O. Nordland, L. Røstad, and I. Tøndel. Safety vs security? In *PSAM Conference, New Orleans, USA*. sn, 2006.
- [279] Ping Liu, Haowen Xu, Qianyu Ouyang, Rui Jiao, Zhekang Chen, Shenglin Zhang, Jiahai Yang, Linlin Mo, Jice Zeng, Wenman Xue, et al. Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 48–58. IEEE, 2020.
- [280] Y. Liu, Z. Xia, P. Yi, Y. Yao, T. Xie, W. Wang, and T. Zhu. Genpass: A general deep learning model for password guessing with pcfg rules and adversarial generation. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.
- [281] Van N. Adrichem LM, C. Doerr, and F. Kuipers. Opennetmon: Network monitoring in openflow software-defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8. IEEE, 2014.
- [282] D Lopez, E Lopez, Linda Dunbar, John Strassner, and Rakesh Kumar. Framework for interface to network security functions. *draft-ietf-i2nsf-framework-04. pdf, IETF I2NSF WG*, 2018.
- [283] Ping Lou, Guantong Lu, Xuemei Jiang, Zheng Xiao, Jiwei Hu, and Junwei Yan. Cyber intrusion detection through association rule mining on multi-source logs. *Applied Intelligence*, pages 1–15, 2020.
- [284] Duo Lu, Dijiang Huang, Andrew Walenstein, and Deep Medhi. A secure microservice framework for iot. In *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 9–18. IEEE, 2017.
- [285] Qinghua Lu, An Binh Tran, Ingo Weber, Hugo O’Connor, Paul Rimba, Xiwei Xu, Mark Staples, Liming Zhu, and Ross Jeffery. Integrated model-driven engineering of blockchain applications for business processes and asset management. *Software: Practice and Experience*, 51(5):1059–1079, 2021.
- [286] Andriy Luntovskyy and Bohdan Shubyn. Highly-distributed systems based on micro-services and their construction paradigms. In *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, pages 7–14. IEEE, 2020.
- [287] Xiaohui Luo, Fengyuan Ren, and Tong Zhang. High performance userspace networking for containerized microservices. In *International Conference on Service-Oriented Computing*, pages 57–72. Springer, 2018.

- [288] Duc-H. Luong, A. Outtagarts, and A. Hebbar. Traffic monitoring in software defined networks using opendaylight controller. In *International Conference on Mobile, Secure, and Programmable Networking*, pages 38–48. Springer, 2016.
- [289] Lucy Ellen Lwakatare, Terhi Kilamo, Teemu Karvonen, Tanja Sauvola, Ville Heikkilä, Juha Itkonen, Pasi Kuvaja, Tommi Mikkonen, Markku Oivo, and Casper Lassenius. Devops in practice: A multiple case study of five companies. *Information and Software Technology*, 114:217–230, 2019.
- [290] Olav Lysne, Kjell J Hole, Christian Otterstad, Øyvind Ytrehus, Raymond Aarseth, and Jørgen Tellnes. Vendor malware: detection limits and mitigation. *Computer*, 49(8):62–69, 2016.
- [291] E. Scheidt M, E. Domanque, R. Butler, and W. Tsang. Access system utilizing multiple factor identification and authentication, February 13 2007. US Patent 7,178,025.
- [292] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. Automap: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference 2020*, pages 246–258, 2020.
- [293] Bouchera Maati and Djamel Eddine Saidouni. Ciotas protocol: Cloudiot available services protocol through autonomic computing against distributed denial of services attacks. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–30, 2020.
- [294] Zoltán Adám Mann. Secure software placement and configuration. *Future Generation Computer Systems*, 110:243–253, 2020.
- [295] Steve Mansfield-Devine. Devops: finding room for security. *Network Security*, 2018(7):15–20, 2018.
- [296] AR Manu, Jitendra Kumar Patel, Shakil Akhtar, VK Agrawal, and KN Bala Subramanya Murthy. Docker container security via heuristics-based multilateral security-conceptual and pragmatic study. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–14. IEEE, 2016.
- [297] G. Manunta. What is security? *Security Journal*, 12:57–66, 1999.
- [298] Xavier Marchal, Thibault Cholez, and Olivier Festor. μ ndn: an orchestrated microservice architecture for named data networking. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, pages 12–23, 2018.

- [299] Gastón Márquez and Hernán Astudillo. Identifying availability tactics to support security architectural design of microservice-based systems. In *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, pages 123–129, 2019.
- [300] Edward M Marszal and Eric William Scharpf. *Safety Integrity Level Selection*. ISA, the Instrumentation, Systems, and Automation Society, 2002.
- [301] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.
- [302] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár. Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, 99:48–61, 2017.
- [303] A. Melis, D. Berardi, C. Contoli, F. Callegati, F. Esposito, and M. Prandini. A policy checker approach for secure industrial sdn. In *2018 2nd Cyber Security in Networking Conference (CSNet)*, pages 1–7. IEEE, 2018.
- [304] Andrea Melis, Davide Berardi, Chiara Contoli, Franco Callegati, Flavio Esposito, and Marco Prandini. A policy checker approach for secure industrial sdn. In *2018 2nd Cyber Security in Networking Conference (CSNet)*, pages 1–7, Oct 2018.
- [305] Andrea Melis, Silvia Mirri, Catia Prandi, Marco Prandini, Paola Salomoni, and Franco Callegati. Integrating personalized and accessible itineraries in maas ecosystems through microservices. *Mobile Networks and Applications*, 23(1):167–176, 2018.
- [306] Rebecca T Mercuri and Peter G Neumann. Security by obscurity. *Communications of the ACM*, 46(11):160, 2003.
- [307] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu. Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 15–28, 2017.
- [308] Microsemi. Ieee 1588 precise time protocol: The new standard in time synchronization. 2017.
- [309] Jennifer U Mills, Jason R Dever, and Steven MF Stuban. Using regression to predict potential insider threats. *Defense AR Journal*, 25(2):122–157, 2018.
- [310] Alok Mishra and Ziadoon Otaiwi. Devops and software quality: A systematic mapping. *Computer Science Review*, 38:100308, 2020.

- [311] M. Mitzenmacher. Compressed bloom filters. *IEEE/ACM transactions on networking*, 10(5):604–612, 2002.
- [312] Tal Mizrahi. Time synchronization security using ipsec and macsec. In *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pages 38–43. IEEE, 2011.
- [313] Mustafa Abshir Mohamed, Moharram Challenger, and Geylani Kardas. Applications of model-driven engineering in cyber-physical systems: a systematic mapping study. *Journal of Computer Languages*, 59:100972, 2020.
- [314] Tareq Abed Mohammed and Ahmed Burhan Mohammed. Security architectures for sensitive data in cloud computing. In *Proceedings of the 6th International Conference on Engineering & MIS 2020*, pages 1–6, 2020.
- [315] V. Mohan, YR. Reddy, and K. Kalpana. Active and passive network measurements: a survey. *International Journal of Computer Science and Information Technologies*, 2(4):1372–1385, 2011.
- [316] Ahmad Mohsin and Naeem Khalid Janjua. A review and future directions of soa-based software architecture modeling approaches for system of systems. *Service Oriented Computing and Applications*, 12(3-4):183–200, 2018.
- [317] Fabrizio Montesi and Janine Weber. From the decorator pattern to circuit breakers in microservices. In Hisham M. Haddad, Roger L. Wainwright, and Richard Chbeir, editors, *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018*, pages 1733–1735. ACM, 2018.
- [318] João Bourbon Moreira, Henrique Mamede, Vasco Pereira, and Bruno Sousa. Next generation of microservices for the 5g service-based architecture. *International Journal of Network Management*, 30(6):e2132, 2020.
- [319] James B Morris. 10 rules for an unhackable data vault. *Ubiquity*, 2017(May):1–10, 2017.
- [320] Jose Moura and David Hutchison. Fog computing systems: State of the art, research issues and future trends, with a focus on resilience. *Journal of Network and Computer Applications*, page 102784, 2020.
- [321] Bassam Moussa, Mourad Debbabi, and Chadi Assi. A detection and mitigation model for ptp delay attack in an iec 61850 substation. *IEEE Transactions on Smart Grid*, 9(5):3954–3965, 2016.

