INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Proactive Discovery and Mitigation of Security Vulnerabilities Leveraged by Software-Defined Networks

João Francisco Rosa Polónio

Master in Telecommunications and Computer Engineering

Supervisors:
PhD José André Moura, Assistant Professor,
Iscte-IUL

PhD Rui Neto Marinheiro, Associate Professor,
Iscte-IUL

October, 2024

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Proactive Discovery and Mitigation of Security Vulnerabilities Leveraged by Software-Defined Networks

João Francisco Rosa Polónio

Master in Telecommunications and Computer Engineering

Supervisors:
PhD José André Moura, Assistant Professor,
Iscte-IUL

PhD Rui Neto Marinheiro, Associate Professor,
Iscte-IUL

October, 2024

To my parents.

Acknowledgment

I would like to start by expressing my gratitude to my supervising professors, José Moura and Rui Marinheiro, for giving me the opportunity to embark on this research. Their guidance, dedication and encouragement were fundamental to my development, and I am deeply grateful for their patience, knowledge, opportunities and unfailing support throughout the course of my dissertation.

I would like to acknowledge Instituto de Telecomunicações, IT-IUL for the support for this research.

A special thanks to my parents, sister and family, who have been my unbreakable pillars of strength throughout my academic career. Your support and affection mean the world to me. You have been there every step of the way, offering advice, affection and motivation. I am immensely grateful for everything you have done to make my dreams come true, pushing me forward when things got tough and making me feel truly loved every step of the way.

I am also grateful to my colleagues and friends, Afonso Alves, Gonçalo Morgado, Lucas Oliveira and Tomás Santos. The challenges we faced, the countless hours spent in study sessions and the laughter and friendship we shared are memories I will forever hold a special place in my heart.

To my dear friends Alexandre Brito, Gonçalo Runa, Manuel Domingues and Martim Andersen: thank you for being a constant source of energy and encouragement. Your support and friendship was inestimable, reminding me of the importance of laughter and companionship, even in the most demanding moments.

Resumo

A deteção e mitigação proativa de vulnerabilidades de segurança em redes geridas por Software-Defined Networking (SDN) é essencial para garantir o seu normal funcionamento. Esta dissertação apresenta um sistema automatizado que deteta vulnerabilidades em dispositivos na rede e mitiga o seu risco, recorrendo ferramentas open-source.

A revisão da literatura identificou a necessidade de uma abordagem proativa, usando indicadores de risco standardizados e automação para melhorar a gestão de vulnerabilidades, destacando a necessidade de soluções que incorporem ferramentas de deteção ativa de vulnerabilidades.

O sistema desenvolvido orquestra a deteção e mitigação automatizada de vulnerabilidades através de uma plataforma de Security Orchestration, Automation and Response (SOAR), utilizando um scanner de vulnerabilidades e aplicando uma medida de mitigação, demonstrando a mudança de Virtual LAN através do controlador SDN, isolando os dispositivos vulneráveis.

Os testes realizados em laboratório e em ambiente virtual mostraram que a arquitetura proposta é rápida e eficaz, não causando atrasos significativos nas etapas do workflow, com exceção do scanning de vulnerabilidades, cuja complexidade implica tempos de execução mais elevados. O aumento do consumo de recursos pelo scanner de vulnerabilidades, ao analisar vários dispositivos em simultâneo, permite a utilização desta abordagem para a maioria dos casos de uso. No entanto, serão necessárias otimizações para a implementação em redes mais dinâmicas. Os resultados comprovam que a abordagem proposta permite detetar e mitigar vulnerabilidades de forma rápida e automatizada.

PALAVRAS CHAVE: *Vulnerabilidades*, *deteção*, *mitigação*, *redes definidas por software*, *automação*, *segurança de rede*.

Abstract

The proactive detection and mitigation of security vulnerabilities in networks managed by Software-Defined Networking (SDN) is essential to ensure their normal operation. This dissertation presents an automated system that detects vulnerabilities in network devices and mitigates their risk, using open-source tools.

The Literature Review identified the need for a proactive approach, using standardized risk indicators and automation to improve vulnerability analysis and mitigation, highlighting the need for solutions that incorporate active vulnerability detection tools.

The developed system orchestrates automated vulnerability analysis and mitigation through a Security Orchestration, Automation and Response (SOAR) platform, using a vulnerability scanner and applying a mitigation measure, demonstrating Virtual LAN switching through the SDN controller, isolating vulnerable devices.

Tests performed in laboratory and virtual environments showed that the proposed architecture is fast and effective, without causing significant delays in the workflow stages, with the exception of vulnerability scanning, whose complexity implies higher execution times. The increase in resource consumption by the vulnerability scanner, when analyzing multiple devices simultaneously, enables the use of this approach for most use cases. However, optimizations will be necessary for deployment in more dynamic networks. The results prove that the proposed approach is capable of detecting and mitigating vulnerabilities quickly and in an automated way.

KEYWORDS: *System vulnerability*, *detection*, *mitigation*, *software defined networks*, *automated operation*, *network security*.

Contents

List of Figures

List of Tables

List of Acronyms

ACL: Access Control List
AG: Attack Graph
AI: Artificial Intelligence
ARP: Address Resolution Protocol
ASP: Attack Success Probability
CTI: Cyber Threat Intelligence
CVE: Common Vulnerabilities and Exposure
CVSS: Common Vulnerability Scoring System
DDoS: Distributed Denial of Service
DHCP: Dynamic Host Configuration Protocol
DSRM: Design Science Research Methodology
DPI: Deep Packet Inspection
DRL: Deep Reinforcement Learning
DRRM: Dynamic Random Route Mutation
FL: Floodlight
FRVM: Flexible Random Virtual IP Multiplexing
FW: Firewall
GVM: Greenbone Vulnerability Management
GSPN: Generalized Stochastic Petri Net
HP: Honeypot
HTTP: Hypertext Transfer Protocol
ICMP: Internet Control Message Protocol
IDS: Intrusion Detection System
IoT: Internet of Things
IP: Internet Protocol
IPS: Intrusion Prevention System
JSON: JavaScript Object Notation
MAC: Media Access Control
MADS: MTD Adaptive Delay System
MISP: Malware Information Sharing Platform
ML: Machine Learning
MTD: Moving Target Defense
NASL: Nessus Attack Scripting Language
NBI: Northbound Interface

NICE: Network Intrusion detection and Countermeasure sElection
NSE: Nmap Scripting Engine
NTT: Nippon Telegraph and Telephone
ODL: OpenDayLight
ONOS: Open Network Operating System
OS: Operating System
OVS: Open vSwitch
PaH: Path Hopping
PoH: Port Hopping
PHCSS: Port Hopping technique with Masked Communication Services
PH-SND: Path Hopping Based SDN Network Defense Technology
PK: Port Knocking
SAG: Scenario Attack Graph
SBI: Southbound Interface
SCEMA: SDN-oriented Cost-effective Edge-based MTD Approach
SDN: Software-Defined Networking
SLR: Systematic Literature Review
SOAR: Security Orchestration, Automation, and Response
SSA: Security Service Adapter
TCP: Transmission Control Protocol
TV-HARM: Threat Vector Hierarchical Attack Representation Mode
VAaaS: Vulnerability Assessment as a Service Assessment as a Service
VLAN: Virtual Local Area Network
VM: Virtual Machine
VT: Vulnerability Test
XML: Extensible Markup Language
ZAP: Zed Attack Proxy

CHAPTER 1

Introduction

Computer networks are the nervous system of modern society, and as such it must be ensured that they are protected against malicious agents that could harm networks operation.

In this chapter, the topic addressed by this dissertation will be contextualized, and the motivation for addressing that topic will be explained (Section 1.1), as well as the objectives (Section 1.2) and the main related research questions (Section 1.3). The dissertation research methodology and the dissertation outline are available in Section 1.4. The dissertation contributions (Section 1.5) will also be discussed.

1.1. Context and Motivation

Networks are growing every day, and the security of networked systems is becoming an increasingly important need to account for. Hackers are well-known for posing a serious security risk, as on a daily basis, security firms and mass media sources report a rising number of sophisticated cyber attacks. Projections indicate that the worldwide expense associated with cyber crime will soar over the next years, escalating from $7.08 trillion in 2022 to $13.82 trillion by 2028, as represented in Figure 1.1 [1].

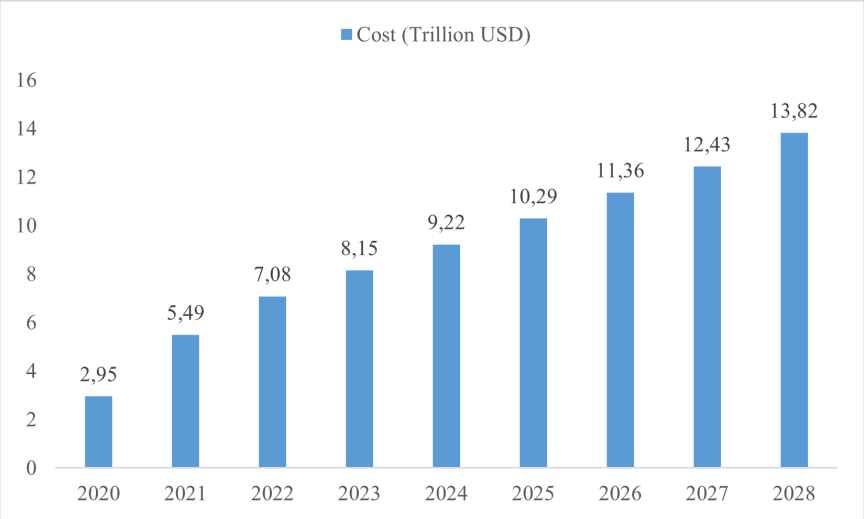

FIGURE 1.1. Estimated cost of cyber crime worldwide.

This is further complicated because, computer networks have become increasingly complex due to the mass dissemination of content and services, the introduction of sophisticated applications, and the growth of the Internet of Things (IoT), which is leading to the number and heterogeneity of devices connected to computer networks increasing exponentially. Controlling a such substantial number of network devices is difficult and prone to mistakes. This situation

occurs because many of these devices are not very robust in terms of security, making them attractive targets for attackers. Normally an attacker has an enduring economic advantage over the system defender, because the defender needs to fix all the system vulnerabilities, while the attacker only needs to exploit one of those vulnerabilities with success. Considering this economic asymmetry, the greater the difficulty for the human defender to keep the system safe and the ever-evolving landscape of cyber security threats against networked systems, it is imperative to delve deeper into the research on scalable, automated and proactive solutions for discovering and mitigating system security vulnerabilities, diminishing the risk of system normal operation being jeopardized [2].

Given these challenges, proactive security assumes an increasingly vital role since it is the first line of defense of networks and allows anticipating possible occurrences of security problems, such as vulnerabilities in the network elements. Consequently, there exists a significant and general necessity to explore novel self-adaptive programmable solutions that facilitate proactive mitigation of cyber threats prior to their occurrence, while concurrently minimizing the impact on normal system performance. However, new strategies for responding to these occurrences have struggled to emerge, since traditional networks are hardware-centric and have serious problems with research and innovation, flexibility and management capacity. So the need for networks with greater capacity, better accessibility, and dynamic management is turning into a crucial issue as the Internet and mobile networks evolve and new technologies arise [3]. Software-Defined Networking (SDN) presents a solution to these challenges by making networks more flexible, programmable, and robust. This allows security issues to be addressed with the utmost attention and care. Two very important points that should be mentioned are the fact that SDN enables the integration of many technologies into a unified network, resulting in a more flexible and scalable solution for heterogeneous networks, and that the logically centralized control plane enables network reconfiguration to be done dynamically and quickly in response to changing network scenarios without impacting the underlying devices. As a result, SDN may be a beneficial option for presenting answers to some of the traditional networks issues in that matter.

To illustrate the importance of proactive management solutions, an example would be an institution concerned about keeping its networked system as secure as possible. In this scenario, new devices may be added or have system updates, which can can make these devices vulnerable to external threats, and thus the institution system suddenly becomes more vulnerable to attacks. The value of integrating SDN with security systems is transpired by the SDN ability to program the network and automatically support security-related tasks such as vulnerabilities assessment, including their risk evaluation, and response. The visibility SDN controllers have from the network operation turn possible to automatically isolate some discovered compromised network hosts, servers or IoT devices inside a specific Virtual Local Area Network (VLAN) with limited access to important system operational parts. Therefore, the SDN is a flexible and scalable solution for heterogeneous networks, enabling the network reconfiguration without impacting the underlying devices, and protecting the normal network operation.

Considering the above issues, clarified with an use case, the identified problem in the current dissertation is how to achieve proactive and automated detection, and mitigation of vulnerabilities in the networked system, allowing the automated integration of active detection tools. In this way, I have decided to drive a comprehensive and systematic literature revision on these subjects, and also propose and develop a system to detect and mitigate vulnerabilities in the networked system where the above capabilities of SDN will be leveraged.

1.2. Objectives

The primary objective of this dissertation is the proactive and automated discovery, and mitigation of security vulnerabilities, addressed within a network environment controlled by SDN. One secondary objective of this dissertation is the development of a robust and comprehensive architecture that integrates various open-source security technologies, with a special focus on the effective orchestration of these tools, enabling automated and coordinated responses to identified vulnerabilities, while recognizing that encompassing all traditional security layers and principles is not the aim of the approach. In addition, this research has the secondary objective of rigorously evaluating the impact of these strategies on network and device performance, ensuring that they are executed efficiently and in a timely manner.

1.3. Research Questions

At the end of this dissertation, it is meant to provide answers to the following questions:

- RQ1 - How to automate device security vulnerabilities detection on networks?
- RQ2 - What resources the automated vulnerability detection consumes and their impact on the system scalability?
- RQ3 - How the usage of the SDN controller capabilities can enhance the network security?
- RQ4 - How timely can be the automated deployment of mitigation strategies?

1.4. Methodology

The methodology used in the development of this dissertation was "Design Science Research Methodology" (DSRM) [4], presented in Figure 1.2.

The DSRM process includes six steps: problem identification and motivation, the definition of the objectives for a solution, design and development, demonstration, evaluation, and communication [5], and four possible entry points: problem-centered initiation, objective-centered solution, design and development-centered initiation, and client/context initiation.

In this dissertation, the methodology starts with a problem-centered approach because the problem that it attempts to answer has already been defined and identified. The next stage is contextualization and motivation, which emphasizes the specific research issue and supports the importance of a solution. In addition to inspiring the researcher and the research's audience to pursue the solution, justifying the worth of a solution also enables the audience to appreciate the researcher's comprehension of the problem at hand, as presented in this chapter at Section 1.1. Knowledge of the problem's current state and the significance of finding a solution is one of the resources needed for this stage, which is covered in Chapter 2, starting with a theoretical

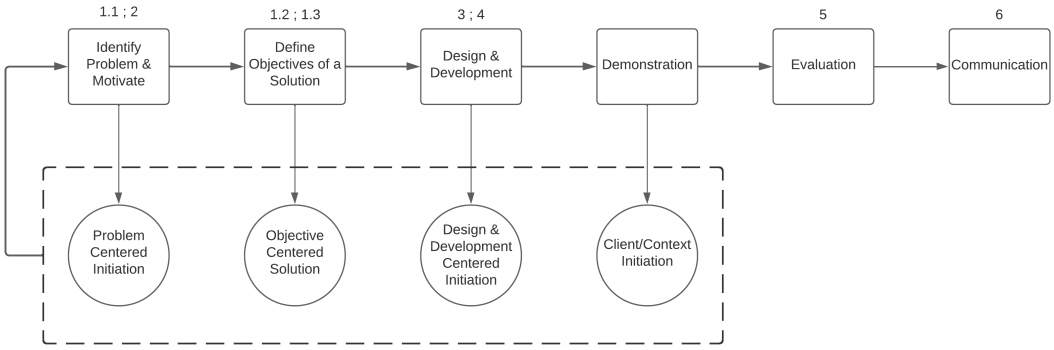

FIGURE 1.2. DSRM Process Model.

background that includes concepts fundamental to the research context and concluded with a presentation of related work. The definition of objectives defines and establishes which objectives the research will accomplish. This third stage is also covered in the current chapter, Section 1.2.