- [322] Bassam Moussa, Marthe Kassouf, Rachid Hadjidj, Mourad Debbabi, and Chadi Assi. An extension to the precision time protocol (ptp) to enable the detection of cyber attacks. *IEEE Transactions on Industrial Informatics*, 2019.
- [323] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.
- [324] M. Munir, S. Siddiqui Ahmed, M. Chattha Ali, A. Dengel, and S. Ahmed. Fusead: unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. *Sensors*, 19(11):2451, 2019.
- [325] Vasudevan Nagendra, Vinod Yegneswaran, Phillip Porras, and Samir R Das. Coordinated dataflow protection for ultra-high bandwidth science networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 568–583, 2019.
- [326] Deeraj Nagothu, Ronghua Xu, Seyed Yahya Nikouei, and Yu Chen. A microservice-enabled architecture for smart surveillance using blockchain technology. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–4. IEEE, 2018.
- [327] A. Nasrallah, A. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury. Ultra-low latency (ull) networks: The iee tsn and ietf detnet standards and related 5g ull research. *IEEE Communications Surveys & Tutorials*, 21(1):88–145, 2018.
- [328] Antonio Nehme, Vitor Jesus, Khaled Mahbub, and Ali Abdallah. Fine-grained access control for microservices. In *International Symposium on Foundations and Practice of Security*, pages 285–300. Springer, 2018.
- [329] Antonio Nehme, Vitor Jesus, Khaled Mahbub, and Ali Abdallah. Securing microservices. *IT Professional*, 21(1):42–49, 2019.
- [330] M. Nergiz, M. Atzori, and C. Clifton. Hiding the presence of individuals from shared databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 665–676, 2007.
- [331] N. Nethercote, P. Stuckey J, R. Becket, S. Brand, G. Duck J, and G. Tack. Minizinc: Towards a standard cp modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.
- [332] Johannes Neyer, Lukas Gassner, and Cristian Marinescu. Redundant schemes or how to counter the delay attack on time synchronization protocols. In *2019 IEEE International*

- Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 1–6. IEEE, 2019.
- [333] Quy Nguyen and Oras Baker. Applying spring security framework and oauth2 to protect microservice architecture api. *Journal of Software*, pages 257–264, 2019.
- [334] Mahmood Niazi, Alok Mishra, and Asif Qumer Gill. What do software practitioners really think about software process improvement project success? an exploratory study. *Arabian Journal for Science and Engineering*, 43(12):7719–7735, 2018.
- [335] F. Niedermeyer, S. Steinmetzer, M. Kroll, and Schnell Rainer. Cryptanalysis of basic bloom filters used for privacy preserving record linkage. *German Record Linkage Center, Working Paper Series, No. WP-GRLC-2014-04*, 2014.
- [336] Naghmeh Niknejad, Waidah Ismail, Imran Ghani, Behzad Nazari, Mahadi Bahari, et al. Understanding service-oriented architecture (soa): A systematic literature review and directions for further investigation. *Information Systems*, 91:101491, 2020.
- [337] N Nikolakis, A Marguglio, G Veneziano, P Greco, S Panicucci, T Cerquitelli, E Macii, S Andolina, and K Alexopoulos. A microservice architecture for predictive analytics in manufacturing. *Procedia Manufacturing*, 51:1091–1097, 2020.
- [338] Yannis Nikoloudakis, Evangelos Pallis, George Mastorakis, Constandinos X Mavromoustakis, Charalabos Skianis, and Evangelos K Markakis. Vulnerability assessment as a service for fog-centric ict ecosystems: A healthcare use case. *Peer-to-Peer Networking and Applications*, 12(5):1216–1224, 2019.
- [339] Seyed Yahya Nikouei, Yu Chen, Alexander Aved, Erik Blasch, and Timothy R Faughnan. I-safe: Instant suspicious activity identification at the edge using fuzzy decision making. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 101–112, 2019.
- [340] NIST. Cybersecurity framework, 2020.
- [341] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu. Using of jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 380–384, 2013.
- [342] Peter Nkomo and Marijke Coetzee. Development activities, tools and techniques of secure microservices compositions. In *International Conference on Information Security Practice and Experience*, pages 423–433. Springer, 2019.

- [343] Mahda Noura, Mohammed Atiquzzaman, and Martin Gaedke. Interoperability in internet of things: Taxonomies and open challenges. *Mobile Networks and Applications*, 24(3):796–809, 2019.
- [344] Mitchell Olsthoorn, Arie van Deursen, and Annibale Panichella. Generating highly-structured input data by combining search-based testing and grammar-based fuzzing. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1224–1228. IEEE, 2020.
- [345] Alexander Oppermann, Federico Grasso Toro, Florian Thiel, and Jean-Pierre Seifert. Secure cloud computing: Reference architecture for measuring instrument under legal control. *Security and Privacy*, 1(3):e18, 2018.
- [346] Amr Osman, Pascal Bruckner, Hani Salah, Frank HP Fitzek, Thorsten Strufe, and Mathias Fischer. Sandnet: Towards high quality of deception in container-based microservice architectures. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.
- [347] Amr Osman, Simon Hanisch, and Thorsten Strufe. Seconetbench: A modular framework for secure container networking benchmarks. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 21–28. IEEE, 2019.
- [348] Christian Otterstad and Tetiana Yarygina. Low-level exploitation mitigation by diverse microservices. In *European Conference on Service-Oriented and Cloud Computing*, pages 49–56. Springer, 2017.
- [349] OWASP Foundation. Open web application security project (OWASP) Application Threat Modeling. https://owasp.org/www-community/Application_Threat_Modeling, Nov. 2020.
- [350] Wojciech Owczarek et al. The ptpd project, 2015.
- [351] Marc-Oliver Pahl and François-Xavier Aubet. All eyes on you: Distributed multi-dimensional iot microservice anomaly detection. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 72–80. IEEE, 2018.
- [352] Marc-Oliver Pahl, François-Xavier Aubet, and Stefan Liebald. Graph-based iot microservice security. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–3. IEEE, 2018.

- [353] Marc-Oliver Pahl and Lorenzo Donini. Securing iot microservices with certificates. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5. IEEE, 2018.
- [354] Nicolae Paladi, Antonis Michalas, and Hai-Van Dang. Towards secure cloud orchestration for multi-cloud deployments. In *Proceedings of the 5th Workshop on CrossCloud Infrastructures & Platforms*, pages 1–6, 2018.
- [355] Yohanes Yohanie Fridelin Panduman, Sritrusta Sukaridhoto, and Anang Tjahjono. A survey of iot platform comparison for building cyber-physical system architecture. In *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 238–243. IEEE, 2019.
- [356] Eunsoo Park and Kyungho Jeon. Secure volume hot-plugging for containers (industry track). In *Proceedings of the 1st International Middleware Conference Industrial Track*, pages 38–44, 2020.
- [357] Adrian Paschke. Provalets: Component-based mobile agents as microservices for rule-based data access, processing and analytics. *Business & Information Systems Engineering*, 58(5):329–340, 2016.
- [358] K. Pentikousis, Y. Wang, and W. Hu. Mobileflow: Toward software-defined mobile networks. *IEEE Communications Magazine*, 51(7):44–53, 2013.
- [359] G Perrone and Simon Pietro Romano. The docker security playground: A hands-on approach to the study of network security. In *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, pages 1–8. IEEE, 2017.
- [360] Jovana Petrovska, Agon Memeti, and Florinda Imeri. Soa approach-identity and access management for the risk management platform. In *2019 8th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4. IEEE, 2019.
- [361] Andrea M Plaza, Jessica Díaz, and Jennifer Pérez. Software architectures for health care cyber-physical systems: A systematic literature review. *Journal of Software: Evolution and Process*, 30(7):e1930, 2018.
- [362] Gordon D. Plotkin, Nikolaj Bjørner, Nuno P. Lopes, Andrey Rybalchenko, and George Varghese. Scaling network verification using symmetry and surgery. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL ’16, pages 69–83, New York, NY, USA, 2016. ACM.

- [363] Francisco Ponce, Jacopo Soldani, Hernán Astudillo, and Antonio Brogi. Smells and refactorings for microservices security: A multivocal literature review, 2021.
- [364] Fabien Pouget, Marc Dacier, and Hervé Debar. White paper: honeypot, honeynet, honeytoken: terminological issues. *Rapport technique EURECOM*, 1275, 2003.
- [365] Catia Prandi, Andrea Melis, Marco Prandini, Giovanni Delnevo, Lorenzo Monti, Silvia Mirri, and Paola Salomoni. Gamifying cultural experiences across the urban environment. *Multi-media Tools and Applications*, 78(3):3341–3364, 2019.
- [366] Davy Preuveneers and Wouter Joosen. Access control with delegated authorization policy evaluation for data-driven microservice workflows. *Future Internet*, 9(4):58, 2017.
- [367] Davy Preuveneers and Wouter Joosen. Towards multi-party policy-based access control in federations of cloud and edge microservices. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 29–38. IEEE, 2019.
- [368] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)*, 19(2):1–41, 2019.
- [369] Navid Pustchi, Ram Krishnan, and Ravi Sandhu. Authorization federation in iaas multi cloud. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, pages 63–71, 2015.
- [370] W. Queiroz, M. Capretz, and M. Dantas. An approach for sdn traffic monitoring based on big data techniques. *Journal of Network and Computer Applications*, 131:28–39, 2019.
- [371] W. Queiroz, M. Capretz, and M. Dantas. An approach for sdn traffic monitoring based on big data techniques. *Journal of Network and Computer Applications*, 131:28 – 39, 2019.
- [372] J. Raghavendran and JA. Schormans. Inferring delay variations using packet-pair probing techniques for network measurement.
- [373] Isuru Ranawaka, Suresh Marru, Juleen Graham, Aarushi Bisht, Jim Basney, Terry Fleury, Jeff Gaynor, Dimuthu Wannipurage, Marcus Christie, Alexandru Mahmoud, et al. Custos: Security middleware for science gateways. In *Practice and Experience in Advanced Research Computing*, pages 278–284. 2020.
- [374] Alireza Ranjbar, Miika Komu, Patrik Salmela, and Tuomas Aura. Synaptic: Secure and persistent connectivity for containers. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 262–267. IEEE, 2017.

- [375] T Ramalingeswara Rao, Pabitra Mitra, Ravindara Bhatt, and A Goswami. The big data system, components, tools, and technologies: a survey. *Knowledge and Information Systems*, pages 1–81, 2018.
- [376] Aruna Ravichandran, Kieran Taylor, and Peter Waterhouse. *DevOps for Digital Leaders*. CA, 2016.
- [377] Mohammadreza Razian, Mohammad Fathian, and Rajkumar Buyya. Arc: Anomaly-aware robust cloud-integrated iot service composition based on uncertainty in advertised quality of service values. *Journal of Systems and Software*, 164:110557, 2020.
- [378] Abdul Razzaq. A systematic review on software architectures for iot systems and future direction to the adoption of microservices architecture. *SN Computer Science*, 1(6):1–30, 2020.
- [379] AJH Redelinghuys, AH Basson, and K Kruger. A six-layer architecture for the digital twin: a manufacturing case study implementation. *Journal of Intelligent Manufacturing*, pages 1–20, 2019.
- [380] J Paul Reed. Beyond the ‘fix-it’ treadmill. *Communications of the ACM*, 63(5):58–63, 2020.
- [381] Hao Ren, Hongwei Li, Dongxiao Liu, Guowen Xu, Nan Cheng, and Xuemin Sherman Shen. Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox. *IEEE Transactions on Cloud Computing*, 2020.
- [382] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, 88:173–190, 2018.
- [383] D. Manalu Robinson, E. Rajagukguk, R. Siringoringo, D. Siahaan Kristanto, and P. Sihombing. The development of document similarity detector by jaccard formulation. In *2019 International Conference of Computer Science and Information Technology (ICoSNIKOM)*, pages 1–4. IEEE, 2019.
- [384] Surya Roca, Jorge Sancho, José García, and Álvaro Alesanco. Microservice chatbot architecture for chronic patient support. *Journal of biomedical informatics*, 102:103305, 2020.
- [385] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*, volume 1. Elsevier, UK, 2006.
- [386] Hang Ruan, Bihuan Chen, Xin Peng, and Wenyun Zhao. Deeplink: Recovering issue-commit links based on deep learning. *Journal of Systems and Software*, 158:110406, 2019.

- [387] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering - Guidelines and Examples*. Wiley, 2012.
- [388] Mark Russinovich, Manuel Costa, Cédric Fournet, David Chisnall, Antoine Delignat-Lavaud, Sylvan Clebsch, Kapil Vaswani, and Vikas Bhatia. Toward confidential cloud computing: Extending hardware-enforced cryptographic protection to data while in use. *Queue*, 19(1):49–76, 2021.
- [389] Olga Safaryan, Elena Pinevich, Evgenia Roshchina, Larissa Cherckesova, and Nadezhda Kolennikova. Information system development for restricting access to software tool built on microservice architecture. In *E3S Web of Conferences*, volume 224. EDP Sciences, 2020.
- [390] Jyothi Salibindla. Microservices api security. *International Journal of Engineering Research & Technology*, 7(1):277–281, 2018.
- [391] Davide Salomoni, Isabel Campos, Luciano Gaido, J Marco de Lucas, P Solagna, Jorge Gomes, Ludek Matyska, P Fuhrman, Marcus Hardt, Giacinto Donvito, et al. Indigo-datacloud: a platform to facilitate seamless access to e-infrastructures. *Journal of Grid Computing*, 16(3):381–408, 2018.
- [392] Ameya Sanzgiri and Dipankar Dasgupta. Classification of insider threat detection techniques. In *Proceedings of the 11th annual cyber and information security research conference*, pages 1–4, 2016.
- [393] M Subrahmanya Sarma, Y Srinivas, M Abhiram, Lakshminarayana Ullala, M Sahithi Prasanthi, and J Rojee Rao. Insider threat detection with face recognition and knn user classification. In *CCEM 2017*, pages 39–44. IEEE, 2017.
- [394] Theo Schlossnagle. Monitoring in a devops world. *Queue*, 15(6):35–45, 2017.
- [395] Theo Schlossnagle. Monitoring in a devops world. *Communications of the ACM*, 61(3):58–61, 2018.
- [396] B. Schneier. Two-factor authentication: too little, too late. *Communications of the ACM*, 48(4):136, 2005.
- [397] Bruce Schneier. *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2015.
- [398] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using bloom filters. *BMC medical informatics and decision making*, 9(1):41, 2009.