The third stage corresponds to "Design and Development" of the artifact. This step explains how the solution's architecture was designed in Chapter 3 to address some of the gaps left by the current literature, such as the need to integrate active probing tools into SDN, adopt proactive countermeasures, use standardized risk indicators and perform automated vulnerability detection and mitigation. The workflow of the solution initiates with the identification of active devices on the network, followed by the assessment of their vulnerabilities, analysis of the results to ascertain appropriate mitigation measures, and ultimately the application of those measures. Chapter 4 covers the practical development of the proposed architecture. Showcasing the flexible and customizable integration of the SOAR Platform with various security technologies, this chapter details the components of the system and their interactions within the workflow, ultimately discussing the implementation of a mitigation measure involving VLAN change to isolate vulnerable devices.

The fourth stage is "Evaluation", where a testing and validation phase of the artifact is conducted and presented. In Chapter 5, tests were conducted in laboratory and virtual environments. In the laboratory, the duration of the scan was measured, as well as the usage of CPU, RAM and bandwidth resources when scanning several devices simultaneously. It was concluded that GVM consumes a lot of CPU resources when scanning more than 8 devices. Average CPU usage varied from 14% to 48% for scans with 1 to 4 devices, and increased to 78% to 88% for scans with 8 to 32 devices. RAM utilization was stable, averaging between 41% and 45%. Average bandwidth utilization for scans with 1 device was 76 Kbps (TX) and 170 Kbps (RX), and for 32 devices, 784 Kbps (TX) and 2036 Kbps (RX). The virtual environment testing showed the system's responsiveness, with no delays in the workflow stages. Only

the vulnerability scanning phase caused a substantial delay, as the complexity of this task, aggravated by substantial CPU demand, led to longer scanning times. The scan time for a single device was 18 minutes, while the test with 8 devices took just four more minutes. With prolonged CPU saturation periods, the time went up to 27 minutes with 16 devices and 49 minutes with 32 devices. Implementing a VLAN change at the SDN controller level took only 19 ms, demonstrating the architecture's ability to quickly isolate vulnerable devices.

This dissertation draws conclusions and makes proposals for future work regarding the "Communication" final step of this methodology. This step is covered in this dissertation's Chapter 6.

1.5. Contributions

The dissertation contributions are as follows:

(1) Analysis and comparison of work concerned with system vulnerabilities detection and vulnerabilities risk against the system normal operation. Discuss the most relevant proposals in the current literature and address the integration of proactive and automated attack deception technologies, with the aim of minimizing the security risks of future threats that exploit system vulnerabilities.

(2) The submission and acceptance of a journal paper where the state of the art was critically discussed [2].

(3) Design and implementation of an novel SDN-based architecture to detect and mitigate vulnerabilities in existing devices.

(4) Integration of a Security Orchestration, Automation, and Response (SOAR) platform with open-source tools in an SDN-based system. Development of an automated playbook, along with various scripts that reduce manual intervention and enable automation.

(5) Development of a dynamic virtualized testbed with Hardware in the Loop.

CHAPTER 2

Literature Review

This chapter presents the state of the art of Proactive Vulnerability Analysis and Mitigation Leveraged by Software-Defined Networking (SDN). This chapter is organized as follows: Section 2.1 describes the Literature Review method applied in this dissertation. Section 2.2 presents the most relevant technologies and security vulnerability assessment metrics that will be discussed along the Literature Review. Section 2.3 addresses research on detecting system security vulnerabilities, which could be potentially explored by threats against the system normal operation. Section 2.4 compares and analyzes several proactive actions to minimize the security risk of future cyber attacks exploring system vulnerabilities.

2.1. Systematic Literature Review

The articles of the current Literature Review were collected using the Systematic Literature Review (SLR), which involves the Planning, Conducting, and Reporting phases [6]. To this end, Parsifal [7] was used to identify and evaluate all the most relevant literature on each paper topic. The Parsifal tool has been successfully used in [8]–[10].

During this investigation, the articles were discovered using the search string visualized in Figure 2.1. Then, the initial found literature was further filtered and aligned, considering the current Literature Review main goals, which are reflected on the selection criteria of Table 2.1. After this second phase, the Table 2.2 shows the number of publications returned by the search string (Figure 2.1) and the final papers accepted for being critically analyzed during this chapter. The average paper acceptance ratio was around 5%. The search conducted on Scopus did not reveal any additional works beyond those already identified through IEEEXplore and ACM Digital Library. Figure 2.2 shows the trend on the number of publications retrieved by the search string between 2016 and 2023.

TABLE 2.1. Selection criteria.

Inclusion Criteria	Exclusion Criteria
Pioneer	Prior to 2016
Number of Citations	Out of Scope
Free or under ISCTE's scientific access license	Paying Access
	SDN Architecture
	Attack-based Investigation
	Studies not written in English

In the search string of Figure 2.1, the first term regarding SDN aimed to mainly discover literature solutions leveraged by the SDN paradigm. The term after AND encompasses the

("SDN" OR "Software Defined Networks" OR "Software Defined Networking") AND ("Vulnerability Scanner" OR "Vulnerabilities Scanning" OR "Vulnerabilities Scanner" OR "Vulnerabilities Detection" OR "Vulnerability Detection" OR "Mitigation" OR "MTD" OR "Moving Target Defense" OR "AG" OR "Attack Graph" OR "Attack Graphs" OR "Honeypot" OR "Honeynet" OR ("P4" AND "Security"))

FIGURE 2.1. Literature review search string.

two domains covered in this Literature Review, namely Vulnerability Detection and Mitigation. Considering the case of detection, the search was restrained to works that also mentioned vulnerabilities, while in the case of mitigation, a more generic search was opted for. The search string was enhanced by adding a set of vulnerability mitigation techniques/technologies, which were separated by ORs, as the goal was to find papers discriminated by the used technique to protect the system operation. Thus, the search string aimed to find out a well representative sample of the more recent literature in the investigation area, and then, carry out a comprehensive analysis, discussion and interesting comparison among the found work. Survey [11] shows similar concerns regarding the use of some terms in its search string regarding SDN, security, vulnerability and mitigation.

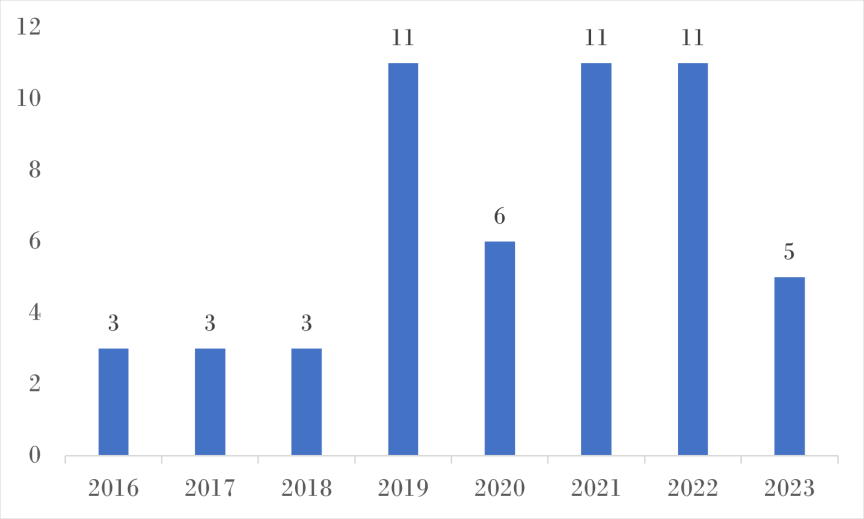

FIGURE 2.2. Post-2016 analyzed detection and mitigation papers per Year.

TABLE 2.2. Accepted related work.

Digital Library	Imported Studies	Accepted	Percentage
IEEEXplore	763	46	6.02%
ACM Digital Library	216	6	2.78%

A very interesting global outcome from the literature analysis is that there is a strong need for further investigation on detecting and mitigating vulnerabilities at networked systems [12], [13]. Although there is a lot of research in the area of SDN to detect and mitigate different

types of attacks, there is not the same degree of concern about the attack prevention phase. Considering my current best knowledge, this Literature Review is the first tentative to comprehensively revise proactive solutions for detecting and mitigating vulnerabilities in network systems controlled by SDN, with the final intent of diminishing the probability of future cyber attacks. It was excluded from this analysis the secure delivery of online content via either IP [14] or Information Centric [15] networks as well as the SDN intrinsic security [16], [17].

2.2. Background

It will be briefly explained the main concepts covered in the literature associated to the dissertation context, which are SDN, security vulnerability assessment metrics, and techniques for both vulnerability detection and mitigation. The concept of SDN is explained to promote a better understanding of the solutions presented in the literature. The security vulnerability assessment metrics discussed by me are used as a standard to identify and assess the vulnerabilities discovered in networked systems. In vulnerability detection it's explained the different types of vulnerability detection and how as a more complete analysis of the discovered vulnerabilities can be very relevant. Vulnerability mitigation defines the different techniques that have been employed in the literature to react to detected security vulnerabilities. This Literature Review is also compared with others to identify holes or areas for future research and ensure the validity and relevance of the dissertation contributions to the associated research areas. It is worth mentioning that the term host will frequently appear throughout this chapter as an equivalent to the term device.

2.2.1. Software-Defined Networking

The physical separation between the control plane and the forwarding plane is the key feature of the SDN architecture. The network's state is maintained by a logically centralized control function, which also gives instructions to the data plane. This separation is essential to achieving the necessary flexibility because it breaks the network control problem into manageable chunks, turns simpler to develop and implement new networking abstractions, and promotes network growth and innovation [18]. Figure 2.3 shows the seven main components that make up the SDN architecture, namely the three planes Data, Control, and Management, as well as the Northbound, Southbound, and East/Westbound interfaces [2].

The management plane is the topmost level of the SDN architecture and aggregates several applications with very distinct responsibilities such as routing, load balancing or security. The management plane is a crucial component of an SDN design because it optimizes, with the maximum abstraction from the network complexity, the global system operation using specialized software and tools. Through the northbound interface (NBI) of the SDN architecture, the management plane communicates with the control plane, enabling applications to submit commands and receive data from the control plane [3].

The control plane oversees and regulates the network's data flow. Through a specified southbound interface (SBI) controllers can program forwarding devices. Depending on several factors, including the packet's destination, traffic type, and the network's resources, every SDN

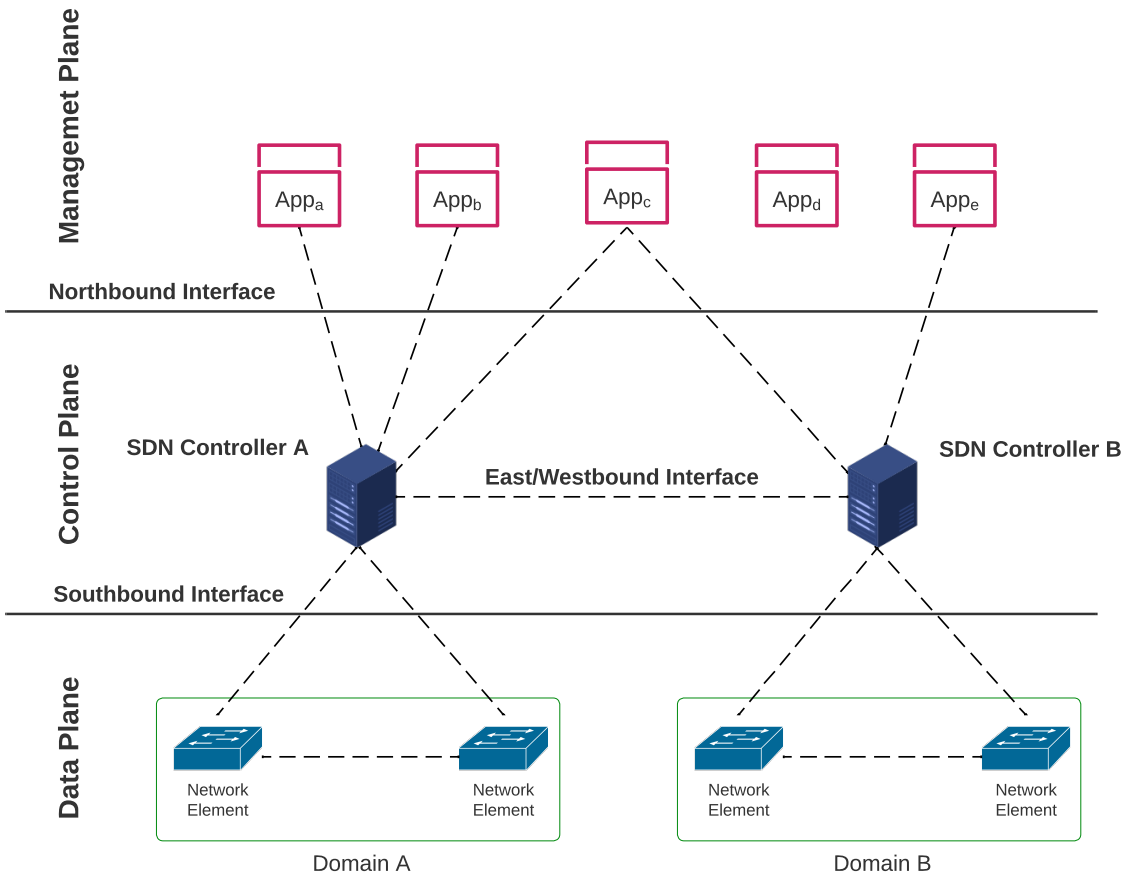

FIGURE 2.3. SDN Architecture.

controller is in charge of deciding how message data should be sent via the data plane. The information about the state of the network retrieved via SBI is used by the control plane to construct the network topology which is used to decide on routing and set up network devices to forward traffic in the more efficient way. The SDN controller enables the network to be more adaptable and agile by allowing the control of the network to be isolated from the physical devices. This facilitates the use of specialized software and tools to optimize the network and makes it simpler to administer and reconfigure the network as necessary. To enhance the performance and reliability of the logical centralized control level, several SDN controllers should decide in parallel about how the network resources should be used. Thus, some coordination among the SDN controllers is required for networks controlled by multiple SDN controllers. The more correct orchestration among the several SDN controllers can be supported by East-West interfaces [3].

The data plane is the bottom-most level of the SDN architecture and is formed by the forwarding devices as switches and routers. These network elements forward the packets according to the data forwarding rules, which are dynamically installed and updated via OpenFlow protocol by the top-level SDN controllers. Alternatively, in P4 programmable devices, the forwarding behavior is specified in devices scripts [19]. After the compilation of each script, it is generated a low-level firmware code compatible with the network device where that code

will be running as well as an intermediate level code for the device (i.e. client) OpenFlow API to ensure a proper comunication with the upper level SDN controller. This allows to modify and adapt the behavior of the data plane to meet new operational capabilities [20], which is a main advantage of using P4 in detriment of static and pre-defined behavior of pure OpenFlow devices.

2.2.2. Security Vulnerability Assessment Metrics

This subsection explores the important topic of Security Vulnerability Assessment Metrics, which includes the Common Vulnerabilities and Exposures (CVE) system and the Common Vulnerability Scoring System (CVSS). These metrics are fundamental components to actively identify, assess, and ranking security vulnerabilities in information systems, considering their risk against the system normal operation. The CVE system offers a defined nomenclature to uniquely find security vulnerabilities, promoting a shared comprehension within the cybersecurity community. Simultaneously, the CVSS offers a numerical method to evaluate the seriousness of these vulnerabilities, assisting in prioritizing their proper response activities [21].

Common Vulnerabilities and Exposure ID: The Common Vulnerabilities and Exposure ID (CVE ID) provide a reliable method of identifying unique vulnerabilities and coordinating the development of security tools and solutions. CVE IDs are formatted as CVE-YYYY-NNNNN, where the YYYY part represents the year that the CVE ID was assigned or the vulnerability was made public. Security flaws that become CVE entries are frequently contributed by members of the open-source community [22].

Common Vulnerability Scoring System: The Common Vulnerability Scoring System (CVSS) aims to assign vulnerability severity scores, allowing to prioritize responses and resources based on the found threat. A CVSS score is made up of three sets of metrics (Base, Temporal, and Environmental), each with its own scoring component [23].