- [399] C. Schulte, G. Tack, and M. Lagerkvist. Modeling and programming with gecode. *Schulte, Christian and Tack, Guido and Lagerkvist, Mikael*, 1, 2010.
- [400] Sandra Scott-Hayward, Gemma O’Callaghan, and Sakir Sezer. Sdn security: A survey. In *2013 IEEE SDN For Future Networks and Services (SDN4FNS)*, pages 1–7. IEEE, 2013.
- [401] S. Seeber, L. Stiemert, and G. Rodosek. Towards an sdn-enabled ids environment. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 751–752. IEEE, 2015.
- [402] Mojtaba Shahin, Mansooreh Zahedi, Muhammad Ali Babar, and Liming Zhu. An empirical study of architecting for continuous delivery and deployment. *Empirical Software Engineering*, 24(3):1061–1108, 2019.
- [403] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky. Advanced study of sdn/openflow controllers. In *Proceedings of the 9th central & eastern european software engineering conference in russia*, pages 1–6, 2013.
- [404] Priyanka Sharma, Sebastian Lawrenz, and Andreas Rausch. Towards trustworthy and independent data marketplaces. In *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, pages 39–45, 2020.
- [405] Ezzeldin Shereen, Florian Bitard, György Dán, Tolga Sel, and Steffen Fries. Next steps in security for time synchronization: Experiences from implementing ieee 1588 v2. 1. In *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 1–6. IEEE, 2019.
- [406] Sajad Shirali-Shahreza and Yashar Ganjali. Flexam: Flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 167–168, 2013.
- [407] Yang ShuLin and Hu JiePing. Research on unified authentication and authorization in microservice architecture. In *2020 IEEE 20th International Conference on Communication Technology (ICCT)*, pages 1169–1173. IEEE, 2020.
- [408] Gion Sialm and Silvia Knittl. Bring your own identity-case study from the swiss government. In *Annual Privacy Forum*, pages 38–47. Springer, 2016.
- [409] Alex Xandra Albert Sim, Okky Putra Barus, and Frandy Jaya. Lessons learned in applying reactive system in microservices. In *Journal of Physics: Conference Series*, volume 1175, page 012101. IOP Publishing, 2019.

- [410] Y. Sinha, S. Vashishth, and K. Haribabu. Real time monitoring of packet loss in software defined networks. In *International Conference on Ubiquitous Communications and Network Computing*, pages 154–164. Springer, 2017.
- [411] Hannah Snyder. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104:333 – 339, 2019.
- [412] Jacopo Soldani. Grey literature: A safe bridge between academy and industry? *ACM SIGSOFT Softw. Eng. Notes*, 44(3):11–12, 2019.
- [413] Jacopo Soldani, Damian Andrew Tamburri, and Willem-Jan Van Den Heuvel. The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146:215–232, 2018.
- [414] S. Song. Improving network health monitoring accuracy based on data fusion for software defined networking. In *Future Information Technology*, pages 469–472. Springer, 2014.
- [415] Murugiah Souppaya, John Morello, and Karen Scarfone. Application container security guide (2nd draft). Technical report, National Institute of Standards and Technology, 2017.
- [416] Dimitri Michel Stallenberg and Annibale Panichella. Jcomix: a search-based tool to detect xml injection vulnerabilities in web applications. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1090–1094, 2019.
- [417] Vasilis Stavrou, Miltiadis Kandias, Georgios Karoulas, and Dimitris Gritzalis. Business process modeling for insider threat monitoring and handling. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 119–131. Springer, 2014.
- [418] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.
- [419] James Michael Stewart, Mike Chapple, and Darril Gibson. *CISSP: Certified Information Systems Security Professional Study Guide*. John Wiley & Sons, 2012.
- [420] E. Stobert and R. Biddle. The password life cycle: user behaviour in managing passwords. In *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*, pages 243–255, 2014.
- [421] Daniel Stock, Daniel Schel, and Thomas Bauernhansl. Middleware-based cyber-physical production system modeling for operators. *Procedia Manufacturing*, 42:111–118, 2020.

- [422] Mirko Stocker, Olaf Zimmermann, Uwe Zdun, Daniel Lübke, and Cesare Pautasso. Interface quality patterns: Communicating and improving the quality of microservices apis. In *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, pages 1–16, 2018.
- [423] Salvatore Stolfo. Simulated user bots: Real time testing of insider threat detection systems. 2018.
- [424] Sari Sultan, Imtiaz Ahmad, and Tassos Dimitriou. Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7:52976–52996, 2019.
- [425] Yuqiong Sun, Susanta Nanda, and Trent Jaeger. Security-as-a-service for microservices-based cloud applications. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 50–57. IEEE, 2015.
- [426] Anders Sundelin, Javier Gonzalez-Huerta, and Krzysztof Wnuk. The hidden cost of backward compatibility: when deprecation turns into technical debt-an experience report. In *Proceedings of the 3rd International Conference on Technical Debt*, pages 67–76, 2020.
- [427] Sahil Suneja, Ali Kanso, and Canturk Isci. Can container fusion be securely achieved? In *Proceedings of the 5th International Workshop on Container Technologies and Container Clouds*, pages 31–36, 2019.
- [428] Nico Surantha and Felix Ivan. Secure kubernetes networking design based on zero trust model: A case study of financial service enterprise in indonesia. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 348–361. Springer, 2019.
- [429] Madiha H Syed and Eduardo B Fernandez. The container manager pattern. In *Proceedings of the 22nd European Conference on Pattern Languages of Programs*, pages 1–9, 2017.
- [430] Madiha H Syed and Eduardo B Fernandez. A reference architecture for the container ecosystem. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–6, 2018.
- [431] Mohammad Bany Taha, Chamseddine Talhi, and Hakima Ould-Slimanec. A cluster of cpabe microservices for vanet. *Procedia Computer Science*, 155:441 – 448, 2019. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019), The 14th International Conference on Future Networks and Communications (FNC-2019), The 9th International Conference on Sustainable Energy Information Technology.

- [432] Salman Taherizadeh and Marko Grobelnik. Key influencing factors of the kubernetes auto-scaler for computing-intensive microservice-native cloud-based applications. *Advances in Engineering Software*, 140:102734, 2020.
- [433] Dmitri Tchoubraev and Daniel Wiczynski. Swiss tso integrated operational planning, optimization and ancillary services system. In *2015 IEEE Eindhoven PowerTech*, pages 1–6. IEEE, 2015.
- [434] Tihomir Tenev and Simeon Tsvetanov. Recommendations for enhancing security in microservice environment altered in an intelligent way. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE, 2020.
- [435] Tran Quang Thanh, Stefan Covaci, Thomas Magedanz, Panagiotis Gouvas, and Anastasios Zafeiropoulos. Embedding security and privacy into the development and operation of cloud applications and services. In *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pages 31–36. IEEE, 2016.
- [436] Johannes Thönes. Microservices. *IEEE software*, 32(1):116–116, 2015.
- [437] C. Thorpe, C. Olariu, A. Hava, and P. McDonagh. Experience of developing an openflow sdn prototype for managing iptv networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 966–971. IEEE, 2015.
- [438] Kleanthis Thramboulidis, Danai C Vachtsevanou, and Ioanna Kontou. Cpus-iot: A cyber-physical microservice and iot-based framework for manufacturing assembly systems. *Annual Reviews in Control*, 47:237–248, 2019.
- [439] Chin-Wei Tien, Tse-Yung Huang, Chia-Wei Tien, Ting-Chun Huang, and Sy-Yen Kuo. Kubanomaly: Anomaly detection for the docker orchestration platform with neural network approaches. *Engineering Reports*, page e12080, 2019.
- [440] Kennedy A Torkura, Muhammad IH Sukmana, Feng Cheng, and Christoph Meinel. Leveraging cloud native design patterns for security-as-a-service applications. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 90–97. IEEE, 2017.
- [441] Kennedy A Torkura, Muhammad IH Sukmana, and Anne VDM Kayem. A cyber risk based moving target defense mechanism for microservice architectures. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 932–939. IEEE, 2018.

- [442] Kennedy A Torkura, Muhammad IH Sukmana, and Christoph Meinel. Integrating continuous security assessments in microservices and cloud native applications. In *Proceedings of the 10th International Conference on Utility and Cloud Computing*, pages 171–180, 2017.
- [443] Reza Tourani, Austin Bos, Satyajayant Misra, and Flavio Esposito. Towards security-as-a-service in multi-access edge. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 358–363, 2019.
- [444] Demetris Trihinas, Athanasios Tryfonos, and Marios D Dikaiakos. Designing scalable and secure microservices by embracing devops-as-a-service offerings. 2016.
- [445] Demetris Trihinas, Athanasios Tryfonos, Marios D Dikaiakos, and George Pallis. Devops as a service: Pushing the boundaries of microservice adoption. *IEEE Internet Computing*, 22(3):65–71, 2018.
- [446] Michal Trnka, Tom Černý, and Nathaniel Stickney. Survey of authentication and authorization for the internet of things. *Security and Communication Networks*, 2018:1–17, 06 2018.
- [447] Ernesto Troiano, John Soldatos, Ariana Polyviou, Andreas Polyviou, Alessandro Mamelli, and Dimitris Drakoulis. Big data platform for integrated cyber and physical security of critical infrastructures for the financial sector: Critical infrastructures as cyber-physical systems. In *Proceedings of the 11th International Conference on Management of Digital EcoSystems*, pages 262–269, 2019.
- [448] Catia Trubiani, Alexander Bran, André van Hoorn, Alberto Avritzer, and Holger Knoche. Exploiting load testing and profiling for performance antipattern detection. *Information and Software Technology*, 95:329–345, 2018.
- [449] Hong-Linh Truong and Peter Klein. Devops contract for assuring execution of iot microservices in the edge. *Internet of Things*, 9:100150, 2020.
- [450] Katja Tuma, Laurens Sion, Riccardo Scandariato, and Koen Yskout. Automating the early detection of security design flaws. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 332–342, 2020.
- [451] Tony UcedaVelez and Marco M Morana. *Risk centric threat modeling*. Wiley Online Library, 2015.
- [452] Markus Ullmann and Matthias Vögeler. Delay attacks—implication on ntp and ptp time synchronization. In *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pages 1–6. IEEE, 2009.

- [453] Sricharan Vadapalli. *DevOps: continuous delivery, integration, and deployment with DevOps: dive into the core DevOps strategies*. Packt Publishing Ltd, 2018.
- [454] Anelis Pereira Vale, Gaston Marquez, and Astudillo. Security mechanisms used in microservices-based systems: A systematic mapping. 2019.
- [455] Anelis Pereira Vale, Gastón Márquez, Hernán Astudillo, and Eduardo B Fernandez. Security mechanisms used in microservices-based systems: A systematic mapping. In *CLEI*, pages 1–10, 2019.
- [456] Nees Van Eck and Ludo Waltman. Software survey: Vosviewer, a computer program for bibliometric mapping. *scientometrics*, 84(2):523–538, 2010.
- [457] Luis M Vaquero, Felix Cuadrado, Yehia Elkhatib, Jorge Bernal-Bernabe, Satish N Srirama, and Mohamed Faten Zhani. Research challenges in nextgen service orchestration. *Future Generation Computer Systems*, 90:20–38, 2019.
- [458] Blesson Varghese and Rajkumar Buyya. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861, 2018.
- [459] Vassilios Vassilakis, Emmanouil Panaousis, and Haralambos Mouratidis. Security challenges of small cell as a service in virtualized mobile edge computing environments. In *IFIP International Conference on Information Security Theory and Practice*, pages 70–84. Springer, 2016.
- [460] D. Vatsalan, Z. Sehili, P. Christen, and E. Rahm. Privacy-preserving record linkage for big data: Current approaches and research challenges. In *Handbook of Big Data Technologies*, pages 851–895. Springer, 2017.
- [461] Julien Vehent. *Securing DevOps: Security in the Cloud*. Manning Publications Co., 2018.
- [462] P. Voigt and A. Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed.*, Cham: Springer International Publishing, 2017.
- [463] Hulya Vural, Murat Koyuncu, and Sinem Guney. A systematic literature review on microservices. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Giuseppe Borruso, Carmelo M. Torre, Ana Maria A.C. Rocha, David Taniar, Bernady O. Apduhan, Elena Stankova, and Alfredo Cuzzocrea, editors, *Computational Science and Its Applications – ICCSA 2017*, pages 203–217, Cham, 2017. Springer International Publishing.
- [464] Andrew Walker and Tomas Cerny. On cloud computing infrastructure for existing code-clone detection algorithms. *ACM SIGAPP Applied Computing Review*, 20(1):5–14, 2020.

- [465] Kevin Walsh and John Manferdelli. Mechanisms for mutual attested microservice communication. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, pages 59–64, 2017.
- [466] Lingzhi Wang, Nengwen Zhao, Junjie Chen, Pinnong Li, Wenchi Zhang, and Kaixin Sui. Root-cause metric location for microservice systems via log anomaly detection. In *2020 IEEE International Conference on Web Services (ICWS)*, pages 142–150. IEEE, 2020.
- [467] Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. Cloudranger: root cause identification for cloud native systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 492–502. IEEE, 2018.
- [468] Muhammad Waseem, Peng Liang, and Mojtaba Shahin. A systematic mapping study on microservices architecture in devops. *Journal of Systems and Software*, 170:110798, 2020.
- [469] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang. Comparison of open-source cloud management platforms: Openstack and opennebula. In *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 2457–2461. IEEE, 2012.
- [470] Zhenyu Wen, Tao Lin, Renyu Yang, Shouling Ji, Rajiv Ranjan, Alexander Romanovsky, Changting Lin, and Jie Xu. Ga-par: Dependable microservice orchestration framework for geo-distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, 31(1):129–143, 2019.
- [471] Magnus Westerlund and Nane Kratzke. Towards distributed clouds: A review about the evolution of centralized cloud computing, distributed ledger technologies, and a foresight on unifying opportunities and security implications. In *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pages 655–663. IEEE, 2018.
- [472] Nils Wieber. Automated generation of client-specific backends utilizing existing microservices and architectural knowledge. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1158–1160. IEEE, 2020.
- [473] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10, 2014.
- [474] S. Shin Won, P. Porras, V. Yegneswara, M. Fong, G. Gu, M. Tyson, et al. Fresco: Modular composable security services for software-defined networks. In *20th Annual Network & Distributed System Security Symposium*. Nds, 2013.