The Base metric group represents a vulnerability's intrinsic characteristic that remains consistent over time and across user environments. It is made up of two types of metrics: Exploitability metrics and Impact metrics [21]. The Exploitability metrics measure how easy and the necessary effort to exploit the vulnerability. The Impact metrics indicate the negative consequence on the target vulnerable system component after the vulnerability has been explored.

The Temporal metric group reflects vulnerability characteristics that may change over time but not across user environments. The inclusion of an exploit kit, for example, would raise the CVSS score, but the creation of an official patch would lower it [21].

The Environmental metric group represents susceptibility factors that are relevant and unique to a specific user's environment. The availability of security mechanisms that may minimize either partially or totally the negative consequences of a successful attack, as well as the relevance of a vulnerable component being present in a technological infrastructure, are all considered by the environmental metric [21].

Table 2.3 shows the Severity Rating Scale for CVSS v3.0 and v3.1 [21]. The system CVSS vulnerability severity is evaluated on a scale from 0.0 to 10.0, where a higher score indicates a

more critical system vulnerability, meaning a vulnerability when explored by an attacker could produce a higher damage to the networked system. Thus, CVSS aids in the prioritization among several system discovered vulnerabilities and next mitigation tasks.

TABLE 2.3. CVSS v3.0 and v3.1 severity rating scale.

Severity	None	Low	Medium	High	Critical
Base Score	0.0	0.1-3.9	4.0-6.9	7.0-8.9	9.0-10.0

2.2.3. Vulnerability Detection

This subsection discusses the main technologies used for vulnerability detection, whether active or passive. Attack Graphs (AGs) are considered by me to be a detection technology because they are often used by authors as a way of improving threat risk classification, but it is worth mentioning that this technology is also used to improve the effectiveness of mitigation technologies, which will be discussed later.

Passive Scanning consists of analyzing traffic passing through a network monitoring point. This monitorization is normally unnoticed by the hosts running the services [24]. Some examples of passive tools are Wireshark, Snort, TCPDUMP and Zeek. Active Probing involves interacting with services by sending packets to each host and monitoring their response [24]. Some examples of active tools are Nmap, Metasploit, hping, Zed Attack Proxy (ZAP), Burp Suite, Greenbone Vulnerability Management (GVM) and NESSUS. AGs can be an addition to these technologies since represent the relationship between different system security vulnerabilities that may be exploited by an attacker, as well as the corresponding system access privileges acquired by the attacker after the exploitation of system vulnerabilities. Various AGs can be created based on the representations of nodes and edges. An AG consists on several node types such as state, host, privilege, or vulnerability. When an attacker successfully exploits a vulnerability, it frequently results in the escalation of privileges on the affected hosts, granting to the attacker the root access in those hosts [25]. This enables the attacker to prepare the next attack phases.

2.2.4. Vulnerability Mitigation

The importance of vulnerability mitigation techniques lies in safeguarding against and minimizing the system negative impact of potential security threats. Therefore, these techniques are extremely important for network security research. Figure 2.4 provides a classification of mitigation techniques further detailed and discussed in section 2.4. The deception was by far the most used mitigation type in the reviewed literature, but other interesting mitigation studies using Deep Packet Inspection (DPI), Firewall, Port Knocking and Port Hopping, are also critically discussed in Section 2.4.

The selection of the mitigation technique to be used on a particular use case controlled by SDN should be aware of that selected mitigation technique impact on the system performance and required system resources, as following explained. For example, MTD is resource-intensive

due to the frequent reconfiguration of network settings. However, applying MTD only to critical network subdomains may reduce the system overhead and keep its performance at a satisfactory level. Considering other alternative, such as DPI, it can also penalize the system performance due to real-time traffic analysis. Nevertheless, the negative impact on the system operation can be diminished by employing traffic sampling techniques, which analyze only a subset of traffic messages. Considering now PK, which typically consumes less resources than DPI, because the former controls the initial traffic access to a network domain via a correct sequence of used transport ports. In addition, the system performance may be protected by triggering PK operation via learned system events. The text below presents the diverse types of mitigation techniques found from the analyzed literature [2].

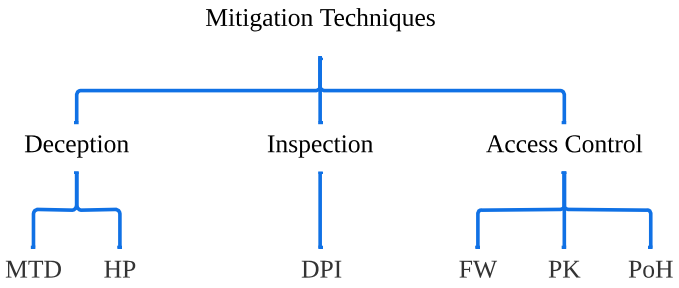

FIGURE 2.4. Classification of mitigation techniques.

Moving Target Defense: The goal of Moving Target Defense (MTD) is to randomly change the components of an underlying system. This guarantees that the information obtained by the attacker during the reconnaissance phase becomes stale during the attack phase since the defense has moved to a new configuration during that time. This creates confusion for the attackers, making it more difficult and expensive to properly exploit the system [26]. This dynamism requires the creation of a framework capable of accurately and timely examining the complex relationships between various hosts and security vulnerabilities, as well as ensuring that any changes made to the environment do not conflict with relevant active security policies neither penalize the system performance [27].

Honeypots: Spitzner [28] gave one of the first formal definitions: "A honeypot is a decoy computer resource whose value lies in being probed, attacked, or compromised". A Honeypot (HP) inhibits attacks because attackers waste time and money targeting HPs rather than the real target. HPs can also accelerate the reaction to attacks, because attack traffic becomes isolated from the production traffic, turning the attack detection and analysis much more easy for the system defense. Additionally, HPs may be taken totally down for inspection, allowing for a complete forensic examination. The insights may then be utilized to clean up production systems and analyze the exploit process, which is the first relevant step towards patching the corresponding vulnerabilities [29]. Honeynets are entire decoy networks made out of one or more HPs.

Deep Packet Inspection: Deep Packet Inspection (DPI) enables network traffic analysis and eventual posterior traffic filtering. Thus, DPI identifies the data portion of a packet, which refers to its content, as well as DPI classifies the traffic via a signature, which corresponds to the packet's ID. DPI devices analyze streaming packets to identify protocol non-compliant situations or domain intrusions. In the case of an incorrect traffic behavior, the associated packets should be dropped or deviated to an alternative destination for further inspection [30]. This technology can be seen as a very interesting way of mitigating vulnerabilities, but the available work only investigated it in a superficial way, requiring much more future work, as suggested in A.1.

Firewalls: Ensures the protection of the internal network's security against network attacks. The system employs established rules to selectively filter and regulate both incoming and outgoing traffic. The SDN switch may be configured to utilize the firewall (FW) functionality. To do this, one must incorporate the corresponding rules and action rules into the flow table of SDN switches [31].

Port Knocking: Port Knocking (PK) is a technique used to externally unlock ports on a firewall. This is done by initiating a connection attempt on a predetermined list of locked ports. PK is an authentication mechanism utilized to transfer data via a closed port. Upon receiving a proper sequence of connection attempts, the firewall rules are adjusted in real-time to let the host that initiated the connection attempts to establish a connection through certain port(s). Once the secure authentication sequence is successfully executed, the server initiates the opening of a port exclusively for the authorized user, therefore establishing a secure and reliable connection between the client and the server [32]. Consequently, an attacker unaware about the correct port knocking sequence cannot directly monitor the server via reconnaissance methods.

Port Hopping: The fundamental concept behind port hopping (PoH) is the dynamic alteration of port numbers for important nodes. Static ports can provide attackers with the opportunity to gradually acquire knowledge about the features of each service port during the reconnaissance attack phase. However, as the ports evolve over time, it becomes challenging to carry out an attack. The benefits of port hopping are straightforward and achievable, without the need for any modifications to existing protocols [33].

2.2.5. Related Surveys

Now surveys related to the dissertation goals are discussed. The available literature is more concerned in discovering attacks [34]–[36] and less in preventing them. There is a lot of research that proposes improvements to the SDN architecture as shown in [11], [17], [37]–[39], but only few works that proposes new ways of protecting the network by detecting and mitigating hosts security vulnerabilities before they become major problems. Survey [40] is interesting because it deals in depth with automation that is one of the parameters used to compare the literature collected in this subsection. In addition, [41] was the survey found to be more similar to this

Literature Review, but with the difference the former be more focused in scenarios involving IoT and the current Literature Review has a more wide network scope.

Survey [40] acknowledges the limitations of manual security operations and posits SDN as a solution that minimizes human error through its inherent design for minimal human intervention. The survey categorizes various security solutions based on their automation level and complexity. The automation is assessed using qualitative parameters like self-healing, self-adaptation, self-configuration, and self-optimization, while complexity is gauged by the resources and implementation requirements. A classification of the security solutions reveals the automation level enabled by each strategy and the complexity related to its implementation. The authors establish a collection of parameters and metrics to proficiently assess the network security design using SDN and shortly anticipate intelligent data planes to enhance the security of open, high-performance, and automated solutions.

The authors of [41] offer an extensive examination of the security vulnerabilities present in IoT devices. They categorize the possible attack surfaces into three layers: Hardware, Software, and Protocol Interface. They emphasize that the attack surface extends due to the growing complexity and interactivity of the devices. This survey provides a detailed comparison and analysis of detection, discovery, and mitigation methods, categorizing them accordingly. The authors provide a comprehensive examination of vulnerability analysis technologies, focusing on four aspects: analysis tools, vulnerability discovery, detection, and mitigation. Furthermore, the study recognizes the difficulties presented by the heterogeneity of IoT devices, requiring the development of automated techniques for generating patches for multi-platform binary code. The survey emphasizes that future research should also focus on AI-based vulnerability discovery and detection, large-scale vulnerability analysis technique, among others.

The greatest contribution of this dissertation Literature Review in comparison to previous work is that the former comprehensively analyzes the literature, performing, as shown in Figure 2.5, both aspects of vulnerability assessment (Detection and Analysis) and mitigation (Containment). This dissertation is also a foundational work of a novel research direction towards the automated and preventative elimination of security vulnerabilities in networked systems controlled by SDN.

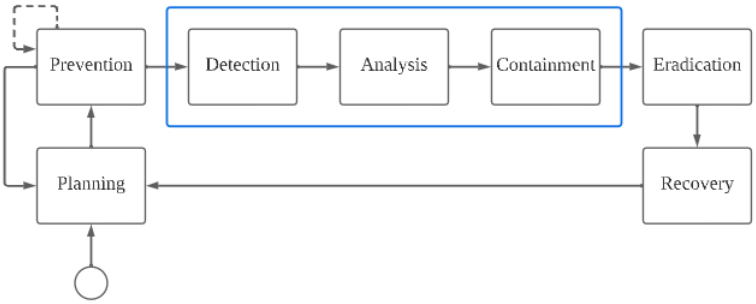

FIGURE 2.5. Scope of this Literature Review.

2.3. Proactive Detection of Security Vulnerabilities

In the previous section, the background to this Literature Review was discussed. We should be aware of that considering the new APIs and associated protocols imposed by typical SDN design, the potential attack surface increases, requiring innovative and more efficient approaches to proactively detect system vulnerabilities.

Referring now to the current section, the Table 2.4 lists the main characteristics of the surveyed work for detecting vulnerabilities using the system control level. This table contains a set of comparison parameters, as follows:

- Vulnerability Assessment: if the analyzed work performs vulnerability assessment, i.e. detecting and/or mitigating vulnerabilities. This parameter is used to highlight papers that are more concerned with the proactive attack prevention rather than the reactive detection of an ongoing attack.
- SDN Controller: SDN Controller is used or supported.
- Automation: the work proposes automated tasks with minimal human intervention, aiming to streamline processes, reduce errors, and enhance productivity.
- Risk Indicator: the work classifies the risk represented by the detected anomaly. CVSS implies that the work used the Common Vulnerability Scoring System; Custom is referred when the work used its own risk classification model. In addition, the work may also have used CVSS as a basis in conjunction with custom metrics to enhance the anomaly classification.
- Passive Scanning: defined in 2.2.3. Analyzes traffic passing through a network monitoring point.
- Active Probing: defined in 2.2.3. Interacts with services by sending packets to each host and monitoring their response.
- Proximity Score: this value can vary between 0 and 20, in an attempt to show how close the work is to the dissertation comparison parameters. This value is not intended to assess the quality of the work.

The most of the surveyed work adopted and tested a solution to automatically detect the system vulnerabilities, using a SDN controller. Nevertheless, only a few [42]–[46] have clearly identified the controller used or supported. In addition, from the analyzed contributions, the passive scanning [45]–[49] was by far more popular than the active scanning [42]–[44],[48] of system vulnerabilities. Further, the incorporation of a risk indicator to assess the system vulnerability level was marginal [42]–[45],[49] among the surveyed references and, among these, only [43] and [44] adopted a standard risk metric. From my analysis, there is a clear opportunity to strength future research on novel active scanning techniques for discovering system vulnerabilities, using well-known SDN controllers and standard system risk metrics. References [42]–[45] deserve to be recognized for their quality and relevance to what it's intended to investigate in the current Literature Review. Further details are given in below text.

Table Guide: ●: when the work satisfies the parameter analyzed and the authors make a direct or indirect mention of it. ◑: when the work does not directly mention the designated

TABLE 2.4. Detection Papers Comparison.

Paper	Vulnerability Assessment	SDN Controller	Automation	Risk Indicator	Passive Scanning	Active Probing	Proximity Score
[42]	●	POX	●	Custom	○	●	16.7
[43]	●	ODL, ONOS	◑	CVSS, Custom	○	●	15.0
[44]	●	ONOS	●	CVSS	○	●	16.7
[45]	●	ODL	●	Custom	●	○	16.7
[47]	○	N/A	●	○	●	○	6.7
[46]	○	ODL	●	○	●	○	10.0
[48]	○	N/A	●	○	●	●	10.0
[49]	○	N/A	●	Custom	●	○	10.0

parameter, but the work seems to partially comply with it. ○: when the work does not meet the designated parameter. N/A: when for the SDN Controller column, the authors do not mention the controller used/supported. This table guide also applies for next Tables 2.5 and 2.6.

The work done at [42] focuses on implementing a system that detects vulnerable IoT devices and attempts to fix their vulnerabilities before being accepted into the network. A POX Controller, firewall, DHCP server and a Host Tracker were used. A new arriving device sends a IP request to the DHCP server which responds and starts a vulnerability scan to inspect the device. Two tools are used to scan for vulnerabilities, NESSUS and a custom weak password scanner. Following the scan result, the new device can be added to one of two lists: whitelisted or blacklisted. Whitelisted devices correspond to devices in which no vulnerabilities have been found. Alternatively, blacklisted devices are considered vulnerable and dangerous to the system security. In addition, for every packet sent, the Host Tracker component checks if both source and destination hosts have been already scanned. This does not seem a good option, because it forces the controller intensively work in reactive mode, increasing the traffic latency.

Paper [43] proposes a systematic approach to evaluate and optimize the security posture of SDN and emphasizes the importance of analyzing the effectiveness of countermeasures in mitigating various threats faced in SDN. It introduces a framework for threat modeling and security assessment, utilizing three security metrics: network centrality measure, vulnerability score, and attack impact metrics. They have also developed a novel graphical security formally designated as Threat model using Threat Vector Hierarchical Attack Representation Mode (TV-HARM). TV-HARM enables security risk assessment of the SDN system. Experimental analysis was conducted to demonstrate the applicability of the framework and TV-HARM in capturing various threat vectors in SDN. The paper provides insights into the potential new threats in SDN and offers a comprehensive approach for evaluating the security of SDN.