- [475] C. Wood. Constructing difficult-to-guess passwords. *Information Management & Computer Security*, 1996.
- [476] Xiaochun Wu, Kaiyu Hou, Xue Leng, Xing Li, YinBo Yu, Bo Wu, and Yan Chen. State of the art and research challenges in the security technologies of network function virtualization. *IEEE Internet Computing*, 2019.
- [477] Jacob Wurm, Khoa Hoang, Orlando Arias, Ahmad-Reza Sadeghi, and Yier Jin. Security analysis on consumer and industrial iot devices. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 519–524. IEEE, 2016.
- [478] Kim Wuyts, Dimitri Van Landuyt, Aram Hovsepyan, and Wouter Joosen. Effective and efficient privacy threat modeling through domain refinements. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1175–1178, 2018.
- [479] Binhuai Xu and Jing Bian. A cloud robotic application platform design based on the microservices architecture. In *2020 International Conference on Control, Robotics and Intelligent System*, pages 13–18, 2020.
- [480] Ronghua Xu, Seyed Yahya Nikouei, Yu Chen, Erik Blasch, and Alexander Aved. Blendmas: A blockchain-enabled decentralized microservices architecture for smart public safety. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 564–571. IEEE, 2019.
- [481] Rongxu Xu, Wenquan Jin, and Dohyeun Kim. Microservice security agent based on api gateway in edge computing. *Sensors*, 19(22):4905, 2019.
- [482] W. Xue, D. Vatsalan, W. Hu, and Seneviratne Aruna. Sequence data matching and beyond: New privacy-preserving primitives based on bloom filters. *IEEE Transactions on Information Forensics and Security*, 15:2973–2987, 2020.
- [483] Tarun Yadav and Arvind Mallari Rao. Technical aspects of cyber kill chain. In *International Symposium on Security in Computing and Communication*, pages 438–452. Springer, 2015.
- [484] H. Yang and S. S. Lam. Real-time verification of network properties using atomic predicates. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–11, Oct 2013.
- [485] H. Yang and S. S. Lam. Scalable verification of networks with packet transformers using atomic predicates. *IEEE/ACM Transactions on Networking*, 25(5):2900–2915, Oct 2017.
- [486] Hongkun Yang and Simon S. Lam. Networking research lab website. <http://www.cs.utexas.edu/users/lam/NRL/>.

- [487] Hongkun Yang and Simon S. Lam. Real-time verification of network properties using atomic predicates. *IEEE/ACM Transactions on Networking*, 24(2):887–900, April 2016.
- [488] Xiaoyu Yang, David Wallom, Simon Waddington, Jianwu Wang, Arif Shaon, Brian Matthews, Michael Wilson, Yike Guo, Li Guo, Jon D Blower, et al. Cloud computing in e-science: research challenges and opportunities. *The Journal of Supercomputing*, 70(1):408–464, 2014.
- [489] Yilong Yang, Quan Zu, Peng Liu, Defang Ouyang, and Xiaoshan Li. Microshare: privacy-preserved medical resource sharing through microservice architecture. *International journal of biological sciences*, 14(8):907, 2018.
- [490] Tetiana Yarygina. Exploring microservice security. 2018.
- [491] Tetiana Yarygina and Anya Helene Bagge. Overcoming security challenges in microservice architectures. In *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 11–20. IEEE, 2018.
- [492] Tetiana Yarygina and Christian Otterstad. A game of microservices: Automated intrusion response. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 169–177. Springer, 2018.
- [493] Y. Yetim, A. Bas, W. Mohsin, T. Everman, S. Abdi, and S. Yoo. P4runtime: User documentation, 2018.
- [494] 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*, 2019.
- [495] Dongjin Yu, Yike Jin, Yuqun Zhang, and Xi Zheng. A survey on security issues in services communication of microservices-enabled fog applications. *Concurrency and Computation: Practice and Experience*, 31(22):e4436, 2019.
- [496] Mingzhu Yuan, Yin Fang, Jingwei Lv, Shiqiang Zheng, and Zhiyi Zhou. Research on power trading platform based on big data and artificial intelligence technology. In *IOP Conference Series: Materials Science and Engineering*, volume 486, page 012109. IOP Publishing, 2019.
- [497] Shuhan Yuan and Xintao Wu. Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security*, page 102221, 2021.
- [498] Adel Zaalouk, Rahamatullah Khondoker, Ronald Marx, and Kpatcha Bayarou. Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring

- and sdn control functions. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–9. IEEE, 2014.
- [499] Zirak Zaheer, Hyunseok Chang, Sarit Mukherjee, and Jacobus Van der Merwe. eztrust: Network-independent zero-trust perimeterization for microservices. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 49–61, 2019.
- [500] Uwe Zdun, Erik Wittern, and Philipp Leitner. Emerging trends, challenges, and experiences in devops and microservice apis. *IEEE Software*, 37(1):87–91, 2019.
- [501] Chong Zhang, Xiao Liu, Xi Zheng, Rui Li, and Huai Liu. Fenghuolun: A federated learning based edge computing platform for cyber-physical systems. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–4. IEEE, 2020.
- [502] Nuyun Zhang, Hongda Li, Hongxin Hu, and Younghee Park. Towards effective virtualization of intrusion detection systems. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 47–50, 2017.
- [503] Yongyue Zhang, Xiangyang Gong, Yannan Hu, Wendong Wang, and Xirong Que. Sdnmp: Enabling sdn management using traditional nms. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 357–362. IEEE, 2015.
- [504] LI Zhiyi, Mohammad Shahidehpour, and LIU Xuan. Cyber-secure decentralized energy management for iot-enabled active distribution networks. *Journal of Modern Power Systems and Clean Energy*, 6(5):900–917, 2018.
- [505] L. Zhu, M. Karim Monjurul, K. Sharif, F. Li, X. Du, and M. Guizani. Sdn controllers: Benchmarking & performance evaluation. *arXiv preprint arXiv:1902.04491*, 2019.
- [506] Olaf Zimmermann. Architectural refactoring for the cloud: a decision-centric view on cloud migration. *Computing*, 99(2):129–145, 2017.
- [507] Olaf Zimmermann. Microservices tenets. *Computer Science-Research and Development*, 32(3-4):301–310, 2017.
- [508] Yuan Zuo, Yulei Wu, Geyong Min, Chengqiang Huang, and Ke Pei. An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis. *IEEE Transactions on Cognitive Communications and Networking*, 6(2):548–561, 2020.

Acronyms

5G 5th Generation. 3, 19, 21, 23, 56

ACM Association for Computing Machinery. 38, 39, 46

AES Advanced Encryption Standard. 81

AES-CTR Advanced Encryption Standard – Counter Mode. 81

AI Artificial Intelligence. 25, 26

AP Atomic Predicate. 15, 16, 17, 18, 95, 96, 97

API Application Programming Interface. 3, 9, 10, 19, 21, 23, 101, 103, 104, 124, 128

ASCII American Standard Code for Information Interchange. 120

ASIC Application-Specific Integrated Circuit. 13

BDD Binary Decision Diagram. 15, 18, 95, 97, 100

BF Bloom Filter. 74, 209

BGP Border Gateway Protocol. 14

BMC Best Master Clock Election. 22, 24, 126, 128, 131

BMCS Best Master Clock Selection. 22, 121, 125, 144

BSD Berkeley System Distribution. 143

CD Continuous Development. 88, 89

CI Continuous Integration. 56, 90

CP Constraint Programming. 26, 27, 100, 101, 104, 105, 106, 110, 114, 116, 117

CPU Central Processing Unit. 70, 82, 89, 91, 94, 112, 120, 126, 206

CSOP Constraint Satisfaction Optimisation Problem. 27

CSP Constraint Satisfaction Problem. 27

CTF Capture The Flag. 4

CTPH Context Triggered Piecewise Hash. 74, 86, 206

CVSS Common Vulnerability Scoring System. 143

DDoS Distributed Denial of Service. 12

DoS Denial of Service. 12, 65, 66, 72, 96, 105, 109, 110, 205

DPI Deep Packet Inspection. 25, 90

ETSI European Telecommunications Standards Institute. 3, 21, 66, 67

EV-DPI Efficient Verifiable Deep Packet Inspection. 25

FEC Forward Equivalent Class. 14, 15, 18

FOSS Free and Open Source Software. 134

FPGA Field Programmable Gate Array. 133

GDPR General Data Protection Regulation. 42, 53

GPS Global Positioning System. 125

gPTP Generalized Precision Time Protocol. 18, 20, 141, 205

HE Homomorphic Encryption. 74, 209

HR Human Resources. 26

HTTP HyperText Transfer Protocol. 94

HTTPS HyperText Secure Transfer Protocol. 92, 94

I4.0 Industry 4.0.

IaC Infrastructure as Code. 69, 71, 88, 90

- ICMP** Internet Control Message Protocol. 112
- IDS** Intrusion Detection System. 11, 12, 42, 53, 59
- IEEE** Institute of Electrical and Electronics Engineers. 18, 19, 20, 21, 22, 23, 38, 39, 46, 121, 122, 125, 133, 134, 141, 144, 205, 209
- IP** Internet Protocol. 10, 12, 14, 18, 23, 95, 110, 122
- IPS** Intrusion Prevention System. 11, 12, 42, 53, 59
- ISO** International Organization for Standardization. 23, 122
- IT** Information Technology. 26
- IV** Initialization Vector. 81
- LAN** Local Area Network. 122, 141
- MAC** Media Access Control. 12, 18, 19, 23, 122
- MANO** MANagement and Orchestration. 66, 67
- MB** MiddleBox. 25
- MD5** Message Digest Algorithm 5. 74, 81, 82, 84, 85, 86, 206
- MDD** Model Driven Development. 96
- MEC** Mobile-Edge Computing. 3, 21, 23, 124
- MFA** Multi Factor Authentication. 72
- MitM** Man in the Middle. 92
- ML** Machine Learning. 25
- MPLS** MultiProtocol Label Switching. 18
- NAT** Network Address Translation. 18
- NFV** Network Function Virtualization. 9, 11, 65, 66, 67, 137, 205
- NIC** Network Interface Card. 126, 133
- NIST** National Institute of Standards and Technology. 24, 40, 41, 42, 74

- NTP** Network Time Protocol. 23
- NVGRE** Network Virtualization Generic Routing Encapsulation. 13
- ODL** OpenDayLight. 67
- ONOS** Open Network Operating System. 67, 70, 95, 96, 97, 103, 104, 105, 112, 207
- OS** Operating System. 35
- OSI** Open Systems Interconnection. 23, 122
- OSM** OpenSourceMANO. 67, 68, 70
- OvS** Open vSwitch. 11
- OWASP** Open Web Application Security Project. 41
- P4** Programming Protocol-Independent Packet Processors. 13, 65, 100, 101, 103, 104, 105, 106, 107, 108, 109, 110, 112, 113, 114, 116, 117, 207, 209, 210
- PDF** Portable Document Format. 61
- PDU** Protocol Data Unit. 20, 22, 23, 124, 131, 133
- PEF** PTP Exploitation Framework. 127, 128, 130
- PFS** Perfect Forward Secrecy. 92
- PHF** Piecewise Hash Function. 74, 209
- PKI** Public Key Infrastructure. 123
- PRP** Parallel Redundancy Protocol. 141
- PTP** Precision Time Protocol. 3, 18, 19, 20, 21, 22, 23, 24, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 133, 134, 143, 144, 145, 205, 209
- RAM** Random Access Memory. 70, 89, 94, 99, 112
- RFC** Request For Comments. 12
- RPC** Remote Procedure Call. 103
- RQ** Research Question. 39

- RTT** Round Trip Time. 112
- SDN** Software Defined Networking. , 9, 10, 11, 12, 14, 18, 56, 65, 66, 67, 95, 96, 101, 104, 105, 106, 109, 110, 137
- SECaaS** SECurity as a Service. 55
- SGX** Intel Software Guard Extensions. 56
- SHA512** Secure Hash Algorithm with 64-byte words. 74, 81, 84, 85, 86, 206
- SIL** Safety Integrity Levels. 123
- SLR** Systematic Literature Review. 36, 209
- SNMP** Simple Network Management Protocol. 12
- SSH** Secure SHell protocol. 94, 126
- SSL** Secure Sockets Layer. 92
- STP** Spanning Tree Protocol. 22
- STT** Stateless Transport Tunneling. 13
- TCP** Transmission Control Protocol. 10, 12, 71
- TLS** Transport Layer Security. 92
- TLV** Type Length Value. 19, 20, 21, 22, 121, 122, 124, 125, 126, 127, 128, 129, 130, 131, 133, 143, 144, 145, 205, 208
- TSDB** Time Series Data Base. 66, 69, 70
- TSN** Time-Sensitive Networking. 18, 20, 23, 126, 133, 141, 142, 205
- UDP** User Datagram Protocol. 10, 12, 122, 127
- UML** Uniform Modeling Language. 96
- USB** Universal Serial Bus. 25, 93, 94, 207
- VCS** Version Control System. 90
- VIM** Virtual Infrastructure Manager. 67, 68

VLAN Virtual Local Area Network. 122, 141

VM Virtual Machine. 82

VNF Virtual Network Function. 25, 56, 67, 70

VNFM Virtual Network Function Manager. 67

VXLAN Virtual eXtensible Local Area Network. 13

List of Figures

2.1	Atomic predicates Verifier - Differences between header space and quotient space	15
2.2	A simple example of TSN network using the specialized PTP protocol (IEEE 1588), with a specialized profile called gPTP (802.1AS).	20
2.3	TLV packet formats. The mapping between the type name and 16-bit unsigned integer value can be found in the IEEE standard. Type and Length are represented by two unsigned 16-bit integers. Value interpretation is dependent from the Type field and its length is based on the homonym field, interpreted as the number of octets.	21
3.1	Schema of the method followed to gather the dataset for this review.	37
3.2	Time and category distribution of publications.	44
3.3	Conferences with the largest number of publications in our dataset.	45
3.4	Journals with the largest number of publications in our dataset.	45
3.5	Word-Net of the abstracts in our dataset.	50
3.6	Type of publications.	51
3.7	Attack type identified following the STRIDE classification.	52
3.8	Blockchain trend.	57
4.1	Schematic of the building blocks and interactions of the NFV architecture	67
4.2	Testbed with database and network probes placement	68
4.3	An example Grafana dashboard with network and system loads.	69
4.4	Network testbed	71
4.5	Statistics extracted from the network load scenarios. The DoS attack was simulated using tools described in Section 4.1.	72
4.6	Insertion procedure with two strings. The strings <i>password1234</i> and <i>password123!</i> are hashed independently. This insertion procedure processes the passwords as single items, leading to different hashed values.	77

4.7	Check procedure with two strings. The strings <i>password1234</i> and <i>helloworld</i> are hashed independently and the resultant indexes from the hash functions are checked in the bucket. If the lookup lead to a 0 value, the string is not present in the filter. Otherwise, it can be a value that is present in the filter or can be a collision (a false positive). . . .	77
4.8	n-gram insertion procedure with two strings. The strings are divided into n-grams (in this case bi-grams) and hashed using an <i>Insert</i> operation for every n-gram.	78
4.9	Different hash functions generation. These functions will lead to different use-cases. In the first two figures (the top-right and top-left ones) we have a fully random generation which leads to different cases every time as we cannot predict the output got from <i>Random</i> function. The fourth case employs a cryptographic function and a set of fixed strings (<i>fixedsalt_n</i>) to generate the same set of hash function based on a secret key <i>k</i> . . .	82
4.10	Speed of initialization of the filter and performances compared to edit distance. (a) Creation time of the filter changing the size of the salt strings. (b) Performances of the filter compared to the edit distance applied to the password data set.	83
4.11	<i>Performance</i> of the filter in terms of False Positive Percentage with different sizing of Γ (number of hash functions employed) and κ (size of the internal bucket). (a) Performance of a Bloom Filter based on MD5 hash functions in respect to number of hash functions used. (b) Performance of a Bloom Filter based on SHA512 hash functions in respect to number of hash functions used. (c) Performance of a Bloom Filter based on MD5 hash functions in respect to size of the internal bucket. (d) Performance of a Bloom Filter based on SHA512 hash functions in respect to size of internal bucket. . . .	85
4.12	<i>Comparison</i> of the two hash functions implemented (MD5 and SHA512) and performances of the different distance metrics (Jaccard coefficient, Dice coefficient and Cosine similarity) compared to a reference implementation of CTPH [248] (ssdeep). (a) Performance of the system by inserting and checking random password of different lengths. (b) Comparison of different functions to calculate distance. Namely: Jaccard coefficient, Dice coefficient and Cosine similarity.	86
4.13	Application scenario: The browser can insert the passwords <i>P4ssword123!</i> and <i>P4ssw0rd123!</i> into the Bloom filter, checking if they are similar enough to throw a warning of password similarity.	87
4.14	Infrastructure layout with the main functions implemented by the insider threat detection platform.	88
4.15	Grafana dashboard, the two circles indicates the number of process running in the machines and the number of alarms. At the bottom the information on the status of the machines is presented such as disk space, memory and CPU usages. The two buttons could be used to power off and reboot the machine promptly.	91