Vulnerability Assessment as a Service (VAaaS) [44] cross-layered system is divided into three layers: Private Cloud, Fog, and Extreme Edge. In the Private Cloud, a Kubernetes container orchestrator manages the infrastructure, namely a primary ONOS SDN Controller. This SDN controller communicates with a monitoring service that communicates with the Decision Engine that takes different decisions depending on the reported event. The decision engine starts the assessment procedure by requesting Kubernetes to deploy an instance of OpenVAS to the associated distributed Kubernetes Slave, which is the Fog node responsible for the specified device. A registry of the underlying hosts and their assessment status is kept locally in a MongoDB database which includes their cybersecurity status, CVSS score, certification timestamp,

VLAN identification, and other valuable information. In the Fog layer, there can be several Fog nodes, and within each a Kubernetes slave and a slave SDN controller. The Extreme Edge is the system's lowest abstraction layer, where all devices, virtual and real, are installed/deployed. The monitoring service will receive the controller's list of connected hosts through its northbound API and send it to the Decision Engine which checks the MongoDB database and if there is a device in the registry that is not certified, the certification process is initiated by instructing the monitoring service to assign each device to a neutral, limited-connectivity-VLAN, and the orchestrator to deploy the OpenVAS-based assessment agent. As a result, it informs the deployed OpenVAS agent to analyze all unassessed devices. Until the assessment is completed, the devices remain members of the limited-connectivity VLAN. The vulnerability assessment generates a score for each device that is based on the CVSS which is used to assign each device to a connectivity-appropriate VLAN. As a result, if a device's evaluated severity is "None", it will be allocated to the full-access VLAN. If the severity of the device is between "Low" and "Medium", it will be allocated to the restricted-access VLAN. Similarly, if the evaluated severity of the device is "High", it will be allocated to the no-access VLAN. Finally, after completing its task, the orchestrator destroys the assessment mechanism. Every new device connected to the network follows the same procedure. Furthermore, the procedure is performed on a regular basis for all connected devices, every 10 days (pre-configured value).

DIVERGENCE [45] aims to provide scalable and intensive network traffic visibility for rapid threat detection and defense. The framework includes two main security services: a DRL-based network traffic inspection mechanism and an address shuffling-based MTD technique. By utilizing DRL, DIVERGENCE learns an optimal traffic inspection resource allocation policy under the uncertainty of malicious flow occurrence and performs MTD according to traffic inspection results reported from multiple Intrusion Detection Systems (IDSs).

Research [47] tries to improve the performance of an IDS system in SDN by identifying and preventing attacks on SDN devices. The proposed framework utilizes techniques based on signatures, an anomaly-based IDS for evaluating patterns of traffic and detects threats.

Study [46] examines the identification and prevention of attacks by utilizing RL techniques inside SDN. An agent modifies network security settings according to the current state of the environment while receiving a curiosity incentive signal that encourages discovery. The framework was assessed by subjecting it to attack scenarios using selected datasets.

SDNRecon [48] assesses the efficacy of cyber deception tactics, with emphasis on collecting sensitive data that may assist attackers in carrying out further harmful actions. The tool evaluates many elements of SDN networks, such as identifying the controller vendor, retrieving host information, and discovering vulnerabilities. SDNRecon emphasizes the importance of reconnaissance in the attackers' process of obtaining information, praising the efficacy of the SDNRecon tool in evaluating and improving cyber deception techniques. It also emphasizes the comparison with alternative technologies and the potential for synergy between reconnaissance tools and cyber deception systems.

Article [49] presents a methodology that examines attacks and generates risk assessment scores. It uses the Generalized Stochastic Petri Net (GSPN) model to analyze DoS attacks. The findings offer insights into attack paths, and investigate the correlation between risk levels and the timing of attacks. The results emphasize the clear relationship between the likelihood of risk and the average duration of attacks, offering useful insights for enhancing security evaluations and developing efficient countermeasures. The work hasn't been tested.

2.4. Proactive Enhancement on System Security

In the previous section we looked at vulnerability/threat detection technologies. The proactive enhancement on system security is critical to reduce the risk of future threats putting at risk the normal operation of programmable networked systems. As the number of threats and their sophistication increases, it is essential to implement proactive measures to help organizations mitigate potential system vulnerabilities before they can be exploited by malicious actors. To attain this, various proactive techniques can be deployed to identify, assess, and mitigate to possible attacks in real-time. Deceptive technologies such as MTD and honeypots can be used to build a virtual environment designed to mislead attackers away from critical systems and private data. Figure 2.6 shows the percentage of mitigation proposal types studied in the revised literature. As we can see, MTD is by far the mitigation technique most covered in the literature (51%), followed by HP with a much lower percentage (18%). Although I don't consider AGs (11%) to be a mitigation technique, but rather a tool to support it. I thought it relevant to consider it for Figure 2.6 and for Tables 2.5 and 2.6, as AG enhances the approaches capability to classify the discovered vulnerability risks.

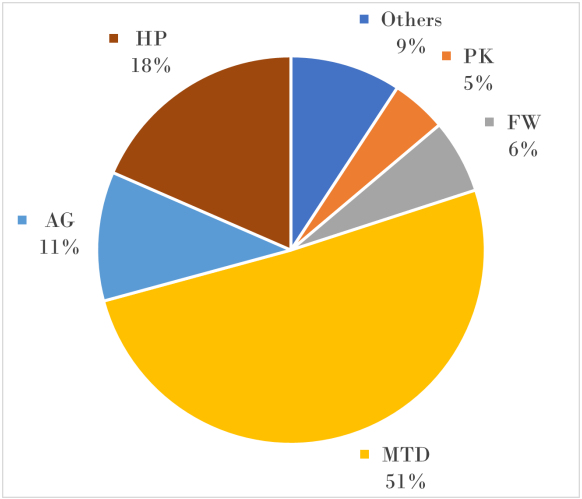

FIGURE 2.6. Percentage of mitigation methods employed.

Tables 2.5 and 2.6 list work focused respectively in the data and control planes, using the next comparison parameters:

- Technique: the used mitigation technique to reduce the risk created by threats. These techniques are taxonomized in Figure 2.4, with the exception of Convolutional Neural Networks (CNN).

- SDN Controller: SDN Controller is used or supported.
- Automation: the work proposes automated tasks with minimal human intervention, aiming to streamline processes, reduce errors, and enhance productivity.
- Elasticity: the proposed system may be considered to be elastic, in other words capable of adapt to real-time varying conditions. This parameter assumes the work fulfills one of three elasticity characteristics: scalability, flexibility/adaptability or dynamic resource allocation.
- Risk Indicator: the work classifies the risk posed by the anomaly.
- Latency: the work proposal has impact on the network traffic delay.
- Throughput: the work proposal has impact on the network throughput.
- Proximity Score: this value can vary between 0 and 20, in an attempt to show how close the work is to the relevant comparison parameters for this dissertation. This value is not intended to assess the quality of the work.

2.4.1. Data Plane

The current Literature Review explores the technologies used to strengthen the data plane, guaranteeing the safe and efficient transfer of data while actively reducing vulnerabilities. Proactive measures implemented in the data plane play a crucial role in constructing robust and reliable network infrastructures. More than half of the studies analyzed performance in terms of latency [50]–[53], but only one looked at the impact on throughput [50]. The methods used were Convolutional Neural Networks (CNN) [50], Moving Target Defense (MTD) [51],[52], Port Knocking (PK) [53]–[55] and Firewall (FW) [50], [55], [56]. None of the studies used risk indicators and only [56] mentioned the used SDN Controller. The [50],[54] and [56] papers tried to replace manual tasks with automated ones, while the [50], [52] and [56] contributions evidenced an elastic capability. In the below text, the discussion on the selected contributions is detailed.

Article [50] focuses on the security risks associated with IoT, highlighting the importance of using firewalls, SDN, and the P4 language for detecting attacks. It proposes a two-stage deep learning method for creating flow rules, which is evaluated and shown to outperform previous approaches. This contribution encompasses a pioneering architecture for safeguarding IoT networks and devices, offering a effective resolution for detecting malevolent data streams.

Investigation [51] presents an MTD method executed at the data plane and on every node along the forwarding path. The approach increases the cost for attackers to carry out network

TABLE 2.5. Mitigation Papers Comparison - Data Plane.

Paper	Technique	SDN Controller	Automation	Elasticity	Risk Indicator	Latency	Throughput	Proximity Score
[50]	CNN, FW	N/A	●	●	○	●	●	14.3
[51]	MTD	N/A	○	◑	○	●	○	7.1
[52]	MTD	N/A	○	●	○	●	○	8.6
[53]	PK	N/A	◑	◑	○	●	○	8.6
[54]	PK	N/A	●	○	○	○	○	5.71
[55]	FW, PK	N/A	○	○	○	○	○	2.9
[56]	FW	POX	●	●	○	◑	○	12.9

reconnaissance by randomizing network addresses. This protection is implemented at the link layer, ensuring secure access at both router switches and end-hosts. The solution additionally tackles a sophisticated threat scenario involving compromised network nodes, which disseminate controller communication information to provide MTD randomization. The results indicate a rise in the cost that attackers need to spend on timely reconnaissance, establishing the scheme as an efficient method for ensuring secure access to forwarding paths in SDN. The authors extend their work in [52] where they incorporate the randomization of IP addresses for MTD and use them for synchronization among network nodes, ensuring that all nodes in the network path understand a randomization parameter. It embeds a synchronization signature within the data packets, using a one-way hash chain to create these signatures, which prevents attackers from predicting future values. The scheme's efficacy is shown by significantly raising the cost of network reconnaissance for attackers by more than ten times compared to the prior literature. The scheme is also capable of being scaled up. By embedding synchronization signatures directly within data packets, the scheme reduces the need for additional control communications, enabling it to handle a growing number of nodes and connections seamlessly. The article conducts a comprehensive examination and verification of the system by implementing and experimenting on a testbed based on OpenvSwitch and CloudLab.

P4Knocking [53], a PK-based authentication mechanism implemented in the P4 language that offloads host-based authentication functionality to the network. The implementation relied on registers that hold values that work like counters to track the state of the knock sequence for a given source IP address where these counter define the access to a given destination IP address. The implementations focused on offloading the host-based authentication functionality to the network and making the mechanism transparent for the end host. It concludes P4Knocking to be more transparent and efficient compared to a host-based PK implementation.

PortSec [54] indicates the susceptibility of conventional PK sequences due to their static nature and introduces three new communication protocols based on sequences: static, partial dynamic, and dynamic. Each protocol provides a greater level of security than the previous. These protocols are specifically intended to operate inside the data plane.

P4Filter [55] leverages P4 to enhance network security through a two-level defensive approach. The first level of defense is a dynamic firewall that incorporates both stateful and stateless firewall concepts, effectively blocking packets from unauthorized sources. The second level of defense is an authentication mechanism that employs dynamic PK, which requires hosts to send packets to specific ports in a unique sequence assigned by the controller, which varies for each host. The P4Filter's packet processing involves three main modules within the P4 switch: two security modules for filtering packets and one forwarding module. When a packet arrives at the switch from an unknown host, it is sent to the controller, which maintains an ACL. The controller uses this ACL to assign rules that determine whether to allow or drop packets based on the security levels. If a packet does not match the internal network's criteria, a 'direction' bit is set, and a hash is calculated using various packet attributes.

Article [56] presents a stateful firewall intended for cloud environments. This FW differs from standard cloud FWs in terms of its approach to security rule configuration and rule matching, which are typically static and basic. The FW utilizes a finite state machine and a state table to directly extract, analyze, and record the connection status information of data packets within the data plane. The data plane packet processing is specifically built to execute stateful inspection and integrity checks by parsing and analyzing the structural information of the packet header.

2.4.2. Control

This subsection explores mitigation solutions that were implemented at the control plane which help in preventing unauthorized access, minimizing the risk of attacks, and maintaining the overall integrity of the network. Figure 2.7 shows the percentage of controller options mentioned in the various mitigation and detection works, from which ODL and Ryu are the most popular controller options in the revised literature.

From Table 2.6 the most researched mitigation technique is MTD [25], [57]–[83], [88], [89]. Honeypot (HP) is also a well-researched technique [84]–[95]. Both of these deception

TABLE 2.6. Mitigation Papers Comparison - Control Plane.

Paper	Technique	SDN Controller	Automation	Elasticity	Risk Indicator	Latency	Throughtput	Proximity Score
[57]	MTD	N/A	○	●	Custom	○	○	8.6
[58]	MTD	N/A	○	○	○	●	●	8.6
[59]	MTD	N/A	○	●	○	○	○	5.7
[60]–[62]	MTD	ONOS	○	○	○	○	○	5.7
[63]	MTD	N/A	○	●	○	●	●	11.4
[64]	MTD	N/A	○	○	Custom	○	○	5.7
[65]	MTD	ONOS	◑	◑	○	◑	○	10.0
[66]	MTD	Ryu	●	○	○	●	●	14.3
[67]	MTD	Ryu	○	◑	○	●	○	10.0
[68]	MTD	N/A	●	○	○	◑	○	7.1
[69]	MTD	N/A	●	●	Custom	◑	◑	14.3
[70]–[72]	MTD	Ryu	○	○	○	○	○	5.7
[73]	MTD	ODL	◑	●	Custom	○	○	12.9
[74]	MTD	N/A	○	○	○	○	○	2.9
[75]	MTD	ODL	◑	◑	○	○	○	8.6
[76]	MTD	ODL	○	●	○	●	○	11.4
[25]	AG, MTD	N/A	○	●	CVSS, Custom	●	○	11.4
[77]	AG, MTD, PoH	ODL	●	●	CVSS, Custom	○	○	14.3
[78]	AG, MTD	N/A	●	●	CVSS, Custom	○	○	11.4
[79]	AG, MTD	N/A	●	●	CVSS, Custom	●	○	14.3
[80]	AG, MTD	N/A	◑	◑	CVSS, Custom	○	○	8.6
[81]	CNN, MTD	N/A	○	●	○	●	○	8.6
[82]	MTD, PaH	N/A	○	○	○	●	○	5.7
[83]	MTD, PoH	NOX	○	○	○	○	◑	7.1
[84]	HP	Ryu	●	●	○	○	○	11.4
[85]	AG, HP	N/A	○	●	Custom	○	○	8.6
[86]	HP	ODL	◑	●	○	○	○	10.0
[87]	HP	N/A	◑	●	○	●	●	12.9
[88]	HP, MTD	Ryu	○	○	○	◑	○	7.1
[89]	HP, MTD	POX	●	●	○	◑	◑	14.3
[90]	HP	N/A	●	●	Custom	○	○	14.3
[91]	HP	N/A	●	◑	○	○	○	7.1
[92]	HP	ODL	●	●	○	○	○	11.4
[93]	HP	Ryu	●	◑	○	○	○	10.0
[94]	HP	FL	◑	●	○	●	○	12.9
[95]	HP	ONOS	●	○	○	●	○	11.4
[42]	FW	N/A	●	◑	○	○	○	7.1
[96]	AG, DPI	N/A	●	●	CVSS, Custom	○	●	14.3

techniques can be combined with AGs, as shown in [25], [77]–[80] for MTD and [85] for HP, but they can also be used in conjunction with DPI [96]. Other access control techniques such as PaH [82], PoH [83] or FW [42] are also present. SDN controllers Ryu [66], [67], [70]–[72], [84], [88], [93] and ODL [73], [75]–[77], [86], [92] are the most widely used/supported in the surveyed control plane literature. The concern to replace manual processes with automated strategies is present in papers [42], [66], [68], [69], [77]–[79], [84], [89]–[93], [95], [96]. Roughly half of the systems show characteristics of elasticity [25], [57], [59], [63], [69], [73], [76]–[79], [81], [84]–[87], [89], [90], [92], [94], [96]. There weren't found many papers to use risk indicators except [25], [57], [64], [69], [73], [77]–[80], [85], [90], [96] and from these only [25], [77]–[80], [96] used a standard risk metric. In terms of performance impact, works [25], [58], [63], [66], [67], [76], [79], [81], [82], [87], [94], [95] studied traffic latency but only [58], [63], [66], [87], [96] examined the impact on throughput. Honorable mention to [77], [79], [87], [96] for their proximity to the main theme of this review. More details are given in the text below.

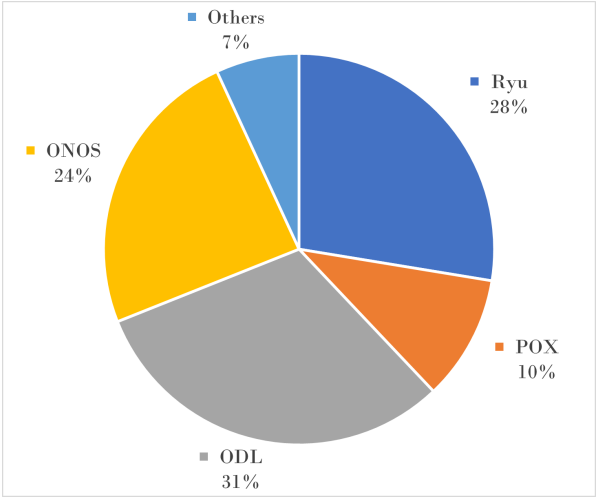

FIGURE 2.7. SDN Controllers mentioned in the revised literature.

2.4.2.1. *Moving Target Defense:* SDN-oriented Cost-effective Edge-based MTD Approach (SCEMA) [57] and [58] present MTD approaches to protect SDNs against DDoS attacks. This is accomplished on SCEMA [57] by shuffling a well-tuned group of hosts that have strong connections to important servers while [58] uses virtualized addresses for end-hosts, regularly remapping virtual IP addresses, hiding the actual ones.

Paper [59] examines the failure rate of algorithms in relation to mutation cost and trusted resource when some of these resources have been already under network attacks. The algorithm successfully mitigates attacks on nodes and ensures a decreased rate of service failures.

Study [60] examines the practical efficacy of a virtual IP shuffling MTD approach in an SDN testbed. It takes into account two categories of attackers: dummy and adjusting. The results confirm the effectiveness to prevent attacks in a practical network environment. It also states the need of continuous adaptation and assessing security in the presence of different levels of attacker knowledge.

MTD Adaptive Delay System (MADS) [63] provides efficient protection against scanning attacks. MADS employs a selective activation of the adaptive delay mechanism, where delays are only applied to response packets upon detection of an attack. MADS exhibits reduced network degradation in several aspects, including latency and throughput. The conclusion recognizes the effectiveness of MTD approaches in protecting SDN, while also emphasizing the potential negative impact on network performance. FRVM (Flexible Random Virtual IP Multiplexing) [64], aims to defend against reconnaissance and scanning attacks by enabling hosts to have several virtual IP addresses that are randomly allocated and changed over time. The multiplexing event guarantees a significant level of network diversity among hosts. FRVM is used by [65] that employs multiple SDN controllers to enhance both security and performance in large-scale networks, reducing the problems of having a single point of failure and scalability limitation. Paper [66] evaluates the effectiveness of MTD as a scan countermeasure, and its ability to detect low and high-rate scans. Results indicate that the combined use of MTD and Intrusion Prevention System (IPS) is effective in countering various rates of scans, and the proposed approach minimally affects communication delay and throughput.

SDN based MTD for Control and Data planes Security (SMCDS) [62] framework is designed to protect control and data planes from reconnaissance attacks. It leverages distributed shadow controllers to secure the control plane, enhancing its resilience and availability by presenting attackers with a constantly changing target. For the data plane, SMCDS combines reactive and proactive strategies, using shadow servers to deflect reconnaissance traffic and shuffling IP and port addresses. Theoretical models for calculating attacker and defender success probabilities are provided, and experimental results validate the defender's success across various scenarios.

Dynamic Random Route Mutation (DRRM) [67] mechanism is presented as a solution for MTD. It increases the randomness of mutations, minimizing the time required to perform them. The approach employs the Jaccard distance matrix and temporal restriction to enhance the mutation space, consequently diminishing the eavesdropping capabilities of attackers at certain nodes. In addition, a pre-distribution approach is employed to minimize transmission delay resulting from route mutation.

Methodology [68] employs a ML-based model to categorize rules and detect various attack types from malevolent network traffic. The paper outlines the design and execution of the suggested hybrid MTD system, and assesses its security benefits.

Frequency-Minimal Moving Target Defense (FM-MTD) [69] aims to minimize resource waste and loss of availability while effectively countering attackers. It achieves this by dynamically MTD across heterogeneous VMs based on attack probability. The scheme considers factors such as VM capacity, network bandwidth, and VM reputation to identify the ideal VM for migration. The results show that the FM-MTD scheme outperforms static schemes in terms of attack success rate.

The article [70] presents an experimental setup using Mininet and Ryu controller, and discusses the implementation of some MTD techniques. It also delves into the analysis of TCP

and UDP traffic in both traditional and MTD SDN network topologies, revealing considerable overhead on the controller, resulting in poor performance on several important network features, such as latency, jitter, and packet loss. Findings emphasize the need for further research to minimize these system overheads. Further work [71] from these authors, indicate the high pertinence on future work for reducing the controller overhead when the MTD technique is used.

The suggested MTD approach in [72] involves the mutation of IP addresses inside SDN. The effectiveness of the MTD approach is demonstrated through experimental assessment utilizing the Ryu controller and Mininet. IP mutation is identified as a viable method to incorporate MTD into SDN systems and reduce the risks of poisoning attacks. The main goal of a poisoning attack is the attacker to collect relevant initial information about the network operational states, to initiate DDoS attacks later more easily.

The article [73] introduce the Shuffle Assignment Problem (SAP), a complex problem formulated to reconfigure network topology, which scales exponentially with network size. This mechanism is further refined by the introduction of a topological distance metric, which aids in selecting the most effective countermeasures in real-time when an attack is detected. The Expected Path Value (EPV) is a critical metric used within the study to evaluate the effectiveness of network reconfigurations by quantifying the probability of an attacker reaching a target through compromised paths. The Defensive Network Reconfiguration Scenarios (DNRS) are a set of pre-computed network topologies, each with an associated EPV, from which an optimal topology can be selected to counter detected threats.

POSTER [74] highlights that the increasing prevalence of Artificial Intelligence (AI) is leading to increasingly advanced attacks, necessitating a re-evaluation of the effectiveness of MTD. The researchers developed a classification system for possible attacks against MTDs. The framework is intended to facilitate the creation of datasets that can be employed to replicate these sophisticated attacks. The paper points out a form of attack where an attacker may use ML methods to analyze network data and determine the MTD interval for a time-based MTD. Thus, an attacker can initiate the attack right after the last MTD trigger, maximizing the time to complete the attack. To increase the difficulty on the attacker having success in his/her reconnaissance attack phase, the authors propose a system defense technique like adding noise to the network data for hiding the MTD trigger from the attacker.

Random Host and Service Multiplexing (RHSM) [75] introduces a dynamic shuffling mechanism for IP addresses and port numbers to obfuscate the real identities of hosts and services at both the network and transport layers. RHSM employs a multiplexing strategy that allows each host to use multiple virtual IP addresses and services to use multiple virtual port numbers, which are periodically and dynamically shuffled. This shuffling is managed by a proxy installed on each end-host, which performs the necessary translations between virtual and real IPs and ports for incoming and outgoing packets. The article demonstrates through simulation experiments that RHSM significantly reduces the attack success probability (ASP) when compared to a static network configuration, while also considering the defense cost.

MTDSynth [76] objective is to improve agility by allowing systems to actively protect themselves against advanced threats through the dynamic adjustment of system configurations. The paper showcases the practicality of the framework by providing examples of temporal and spatial IP mutation, route mutation, and detector mutation implemented using ActivSDN on the ODL SDN controller.

2.4.2.1.1. Attack Graphs: Framework [25] emphasizes the rearrangement of network configurations for hosts that are both critical and vulnerable. The frequency of this rearrangement is defined by the level of criticality assigned to each host. The assessment of host exploitability is based on severity rankings assigned to vulnerabilities and the probability of successful attacks. The system includes the Three-tier Attack Graph (TAG) graphical model, which simplifies the analysis of host exploitability. In addition, a mechanism for predicting attack paths has been developed to safeguard important assets, and a method for managing system performance is proposed, which allows for control over address shuffling. The findings emphasize the efficacy of the solution in hiding network information from attackers and its ability to offer scalable and adaptive security, while maintaining an acceptable level of computation cost and latency.
MASON [77] introduces a threat score system that relies on vulnerability information and intrusion attempts to identify high-risk VMs. The MTD countermeasures, specifically port hopping, are then implemented to evaluate the reduction in threat scores within the cloud network. The study includes experiments that analyze network threat scores based on software vulnerabilities and IDS alerts, which help prioritize the most vulnerable services for MTD countermeasures. Additionally, the research examines the effectiveness of MTD countermeasures in relation to the attacker's reward, considering varying numbers of services and VMs. The findings indicate that as the number of nodes or services on the network increases, so does the threat score, highlighting the increased complexity and potential attack paths in larger networks.

In reference [78], MTD incorporates the use of AGs. The analytical models employed in this study integrate both random and weakest-first attack behaviors, effectively reflecting the preferences of attackers as well as the topological aspects of the system under analysis. The MTD system under consideration employs Bayesian Attack Graph (BAG) analysis as a means to evaluate security concerns and provide guidance for adaptive MTD operations. This approach is essential in the estimation of ASP for vulnerabilities, taking into account various elements like topological degrees and attacker scanning habits.

Paper [79] presents an integrated defense mechanism designed for IoT environments, with a focus on resource-constrained devices that are highly susceptible to attacks. The authors propose a combination of deception techniques, which involves the use of decoy nodes, and MTD. The authors develop and analyze four strategies for determining "when" to shuffle the network topology—fixed, random, adaptive, and hybrid and three strategies for "how" to perform the shuffling—genetic algorithm, decoy attack path-based optimization, and random. The study is conducted in the context of a smart hospital scenario, but the approach is applicable to any IoT environment. The results demonstrate that the proposed technique can extend the system's lifetime, increase the complexity of attacks on critical nodes, and maintain high service availability

compared to IoT networks without this defense mechanism. Additionally, the authors provide insights into the best combination of "when" and "how" strategies to achieve specific system goals, such as maximizing system lifetime and service availability while minimizing defense costs.

Article [80] proposes a proactive defense mechanism for IoT networks by combining MTD with cyber deception through the use of decoy nodes. The proposed framework introduces security metrics such as Attack Cost (AC), Return on Attack (RoA), and Risk (R) from both attacker and network defense perspectives. The authors have also provided a technique for reducing the cost associated to system defense.

2.4.2.1.2. Convolutional Neural Networks: AHIP [81], is an adaptive IP hopping technique designed for MTD, intending to mitigate a range of network threats. By employing a lightweight one-dimensional convolutional neural network (1D-CNN) detector, AHIP can dynamically choose IP hopping techniques according to network conditions, therefore efficiently mitigating reconnaissance and DoS attacks. The suggested approach is implemented using SDN, where the SDN controller is responsible for installing the 1D-CNN detector and the IP hopping module. The experimental findings conducted within a simulated environment provide evidence of AHIP's efficacy in safeguarding against network threats, while concurrently decreasing the burden on the system. The activation of adaptive IP hopping tactics is contingent upon the output of the trained 1D-CNN detector, hence facilitating AHIP's ability to successfully counter various attack scenarios.

2.4.2.1.3. Path/Port Hopping: Path Hopping Based SDN Network Defense Technology (PH-SND) [82] presents a technology known as Path Hopping which is designed to improve the network's defense by dynamically altering the path that data packets take through the network. PH-SND involves modeling the path hopping problem as a constraint-solving issue, where the goal is to find multiple paths that meet specific overlap and capacity constraints. Once suitable paths are determined, the controller installs the necessary flow entries into all switches along each path. These switches are then responsible for forwarding the protected flow and are capable of randomly changing the address and port information of the flow to further obscure the communication details between the sender and receiver.

Port Hopping technique with Masked Communication Services (PHCSS) [83], addresses the limitations of service port masking and the need for additional hardware, which typically result in increased overhead. The proposed technique is designed to efficiently detect and filter malicious packets, thereby reducing the server's port hopping costs and resisting against various network attacks. The experimental setup utilized Mininet, Open vSwitch, and NOX controller to simulate an SDN network environment, demonstrating that PHCSS can secure a network from port scanning and DoS attacks without overburdening the SDN controller's resources.

2.4.2.2. *Honeypots and Honeynets:* Article [84] describes a honeynet system that captures and monitors incoming traffic to identify and gather data on malicious attacks and the behavior of the attackers. The system employs a combination of physical and virtual HPs, with SDN and

container technologies, to facilitate the dynamic generation of honeynets and strengthen their deceptive capacity.

Paper [85] presents a deception resources allocation model based on SDN, which incorporates a multi-layer AG and a signaling game. This allows for the dynamic allocation of deception resources according to the severity of the threats. The system integrates geographical data into a multi-layer AG and offers a Top-N module for filtering attack paths. In addition, the model utilizes a signaling game strategy to determine flow scheduling, which is measured by the AG.

Dynamic Virtual Network Honeypot (DVNH) [86] employs dynamic instantiation of honeypot systems to efficiently divert attacks and protect targeted systems. It tackles the conventional issues of deploying HPs and managing costs by dynamically adjusting capacity according to demand while simplifying the operational complexity linked to honeypot administration.

HONEYPROXY [87] provides a flexible network access management system, globally monitoring internal traffic and supporting dynamic transitions between low-interaction and high-interaction HPs. By utilizing SDN, it enhances data control and capture capabilities, preventing fingerprinting attacks and improving overall resilience. The proposed architecture redistributes malicious traffic to HPs, allowing response selection without relying on fingerprinting indicators. Experimental results demonstrate HONEYPROXY's high throughput and minimal latency overhead, establishing it as an effective solution for advanced honeynet functionality.

The authors of [88] introduce a combination of MTD and SDN-based honeypots. The MTD architecture involves constantly changing the IP addresses of IoT devices and servers. Additionally, SDN-based honeypots are deployed to mimic IoT devices, luring attackers and malware. Experimental results demonstrate the effectiveness of this approach in defending against DDoS attacks and hiding network assets from malicious scanning.

MTD Enhanced Cyber Deception protection System (MTDCD) [89] offers protection mechanism against Advanced Persistent Threat (APT) attackers, which are commonly initialized via network reconnaissance. The method utilizes IP address randomization and the system's architecture consists of three primary components: the module for virtual network topology, the module for IP randomization, and the deception server. The primary role of the deception server is to detect and counteract malicious scanners while maintaining the appearance of an authentic network. The implementation of the MTDCD system resulted in a seven fold increase on the time for adversaries to identify susceptible hosts, and it also decreases the probability of successful attacks by 83%.

HoneyV [90] utilizes multi-phase data monitoring to improve the detection of malicious activities. Instead of terminating all sessions deemed suspicious, HoneyV dynamically routes traffic to server replicas with varying levels of monitoring intensity, depending on the assessed risk of attack. By doing so, HoneyV provides IDSs with training capability.

Paper [91] proposes a honeypot system that utilizes the combination of SDN and Recursively Defined Topologies (RDT). The authors present a mathematical approach to describe

RDTs and propose an algorithm for its generation. The objective is to develop efficient honeypots that can simulate complex data center environments on a single physical host. The design combines SDN with an orchestrator engine to create containerized infrastructure, which enables the creation of high-interaction honeypots. The work hasn't been tested.

S-Pot [92] is a smart honeypot framework with dynamic flow rule configuration for SDN. The authors conducted a performance evaluation of S-Pot in an enterprise SDN testbed network, simulating various types of attacks. The results demonstrated that S-Pot could detect attacks with a high accuracy of 97%. Furthermore, the study showed that S-Pot could improve the security of SDN networks by effectively generating rules and dynamically configuring the network, leading to greater accuracy and better performance.

The article [93] presents a network model for an intelligent honeynet. The proposed model is structured into three layers: the infrastructure layer, which includes network and honeynet exchange equipment; the controller layer, which interacts with the infrastructure layer via the SBI; and the application layer, which develops specific applications using the NBI API provided by the control layer. The intelligent honeynet comprises the attack migration mechanism, the topology management mechanism, the attack detection module, the policy generation module, and the honeynet management module. The attack migration mechanism involves the detection of attacks and the generation of strategies for traffic forwarding. The topology management mechanism dynamically generates honeynet nodes, links, and routing information to adapt to ongoing attacks. The article validates the model through experiments using Mininet to simulate attacks and demonstrate the honeynet's performance.

The paper [94] intends to overcome the obstacles of flow management and topology modeling. The suggested solution leverages the scalability and manageability of the SDN controller to emulate intricate network structures and stealthily redirect malicious data from a basic interface to a more advanced interface for extensive analysis. The design consists of a topology management module that maintains virtual topology information, an ARP simulation that handles ARP queries, and a flow table lifecycle management module that manages the metadata of flow tables and ensures their effective timing in the OpenFlow switch. The system incorporates a method for migrating attack traffic, which categorizes attacks and redirects them to suitable honeypots according to their degree of complexity.

SDNHive [95] aims to counter the spread of ransomware within a network. SDNHive utilizes the capabilities of the SDN controller to perform intrusion prevention measures like address blacklisting, connection blocking, and transparent traffic rerouting. The system includes a honeypot that functions as an active intrusion detection device.

2.4.2.3. *Other Methods:* One of the interesting things about the proposed system in [42] is that for devices for which vulnerabilities have been found the system will try to resolve them. If successful, the device will be allowed to join the network (whitelisted). If the vulnerabilities remain, the device will be blacklisted and an email will be sent to the device user offering suggestions to fix them. The decisions made by the scan server are translated into an ACL managed by the DHCP server, which is used by the firewall to enforce security policies and

restrict access to the network accordingly. The firewall inserts rules in the SDN controller which in turn updates the flow tables of the OpenFlow switch.

Network Intrusion detection and Countermeasure sElection (NICE) [96] uses AG Analytical procedures and allows the cloud to inspect and isolate suspicious machines (VMs) according to the current state of Scenario Attack Graph (SAG) which is defined by parameters such as IP addresses, vulnerability information (e.g. CVE) and alarmistic data. There is an agent called NICE-A in each cloud server that performs periodical vulnerability scans in VMs. Based on the severity of each vulnerability found, the solution can quarantine the VM and perform DPI or traffic filtering, avoiding to block the communications to the VM. Security indexes are specified for all VMs depending on factors such as connectivity, number of vulnerabilities, and respective CVSS. VMs can be profiled to obtain detailed information about their state, services running, open ports, and so on. The connection of a VM with other VMs is a crucial aspect that counts toward its profile. Any VM that is connected to a greater number of machines is more critical than one that is connected to fewer VMs since the result of a highly connected VM breach might cause more damage to the system. Knowledge about services operating on each VM is also necessary to validate the alerts related to it. Another important factor is the number of open ports on the machine, as these are highly targeted by cyber-attacks.

The work cloud visualized in Figure 2.8 highlights the more important terms referred in the analyzed literature. These terms are sized in the figure according to the frequency of their appearance in the surveyed papers. It were excluded from this extensive analysis the more popular non-technical terms used in the English writing (e.g. the, that, etc.). Analyzing Figure 2.8, the research community has been much more focused on the mitigation of network security attacks, eventually enhanced by SDN-based solutions, but significantly less involved in the detection of server vulnerabilities, before these could be explored by attackers. The conclusions are a strong motivation to the main goal of this dissertation.

FIGURE 2.8. Top-20 word cloud from the analyzed literature.

CHAPTER 3

System Design

This chapter discusses the system architecture and design concepts, with the objective of addressing the challenges mentioned in Section 3.1, which discusses the literature foundations that guided the system's development, addressing current weaknesses in the available related literature, and emphasizing the need for proactive security solutions. Section 3.2 outlines the fundamental principles that shape the system. Section 3.3 details each system component, explaining their functions, relationships, and contributions to the overall system operation.

3.1. Literature Foundations

Most of the revised work in the literature [2] aims to guarantee the system security in a reactive way, except for few work proposing mitigation techniques such as Moving Target Defense (MTD) or other similar obfuscation methods to mislead eventual attackers. This means the great majority of the proposed solutions try to do their best in successfully detecting and mitigating running attacks. Thus, there is a strong and generic need to investigate new self-adaptive programmable solutions, offering proactive prevention of cyber threats before their concretization and without penalizing too much the system normal performance.

Regarding the detection, one of the weaknesses found in the literature is the extremely small number of studies that perform vulnerability assessments using standardized risk metrics. The use of standardized risk metrics is crucial for firms seeking to strengthen their security defenses and actively mitigate possible threats by allowing to prioritize vulnerabilities based on severity and potential impact. In this way, security breaches and data leaks are averted while, at the same time, it also avoids less informed strategic security decisions, and the risk of allocating resources inappropriately or leaving serious vulnerabilities without an adequate response. Furthermore, vulnerability assessments with standardized risk metrics enhance strategic decision-making by offering valuable insights into new threats and directing investments in security. Therefore, further research into this matter is crucial. Another aspect where the literature does not give enough attention is the integration of active probing tools in Software-Defined Networking (SDN), since they allow organizations to promptly detect vulnerabilities by actively examining their systems, uncovering potential ports of non authorized entry and containment flaws. These active probing techniques facilitate the prioritization of mitigation operations, enabling organizations to rapidly and efficiently address the most crucial system vulnerabilities. This methodology helps to prevent incidents, since the knowledge obtained from active scanning informs about the creation and improvement of proactive measures against such imminent vulnerability exploitation by external attackers.

Data plane mitigation papers lack of risk indicators to accurately assess risks. This poses as a notable hole as it impedes security measures to prioritize the most crucial vulnerabilities or hostile behaviors, resulting in a less efficient response to potential threats. Taking advantage of the capabilities of the data plane offers great potential, especially in tackling issues related to latency and resource allocation. No mitigation solutions were implemented at the data plane level because, despite promising, this approach turns difficult to proactively detect security flaws that may lead to incidents in the future. It is therefore preferable to consider other approaches that promote prevention.

In mitigation papers of the control plane, it was found that, although there are already some studies that present the use of risk indicators, this area still requires more attention for the reasons mentioned in the previous paragraph and therefore in this dissertation special attention was given to use the standardized format of the Common Vulnerabilities and Exposures (CVE) and respective Common Vulnerability Scoring System (CVSS) to identify the level of vulnerabilities severity, before applying a mitigation measure. Automation, a key factor in dealing effectively with threats, was not sufficiently explored, with many studies relying on manual intervention. In addition, performance testing is often neglected, as most work does not assess the system's impact on network resources, latency or throughput. Finally, most studies focus on mitigating ongoing attacks rather than preventing vulnerabilities before the attacks occur.

It was also found in the literature that the number of studies that addressed both vulnerability detection and mitigation was very small and although some automated processes were noted, to my best knowledge there was no previous work integrating a Security Orchestration, Automation, and Response (SOAR) to orchestrate the various tools used. The systems developed did not allow the change or the addition of new features for detection or mitigation measures, which is not ideal due to the constant evolution of threats. In addition, there is a need to integrate active probing tools into the SDN, and the development of new proactive approaches. Moreover, there is a lack of comprehensive solutions that demonstrate measurable impacts on network performance and the need for standardized metrics when applying mitigation measures. Collectively, these observations point out the potential to design a more robust architecture that addresses these issues.

3.2. Principles of the System

It was designed a system with the ideas from the previous section in mind to address some of the identified open issues. The diagram in Figure 3.1 illustrates the main building blocks and workflow that make up the designed system. It begins with the detection phase, in which the system continuously monitors network devices for vulnerabilities, using active probing tools. The vulnerabilities detected are classified using formats such as CVE, which standardizes the information and allows for easier comparison and prioritization. Once the vulnerabilities have been classified, the system moves on to the analysis phase, where the severity and risk associated with each vulnerability are assessed. This analysis helps to determine the most effective mitigation measure. Based on the results, the mitigation phase is triggered, where appropriate

mitigation measures are implemented, such as isolating compromised devices through VLAN changes or blocking malicious traffic.

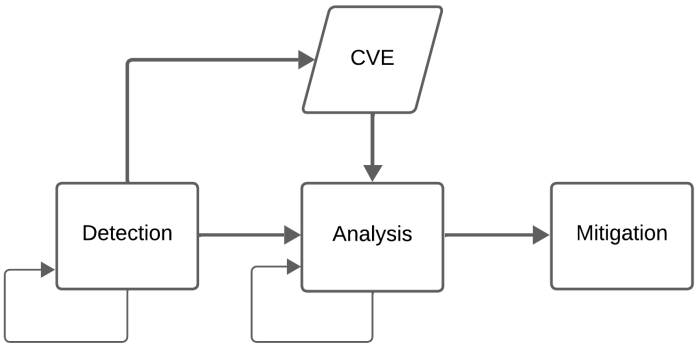

FIGURE 3.1. System building blocks.

In a scenario, where security policy is more relaxed, such as in information systems development environments, the architecture presented can be extremely useful for ensuring continuous and automated security. In a development environment, devices may be frequently updated or exposed to software in unstable versions, which increases the likelihood of vulnerabilities. Proactive vulnerability detection through active probing tools is essential to monitor security flaws as they arise, without disturbing the developers work.

Once the architecture and phases of the system had been defined, it became clear that a structured sequence of actions was needed to ensure that each vulnerable device on the network could be systematically discovered, classified and its risk mitigated. Therefore, in order to develop the proposed system, the following stages were identified:

- Network device discovery.
- Detection of vulnerabilities in devices.
- Generation of a vulnerability report.
- Parsing and analysis of the report.
- Apply a mitigation measure.

Every stage of the system development process is critical for ensuring comprehensive network security. Discovering the devices is essential, as it offers an overview of all the active ones. Identifying vulnerabilities on these devices enables the detection of security flaws prior to their exploitation by malicious actors. Producing a vulnerability report enables consistent and precise communication of security issues, making it easier to make informed decisions about mitigation measures. It is essential to parse and analyze the report in order to prioritize vulnerabilities according to their severity, allowing for prioritized responses to the most critical threats. Implementing a mitigation measure is crucial for effectively reducing the risk of exploitation and safeguarding the network from potential future attacks.

The successful implementation of these development phases was guided by a set of key design principles, ensuring that the system fulfill essential quality attributes defined in [97]:

- Interoperability.

- Proactivity.
- Adaptability.

Interoperability guarantees that the system can effortlessly integrate with different tools, platforms, and technologies, thereby optimizing its compatibility and overall functionality. Proactivity allows the system to detect and address potential threats and vulnerabilities in advance, thereby enhancing network security. Finally, adaptability maintains the system's flexibility and responsiveness to changes, allowing it to evolve in accordance with new requirements, technologies, or emerging threats.

3.3. System Components

A decision was made to improve the integration of vulnerability detection technologies and mitigation measures, seeking to automate the entire process utilizing a SOAR within the system design.

The system's architecture has several fundamental components described in the component diagram in Figure 3.2. It consists of two primary elements: the SOAR Server and the Security Tools Server. The SOAR Server hosts the SOAR Platform, while the Security Tools Server houses essential tools, including the Vulnerability Scanner and the Device Discovery Module.

The separation of the SOAR Server from the Security Tools Server enhances scalability, and flexibility in updates and maintenance, allowing independent changes without direct negative repercussions on the other, thereby minimizing interference and resource conflicts, and ensuring flawless functionality. Furthermore, it enhances the segregation of tasks, a crucial practice for ensuring that the centralized control of the SOAR Server remains unobstructed by the operations of the tools, thereby enhancing structural integrity. Additionally, this architecture allows the deployment of Security Tools Servers in areas where the SOAR Server may not have visibility or access, extending the reach of the SOAR's capabilities.

There are other important components. The DHCP server is responsible for dynamically assigning IP addresses to clients on the network, eliminating the need for manual configuration. One (or more) devices that will be scanned for vulnerabilities. An SDN architecture made up of a controller that manages the network and a switch that bridges the devices, while the controller also offers an additional interface for applying mitigation measures. VulnMatrix which is a database that stores information about the devices discovered on the network, as well as the data needed for vulnerability scanning.

SOAR Platform

The SOAR Platform is hosted within the SOAR Server. This platform is crucial as it enables the system to address vulnerability response and automate the orchestration of the different tools, leading to improved operational efficiency and consistency, which not only speeds up response times but also avoids human errors. This component allows for the centralization of the system's intelligence within a single module, thereby simplifying future enhancements and updates.

A number of functional requirements were defined that the SOAR Platform had to fulfill in order to be implemented in the system:

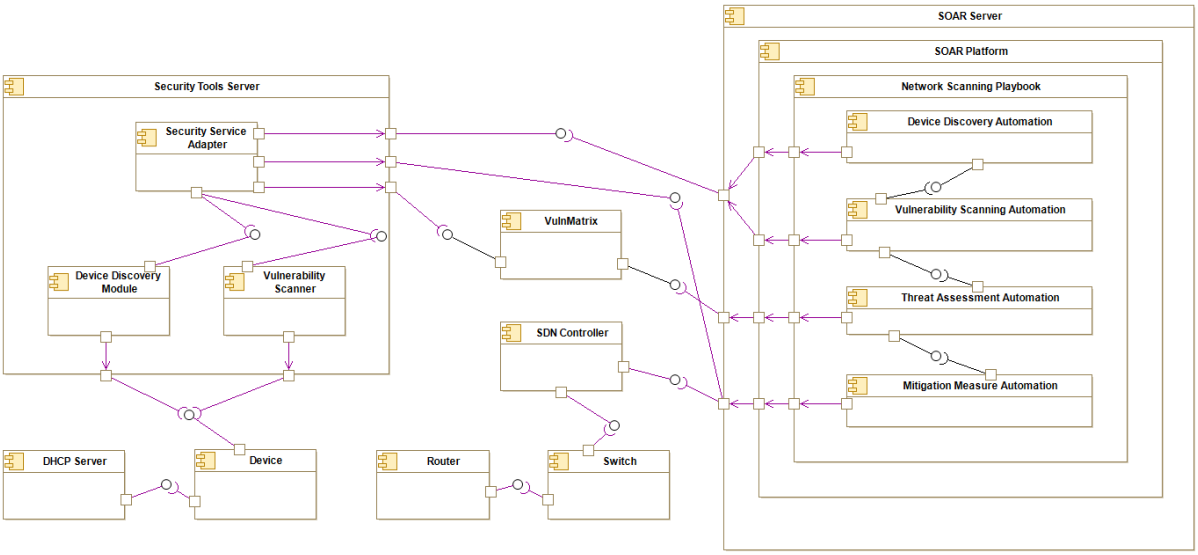

FIGURE 3.2. Components Diagram.

(1) Ticket opening and closing.
(2) Writing automation scripts in a popular programming language.
(3) Creation of playbooks to create automated processes.
(4) API to run tasks outside the graphical environment.

These functional requirements ensure that the SOAR Platform offers a comprehensive and flexible solution for handling vulnerability analysis and mitigation. The ability to open and close tickets enables the efficient administration of workflows for resolving security flaws. The availability of writing automation scripts in a popular programming language has several positive aspects. Firstly, it makes easier to maintain and improve the scripts, as documentation and learning resources are widely available. Another advantage is a more easy integration with other tools and platforms, since many software solutions provide libraries or APIs for popular languages. Playbook creation facilitates the implementation of standardized procedures for managing security problems, thereby promoting uniform and replicable responses, which improve operational dependability and effectiveness. In addition, offering an API for executing tasks outside the graphical interface enables enhanced flexibility and advanced automation capabilities.

Figure 3.2 shows the “Network Scanning Playbook” component, which corresponds to the sequence of automations used to perform the functions of this system. Section 3.2 highlights the main steps to be taken to complete the system’s objective, which are translated into the automation scripts (Device Discovery, Vulnerability Scanning, Threat Assessment and Mitigation Measure) that run within this Playbook.

The Device Discovery Automation is designed to discover active devices on the network. It communicates with the Security Service Adapter to send its request to the Device Discovery Module. After discovering the devices, the Vulnerability Scanning Automation inspects these for security vulnerabilities, instructing the Security Service Adapter to indicate the device to be inspected to the Vulnerability Scanner. The data generated, in the form of a report, is used

as input for the Threat Assessment Automation. This automation assesses the severity of the vulnerabilities identified, allowing for informed risk management decisions. Subsequently, the Mitigation Measures Automation implements a countermeasure to mitigate the risk of the vulnerable device, where it can communicate with the SDN controller to, for example, change the VLAN and isolate the device at risk, or communicate with the Security Service Adapter to apply another mitigation measure.

Security Service Adapter

The Security Service Adapter (SSA) connects the tools to the SOAR Platform. By developing the SSA, it was possible to encapsulate the complexity of the individual device discovery and vulnerability scanning processes, allowing greater control over the communication between these tools and the SOAR Platform. In addition, the abstraction provided ensures that automations on the SOAR Platform become considerably simpler. This is because SOAR does not need to deal directly with the protocols or implementation nuances of each tool, as the SSA centralizes the management of these operations and provides a consistent, unified interface.

Vulnerability Security Scanner

The Vulnerability Security Scanner is one of the key components of system design. It scans for known security vulnerabilities, misconfigurations, and obsolete software that may be susceptible to exploitation by malicious actors.

A number of vital features have been identified that a Vulnerability Security Scanner must possess in order to ensure its effective operation in the system:

- The ability to update its vulnerability database regularly to reflect the latest security threats and vulnerabilities.
- Flexibility to modify scan parameters, such as target IP ranges, ports, and specific vulnerability checks.
- Support for API scripting to enable automation and integration with other security tools and workflows within the network environment.

Vulnerability Security Scanner should include comprehensive reporting to provide detailed insights into detected vulnerabilities in addition to these core features. In reports, severity ratings like CVSS scores, potential impacts, and remediation actions should be included. This helps prioritize and address critical vulnerabilities. Integration with threat intelligence feeds should also be available, keeping the scanner updated on new threats.

Device Discovery Module

The Device Discovery Module is responsible for detect and map all currently active devices in the network. Ensuring comprehensive coverage of the entire network during vulnerability assessments.

SDN Controller

The SDN Controller is fundamental in enhancing network security by supporting diverse dynamic mitigation measures to address identified vulnerabilities or security risks.

Network segmentation through VLANs: An effective approach is using VLAN changes to isolate possibly compromised devices. The SDN controller can promptly isolate vulnerable devices by dynamically allocating devices to different VLANs according to their security risk, thereby mitigating the risk of threat spreading throughout the network.

Deep Packet Inspection (DPI): The SDN Controller may be configured to perform DPI on devices identified with low-risk vulnerabilities. This allows extensive traffic analysis, enabling the detection and filtration of potentially malicious data packets for the monitored devices.

Moving Target Defense (MTD): MTD techniques can be used to enhance the resilience of the network environment. This involves the regular alteration of network configurations, including IP addresses or routes, thereby complicating efforts for attackers to exploit identified vulnerabilities.

To implement each of these mitigation measures, it is necessary to adjust the Mitigation Measure Automation in the SOAR Platform so that it invokes the measure developed. Furthermore, it is necessary to ensure that the implementation of the mitigation measure is located in the appropriate place, either in the SDN Controller, if it involves actions directly related to the network, or in a new process to be run in the Security Tools Server, depending on the nature of the mitigation.

CHAPTER 4

Implementation

This chapter details the process of implementing the proposed system, describing the major elements and their interactions. Section 4.1 presents the logic of the system, explaining the location and interaction between the various components and services present in the system through the activity and deployment diagrams. The following sections provide a detailed explanation of each tool used in the implementation. Section 4.2 presents a detailed overview of the SOAR Platform used. Section 4.3 explains the Security Service Adapter. Section 4.4 outlines the process of device discovery within the network. In Section 4.5, the Vulnerability Scanner is discussed and its integration with the SOAR Platform is explained. Section 4.6 discusses the components of the SDN architecture. Section 4.7 focuses on the DHCP server, exploring its configuration. Finally, Section 4.8 presents the implementation of the mitigation measure.

4.1. System Logic

The Deployment Diagram is represented in Figure 4.1. It provides an overview of how the logical components of the system, previously outlined in Chapter 3, are deployed. It highlights the distribution of key services and applications, the interactions between them, and the communication protocols that are used. Components represented in blue have been created or their behavior modified. The implementation source code for this research is available in the associated GitHub repository [98].

Looking at the Deployment Diagram we can see that the VulnMatrix database is inside Kali Linux (equivalent to the Security Tools Server), as are the Dynamic Host Configuration Protocol (DHCP) Server, the Switch and the Software-Defined Networking (SDN) Controller. This configuration was adopted due to the lack of resources available in the deployment environment. The complete physical separation of the components would require greater hardware capacity, which was not available in the current context. It was therefore decided to group these components together in the Security Tools Server, ensuring that the system continues to operate effectively despite the sharing of resources. Table 4.1 summarizes the software options for the components and justifications for these choices are presented further on. Only one type of vulnerable device, Windows 2003, was used during testing due to its numerous known vulnerabilities and the ease of deployment it offers. While additional vulnerable machines could have been included, this specific choice was made to prioritize rapid setup and testing.

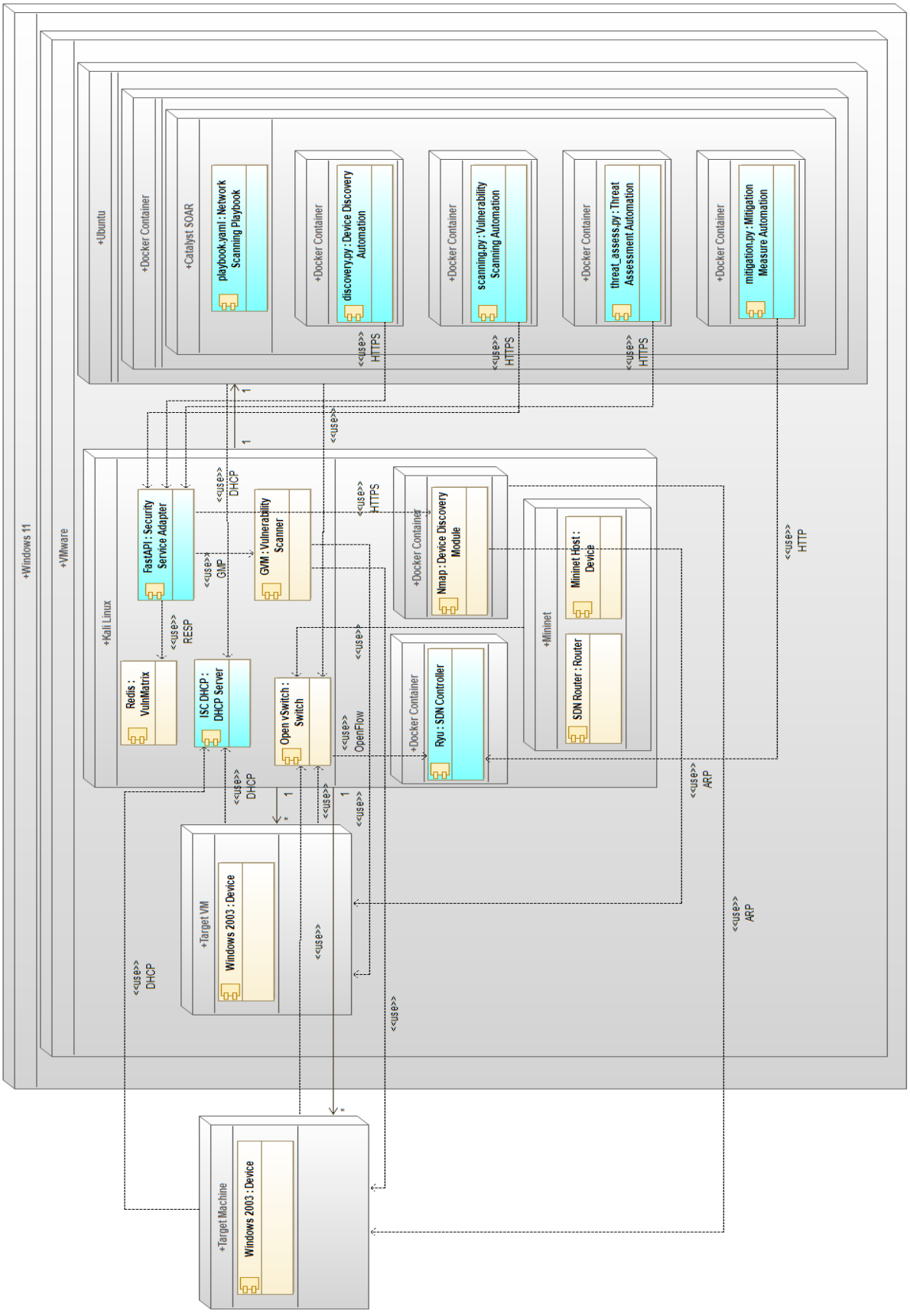

FIGURE 4.1. Deployment Diagram.

TABLE 4.1. Summary of software options.

Component	**Function**	**Host**	**Software/Version**
SOAR Server	Hosting SOAR Platform	Ubuntu VM	22.04.1
Security Tools Server	Hosting vulnerability scanning and network discovery tools	Kali Linux VM	2022.2
SDN Controller	Manages network flows and VLAN-based segmentation	Security Tools Server	Ryu 4.34, OpenFlow 1.3
SDN Switch	Forwards network traffic	Security Tools Server	Open vSwitch 3.1.0
SOAR Platform	Orchestration and automation	SOAR Server	Catalyst SOAR 0.10.3
VulnMatrix	Stores network device and vulnerability data	Security Tools Server	Redis 7.0.15
DHCP Server	Dynamic IP allocation to network devices	Security Tools Server	ISC-DHCP isc-dhcpd-4.4.3-P1
GVM	Vulnerability scanner	Security Tools Server	GVM 22.04
Nmap	Device discovery	Security Tools Server	Nmap 7.91
Vulnerable Devices	Network devices to be scanned	Virtual and physical machines	Various OSs (Windows 2003, etc.)

To gain a comprehensive understanding of the functioning of the proposed system, it is important to examine the sequence of activities that compose its workflow. Figure 4.2 depicts a comprehensive activity diagram that demonstrates the stages and the interactions among the components of the system.

In short, the process begins in the Security Orchestration, Automation, and Response (SOAR) by the creation of a network scanning ticket that requests a device discovery (1), thus initiating a device discovery task (2) from the Security Service Adapter (SSA). The system then checks for any active devices on the network (3) after sending the request. If no devices are detected, the process waits for a timeout period (4) before returning to the initial device discovery request, restarting the loop. After detection, the status of the device is checked in the database (5). This status refers to the date of the last vulnerability scan carried out on the device, and, based on this parameter, it is decided whether or not the device should be scanned (6). The SOAR then generates a vulnerability scanning ticket for each identified device (7) and starts the vulnerability scan (8). The scan results are analyzed (9) to calculate a severity score (10). Based on the calculated score, the SOAR initiates a Virtual Local Area Network (VLAN) change (i.e. Mitigation Measure 1) to relocate the device to a quarantine zone via the SDN controller (11). This mitigation strategy is just one of several possible approaches. Other measures (i.e. Mitigation Measure 2), such as Deep Packet Inspection (DPI) to analyze network traffic for malicious content, the use of firewall rules to block specific communication, or even Moving Target Defense (MTD) to dynamically shift the network configuration, could also be implemented. However, for the objectives of this dissertation, these alternatives were not explored, with the focus being on VLAN isolation as a proof of concept for automated containment. If the device is considered safe, the process concludes and a new scanning is scheduled in the future (12).

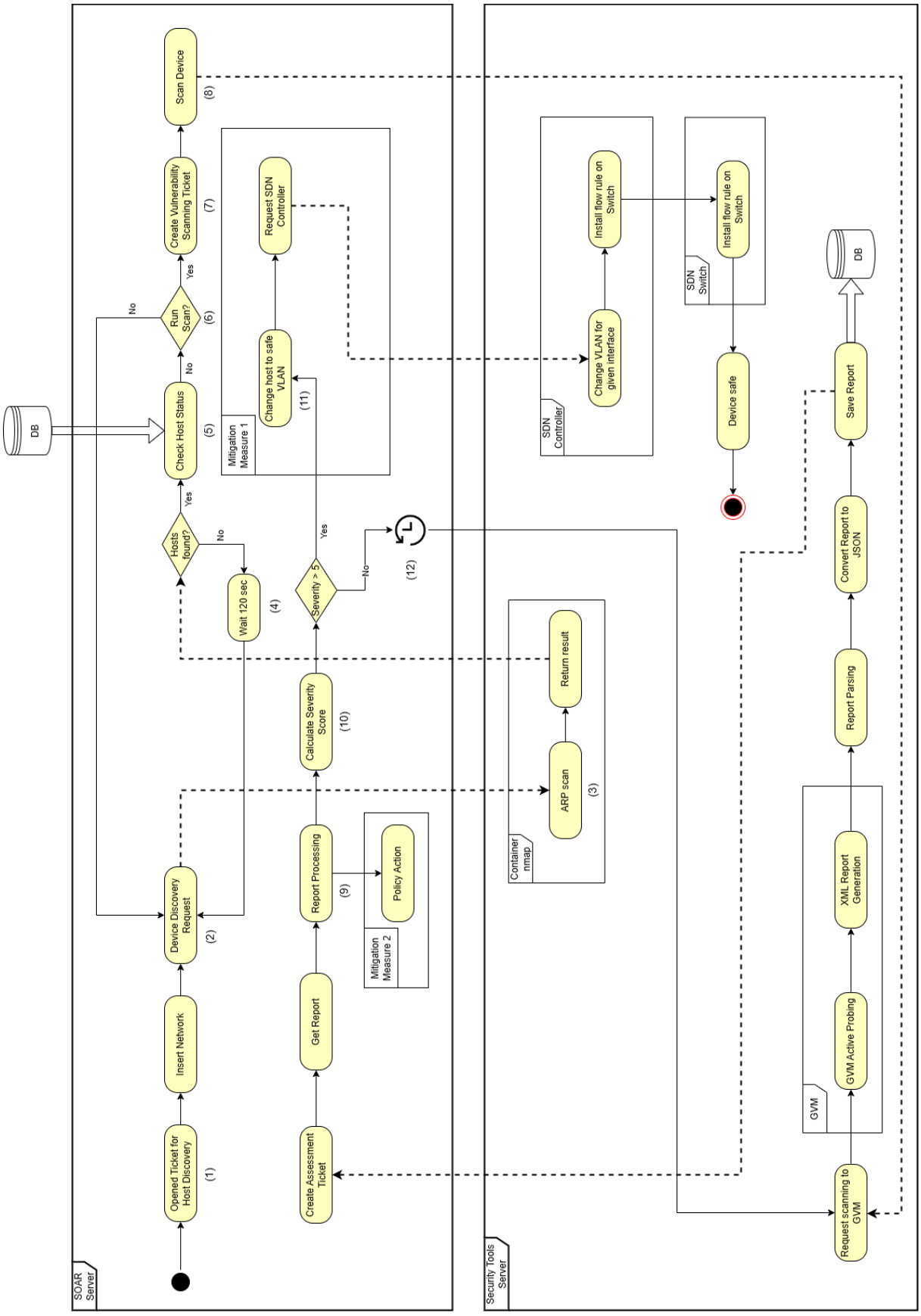

FIGURE 4.2. Activity Diagram.

4.2. SOAR Platform

This section presents the SOAR platforms explored throughout the project, starting with an look into TheHive, a tool that was initially considered but at the end not adopted due to its limitations in terms of the system's needs. The Catalyst SOAR platform is then detailed, explaining its features and why it was chosen for the implementation.

TheHive

TheHive Project [99], is an open-source platform for Security Incident Response (SIR). Through TheHive, it is possible to create tasks easily for team members, as well as offering a central point for displaying and cross-referencing findings. It is also possible to export the results of an investigation as a Malware Information Sharing Platform (MISP) event to help other collaborators detect and react to attacks that the user has dealt with. When TheHive is used in conjunction with Cortex [100], users can easily analyze many observables.

The choice not to use TheHive was due to the fact that, although it is a good SIR platform, it does not have the integrated automation features that were essential for the dissertation needs. Functions such as script development and playbooks are not natively available in TheHive. Although it would be possible to develop these features externally and integrate them into TheHive, this approach would significantly increase the complexity of the system. The goal for the chosen platform was for all the functionalities, including automation, to be carried out in an integrated way, without the need to resort to external tools or software.

Catalyst SOAR

Catalyst [101] is an open-source SOAR platform designed to automate alert handling and incident response processes. The platform is adaptable to various processes and workflows, allowing customization through ticket types, conditional custom fields, and playbooks to meet specific needs. Catalyst enables organizations to improve their operations and focus on more critical tasks, as well overall efficiency in managing security alerts.

Catalyst version 0.10.3 was chosen because of its support for creating customized automation scripts written in Python. With the ability to create detailed playbooks and execute processes remotely via API.

Playbooks proved to be a very interesting feature offered by Catalyst, since they can be created from scratch using the YAML programming language and shown graphically in the UI. The concept of the user interface is appealing, however, it could be object of further development, as generating more intricate playbooks appears to be very difficult. In contrast, YAML offers greater flexibility, yet it may be more engaging to construct playbooks using an alternative, more intuitive programming language that facilitates iterative processes, such as loops and which allows to return to a previous automation. As playbooks increase in complexity, monitoring modifications and detecting errors in the YAML code can prove challenging without an integrated debugging tool within Catalyst. The documentation on creating playbooks proved to be insufficient, requiring an understanding of how they work to be obtained by analyzing the examples provided and using a trial and error approach.

Writing automations offers numerous advantages, including the ability to write Python scripts, which provides an extensive array of options. In Catalyst, automation scripts are executed within containers, with each script running in its own Docker container. The Docker Python image has been modified to include essential libraries for needed functionalities. This method enhances compatibility and ensures process isolation, yet executing each automation script in an individual container presents some disadvantages. Each time a script is executed, a new container must be instantiated, resulting in increased overhead, latency and resources used.

Catalyst turned out to be a tool with a very interesting and promising concept. As has been said, its ability to improve automation while maintaining a traditional ticket opening and closing interface shows tremendous innovation compared to other open-source tools available. However, this software still has a long way to go before it can begin to be used in organizations. During the implementation of this tool, several obstacles and difficulties were encountered which will have to be taken into account for future updates:

- Catalyst API had several limitations. For example, it was necessary to make requests using a complete JSON object, in which the parameters had to be adjusted in the object itself, hardcoded style. For example, when creating a ticket via the API, the system did not automatically generate an ID, which required the user (application using the API) to generate an ID manually. In addition, it was not possible to know which was the last ID assigned without analyzing all the existing tickets, since there was no native function that allowed the last ID generated to be consulted directly.
- Another problem identified in Catalyst was the impossibility of manually closing a task while the playbook was running. This behavior resulted in the unnecessary execution of automations, even when it was no longer necessary or advisable to continue processing. The lack of this functionality not only increased system overload, but also created situations where continuing the playbook could result in unwanted consequences, especially on the Security Tools Server side, where sometimes the socket didn't close and the service had to be restarted.
- The debugging process in Catalyst proved to be extremely difficult, particularly due to the need to run an entire playbook in order to test a single automation script at the end of the chain. This procedure resulted in an inefficient development cycle.
- The tool's documentation was scarce and unclear, which resulted in significant difficulties during the process of installing, implementing and using it.

4.3. Security Service Adapter

The Security Service Adapter (SSA) was completely developed in Python and its purpose is to act as an intermediary between the SOAR Server and the security tools, Greenbone Vulnerability Manager (GVM) and Nmap. Through its REST API, the SSA receives instructions from Catalyst SOAR to execute processes related to the GVM, as well as store and read data from a database. Although the focus of this dissertation is not secure communication, the HTTPS protocol was adopted to improve the security of communications between the SSA and SOAR, thus ensuring the integrity and confidentiality of the data exchanged.

Development and Features

The decision logic for device vulnerability actions was implemented in SOAR, as it is the most appropriate place for this functionality. The SOAR's implementation allows more flexibility for users to adjust vulnerability analysis and mitigation processes, without compromising the overall operation of the system. In this way, SOAR takes responsibility for the decision logic, while the SSA remains focused on providing the necessary support functionality, ensuring a modular and scalable approach.

The scanning process begins when a request is made to scan a specific target, providing an IP address as input. This request activates the function responsible for starting the scanning process. In the activity diagram (Figure 4.2), this stage (8) is represented by the block that indicates the execution of a scan task until the scanner generates a report containing the vulnerabilities found. During the execution of a scan, it is crucial to monitor progress in order to know when it has been completed. In the code, this monitoring is carried out by a function that periodically checks the status of the scan via the identifier of the generated report. This continuous monitoring is done by periodic checks, where the system queries the current status (i.e. "Running", "Stopped", "Done") of the scan returned by the GVM. The check interval used was 2 minutes, adjustable via Device Discovery Automation in SOAR, according to the needs of the environment. There are two approaches to implementing this monitoring: by callback or through periodic checks. The callback method would allow the GVM to automatically notify the system as soon as the scan is complete, however, this solution would be more complex to implement. On the other hand, periodic scanning, which was the approach adopted, performs queries at adjustable intervals. Smaller intervals offer the advantage of faster detection of scan completion, but uses more resources by issuing more queries. Longer intervals reduce this system burden, but it can delay the detection at the end of the scan, impacting the scanning response time. Once the device scan is complete, the system must process the scanning results, extracting the relevant information and storing it for later analysis. In the code, this stage is started as soon as the monitoring indicates that the scan has finished. The function responsible for collecting the report data is called, and it extracts the relevant information, such as severity and Quality of Detection (QoD). The report extraction process is designed to go through the XML data generated by GVM, converting it into JSON format that can be sent for analysis by the SOAR Platform. Therefore, the SSA had to read the report before sending it and collect all the relevant information that could be used for SOAR to make decisions. This implementation gives the flexibility of customizing and adjusting the report data been offered to the main goals behind the system, including specific particularities of the devices being scanned.

API

FastAPI [102] was selected as the library for the development of the SSA API. This Python library was selected for its simplicity and performance. FastAPI allows SOAR to quickly access the functionalities offered by the SSA, such as running scans and extracting reports. The choice of this technology was reinforced by its use in network automation scenarios, as emphasized in [103].

Database

Redis [104] was used to manage the work queues and temporary storage of data critical to the system's operation. In the SSA, tasks such as device discovery and report parsing, are queued in Redis to be processed asynchronously. This allows multiple tasks to be processed in parallel, reducing the response time to new requests. In this way, the SSA can continue to receive and queue new requests for scans or other operations without having to wait for each previous individual scan to be completed.

Various options to Redis were tested to enhance the implementation, such as the OpenCTI platform which was tested to assess its ability to complement the solution deployment. OpenCTI is a open-source platform that enables organizations to effectively handle and control their knowledge and data related to Cyber Threat Intelligence (CTI). The purpose of its creation is to systematically arrange, store, categorize, and present both technical and non-technical data related to cyber threats. However, its focus is not on actively responding to vulnerabilities, which is the central objective of this dissertation.

4.4. Device Discovery Module

Nmap (Network Mapper) [105] is an open-source software used for exploring and evaluating network systems for both exploration and security purposes, including network inventory, and monitoring the availability of devices or services. Although its primary purpose is to rapidly scan large networks, it is also effective for targeting individual devices [106]. The efficiency and performance of Nmap are highly recognized, offering rapidly and extensive network scans.

In this dissertation deployed system, Nmap (version 7.93) is running inside a Docker container, providing an isolated and consistent setup for detecting devices. To write the program, "python-nmap" library was used which further increases its versatility, allowing seamless integration and automation scripts.

The Nmap scan type and other options used for discovering devices on the network are explained below:

- sP: this option tells Nmap to only perform a scan using ping (device discovery), and then display the available devices that responded to the scan. No additional tests (such as port scanning and OS detection) are performed [107].
- PR: tells Nmap to execute a Address Resolution Protocol (ARP) scan. When Nmap attempts to send a raw IP packet, such as an Internet Control Message Protocol (ICMP) echo request, the operating system (OS) needs to determine the destination hardware Media Access Control (MAC) address associated with the target IP address in order to correctly address the ethernet frame. Alternatively using a ARP scan if Nmap receives a response, it can skip the IP-based ping packets because it already has confirmation that the device is active. This makes ARP scanning significantly faster and more reliable compared to scans based on IP addresses. ARP scanning overloads less the network than the ping method [107].

Another feature of Nmap is to execute scans using various scripts. These scripts allow to improve Nmap's adaptability and capabilities by leveraging its built-in Nmap Scripting Engine (NSE), which was investigated to see if it could be a viable option for my vulnerability scanning. The NSE is equipped with an extensive assortment of pre-installed scripts, which are categorized based on a predefined list of categories. One of these is the vulnerability detection, called "vuln". These scripts check for specific known vulnerabilities and generally only report results if they are found.

Despite the advanced capabilities of the NSE and its collection of scripts, its use for vulnerability detection was not selected over Greenbone Vulnerability Manager (GVM). The reason for this is that NSE is limited to the detection of specific vulnerabilities, based on predefined scripts, which are constantly changed by the community. The NSE's efficacy is constrained by the restricted quantity of available scripts, despite its use in certain audits and targeted exploitation contexts. Consequently, the quantity of tests conducted on the device diminishes, resulting in a lower probability of identifying possible vulnerabilities in comparison to GVM. Alternatively to NSE, GVM also offers a more complete interface, which facilitates automation and continuous monitoring of the security status of the network.

4.5. Vulnerability Security Scanner

The purpose of the Vulnerability Security Scanner is to perform vulnerability scans on the different devices in the network. The used scanning tool was the GVM [108] which is the successor to OpenVAS.

GVM uses two scanners to scan the devices on the network: OpenVAS Scanner and Notus Scanner. The OpenVAS Scanner is a full-featured engine that relies on individual vulnerability tests (VTs) executed one by one, primarily using Nessus Attack Scripting Language (NASL) scripts, against target systems. It is designed to be comprehensive, leveraging either the Greenbone Enterprise Feed or the Greenbone Community Feed for up-to-date vulnerability data. However, this approach can be resource-intensive and slower, as each NASL-based local security check (LSC) is processed individually for every device. In contrast, the Notus Scanner offers improved performance by replacing the NASL LSC logic. Instead of executing individual scripts, it performs a bulk comparison of the installed software on a device against a consolidated list of known vulnerable software. This approach significantly reduces system resource consumption and scanning time, making the Notus Scanner faster and more efficient, particularly for LSCs.

Scan Configurations

GVM offers various scan configurations, adapted to different needs and detail levels. Each configuration aims to optimize the balance between the depth of analysis and the potential impact on target systems. These options allow users to choose between a quick and safe approach, or a more exhaustive and potentially disruptive one, depending on the environment and the purpose of the scanning.

- Full and Fast: For most environments, this is the best option for start scans. This analysis configuration is based on the information gathered in the previous port scan and uses almost all Vulnerability Tests (VTs). Only VTs that do not damage the target system are used. The VTs are optimized as best as possible in order to maintain a particularly low false negative rate. The other “Full” settings only provide more value in rare cases, but with considerably more effort.
- Full and Fast Ultimate: This scan configuration expands the "Full and Fast" configuration with VTs that can interrupt services or systems, or even cause them to stop. This configuration may not always be absolutely reliable, depending on environmental conditions, which may increase the false positive rate. Identifying suspicious false positives may require manual analysis and the definition of exceptions.
- Full and Very Deep: Is based on "Full and Fast", but the results of port scanning or application/service detection have no impact on the selection of VTs. This means that, regardless of the ports that are open or the services identified, the scanner selects VTs that assess vulnerabilities in applications/services that have not previously been detected. In this way, the configuration seeks to identify potential security flaws, even without prior knowledge of the services running. A scan with this configuration is very slow.
- Full and Very Deep Ultimate: Expands the "Full and Very Deep" configuration with dangerous VTs, which can cause possible service or system interruptions. An analysis with this configuration is very slow. As with the "Full and Fast Ultimate", this configuration may not be completely reliable, depending on the environmental conditions, which can result in an increase in the false positive rate. In these cases, it may be necessary to carry out a manual analysis and define exceptions for suspected false positives.

Given the context of this dissertation, the Full and Fast scan configuration was chosen for the tests described in Chapter 5, due to its efficiency and ability to guarantee fast and accurate results without causing operational damage to the target system.

Report

Upon completion of the scan, it is necessary to extract the report. GVM enables data exports in formats including PDF, CSV, and XML. XML was chosen for this dissertation due to its ability to encapsulate all pertinent scan details and facilitate straightforward processing by other tools. GVM XML files are comprehensive, encompassing numerous fields, some of which are exclusively for internal system use. For practical purposes, it is advisable to focus on the report’s critical data fields rather than all available information. The report data are presented as follows.

- Report Identification:
  - Report ID
  - Creation Time
- Scan Information:

  - Scan Name
  - Progress (%)
  - List of Scanned Devices
  - List of Ports
- Results Data:
  - Highest Severity Value (0-10)
  - Result Count (total vulnerabilities detected)
  - List of Results:
    * Result Name
    * Vulnerability Description
    * Threat Level (Log, Low, Medium, High)
    * Severity (0-10)
    * Device IP and Port Number
    * Quality of Detection (QoD), indicating reliability
    * VT OID (Object Identifier)
- VT Details:
  - Tags: containing summary, impact, solution, etc.
  - External References: CVEs, CERTs, and additional URLs

The data in the report are used to determine whether the device is vulnerable, as shown in Equation 4.1. Given that the system is designed to generate a report per device, the key data used are the severity of each vulnerability and its QoD. While GVM assigns a severity score to the device, the SOAR system performs its own assessment based on the vulnerabilities detected, which aligns with one of the objectives.

$$V = \max\left(S_i \times \frac{QoD_i}{100}\right) \quad \forall i \in \{1,2,\ldots,n\}, \quad V > 5 \tag{4.1}$$

Where:

- V is the highest vulnerability score.
- S_i is the severity score of vulnerability i.
- QoD_i is the quality of detection value of vulnerability i.
- n is the total number of vulnerabilities.
- The operator max selects the highest score.
- The condition $V > 5$ means the device is considered vulnerable if the severity score exceeds 5.

The equation calculates the highest vulnerability score for each device, factoring in the severity and QoD. Severity is the parameter closest to measuring the risk of vulnerability and this is multiplied by QoD/100 to account for the reliability of the vulnerability detected, which leads to a more accurate assessment. If the severity score V is greater than 5, the device is deemed vulnerable. The decision value 5 was choosen because it is in the middle of the range

of the severity score. However, this value can be easily changed in the Mitigation Measure Automation script.

GVM Integration

GVM offers the capability to perform tasks remotely, such as initiating a scan and retrieving its report, thus allowing it to be managed by the SSA. To do the automated communication with GVM a toolkit named gvm-tools was installed. It was decided to use gvm-script, one of the tools available, which provides a simple and efficient method for interacting with the GVM by executing Greenbone Management Protocol (GMP) commands directly through the command line. This interface, which is easy to understand and use, enables to carry out intricate tasks like starting vulnerability scans, selecting scan configurations, and obtaining comprehensive reports using concise, single-line commands. This approach mitigates the complexity commonly associated with API-based interactions. Furthermore, the gvm-script's reliance on scripting makes it simple to maintenance. Through the utilization of scripts, it was successfully implemented a distinct partition of responsibilities. This partitioning enables more convenient updates and alterations, as modifications to the scanning logic implemented within the scripts. This not only simplifies the integration process but also minimizes the likelihood of bugs and errors that are commonly associated with intricate code. In addition, this approach improves the process of testing and debugging. This is because each script can be verified separately, ensuring a dependable and predictable operation.

4.6. SDN Architecture

Controller

Ryu [109] is an open-source, component-based SDN controller developed by Nippon Telegraph and Telephone (NTT), a Japanese telecommunications company. In Japanese, the term "Ryu" translates to "flow," which accurately represents the controller's ability to dynamically manage network traffic flows. Ryu is a Python-based software that is licensed under Apache 2.0. Ryu is compatible with various network management protocols, including NETCONF, OF-Config, and Open vSwitch Database Management Protocol, in addition to the popular OpenFlow protocol, which it supports up to version 1.5. The controller went through rigorous testing and received certification for its compatibility with a range of OpenFlow switches via their southbound interface. This implementation used the OpenFlow 1.3 protocol, which ensured integration with the Open vSwitch switch. Ryu offers an extensive range of libraries for handling packets and supports multiple tunneling and encapsulation techniques, such as VLAN.

The selection of