4.16	Alert originated by the insertion of a not allowed USB device. In this case the metadata of the analyzed machine and USB dongle are presented in the alert report.	93
4.17	SSL Inspection performances. The test was conducted over one hour of network downloads, brute-forces and network scans.	95
4.18	Average BDD Creation time / Policy time check after a link up / link down upgrade and node, links and forwarding rules increase.	98
4.19	Policy time check after a link up / link down upgrade and node, links and forwarding rules increase.	99
4.20	P-SCOR Overview of the three main component level	102
4.21	A block diagram showing the interaction of the various components with the integration between ONOS and P4 via P4-runtime and the ONOS pipeline. The components that have been originally implemented for this specific work are those highlighted with thicker lines.	103
4.22	Delay Link program work-flow	107
4.23	Asymmetric flow detection, example of work-flow. Three data flows are active and the P4 application is active in all switches. In S2 and S3, anomalies are detected since some flows show sensible asymmetries between packet numbers observed in the two directions. 109	
4.24	Comparison of the forwarding performance of an OpenFlow switch, a P4 enabled switch and a P4 enabled switch running the P-SCOR related programs.	114
4.25	Comparison of the total time transmission time of a burst of packet (with burst size from 20 to 200)	115
4.26	The link delay calculated by the P4 program, with an increased delay on the link from 1 to 20 s. The 95% confidence interval is plotted with the average. This confidence interval is quite good and was achieved with 5 experiments per point.	116
4.27	A graphical example of the application for asymmetric flow detection. The adopted policy is to drop until the number of packet in the overloaded direction decreases. The figure plots the link bandwidth used by the source, the number of sent packets and the threshold. When the threshold is reached and overcome the packets are dropped (used bandwidth goes to 0).	117
4.28	Detection time of traffic flows very asymmetric of P-SCOR, compared with the detection time reported in [14]. The threshold of detection is set to 100 packets per window. . .	118
5.1	TSN with safety scanners. In this case the field of view of the scanners is synchronized using TSN/clock synchronization protocols to cover complex view (e.g., 3d) areas. . . .	122
5.2	PTP hierarchy. The clock symbol specifies which nodes are synchronized each other using PTP.	123

5.3	Testbed design, all entities in the figure are optional and configurable. The system is built with IaC technologies and it can be reconfigured to include many clocks, attacker or different topologies. In black, entities that are not part of the time synchronization platform; in green entity mutually synchronized; in red de-synchronized entities.	127
5.4	PTP Exploitation Framework demonstration, the reconnaissance module displays the information got from the running PTP daemons.	129
5.5	Results of a clock de-synchronization attack in the virtualized testbed. In this attack we have exploited a <code>DISABLE_PORT</code> TLV. The clock then proceed to drift from the reference master, during the attack we keep the port disabled and then proceed to re-enable it after one minute.	130
5.6	Hierarchy-related attack: In the testbed the attacker issue a <code>CLOCK_ACCURACY</code> TLV. This TLV will poison the hierarchy management algorithm, when an election occurs, the attacker will become the Master / Grand-Master of the system. The synchronized clocks not controlled by the attacker are represented in green, in red the ones controlled by the attacker in every step.	131
5.7	When the attacker is elected as the master, the other clocks in the system are completely controllable. This representation of the clock is a query of the internal clock of a victim and compared with the internal clock value of a previously synchronized external observer.	132
5.8	Comparison between Bandwidth-requirements of a Denial of Service (DOS) attack and normal PTP operations (PTP). The data are a result of an average between five runs of 10 minutes and represented using logarithmic scale. The Denial of Service attack was done using <code>hping3</code>	132
5.9	Clock accuracy attack in the physical testbed. In this scenario, a slightly modified version of the previously presented attack is pictured. Over a time span of thirty minutes, the clock is taken back by the spoofed master by 10 seconds every 60 seconds.	134

List of Tables

3.1	Summary table and comparison with related works. For each row/work in the table, we report: its reference; its publication year; its type (Systematic Literature Review (SLR), survey, etc.); the number of publications it encompasses; whether it analyses white (peer reviewed) literature; whether it analyses grey (blog posts, etc) literature; the sources it used to search its dataset.	36
3.2	Cluster Authors Correspondence.	47
3.3	Correlation matrix among research questions.	59
4.1	Comparison between similar approaches. The ● symbol defines a full compliance to the row, the ◐ symbol denotes a partial compliance, and the ○ symbol denotes an absence of compliance to the row. The acronyms legend is the following one. BF: Bloom Filter, HE: Homomorphic Encryption, PHFs: Piecewise Hash Function.	74
4.2	The link delay calculated by the P4 program.	113
5.1	Exploited TLVs, details on attacks possible with these TLVs are presented in Section 5.2.	125
B.1	PTP IEEE-1588 2008 TLVs that seems to be unexploitable.	144
B.2	PTP IEEE-1588 2008 TLVs that can be used in the Reconnaissance phase.	145
B.3	PTP IEEE-1588 2008 TLVs that can be used to exploit the network.	146
B.4	PTP IEEE-1588 2008 TLVs that can be used to tamper the network.	146
B.5	PTP IEEE-1588 2008 TLVs that can be used to cover tracks of a PTP attack.	146

Listings

4.1	Threshold computation and checks using P4.	110
-----	--	-----

Acknowledgments

Acknowledgments

One of the goals of a hacker is the constant learning process.

This thank you is for all the people that helped me to learn something. Not just my masters, above all: Franco Callegati and Marco Prandini, which followed my entire Ph.D. student career, and Renzo Davoli, from which I constantly learn something, even during casual conversations. But all the people who were here to add a bit of information to my learning process and development.

Outside of learning, one of the main goals is surviving and carrying on, not always easy when doing something as particular as a Ph.D. I must mention my irreplaceable mentor in this course: Andrea Melis, which patiently guided me through the difficulties and removed a lot of burden from my shoulders¹.

To the co-workers of ULISse, which spent a lot of time debugging, swearing, sweating with me, playing CTFs, and drinking beers.

Even now that I am writing this thesis and it is not my best period for personal reasons, they're supporting me and distracting me from the sadness.

This learning and surviving process are not limited to Computer Science, work, or Academia, but that was just to introduce the punchline: For you all, farewell, and thank you for all the fish.

My past, current and future goals are and will always be dedicated to my family.

Bye, D.

¹Andrea can be cited also to alter this surviving process, fortunately, I am not writing this thesis in a Canyon searching for Pioneers' signs, but that's another story.

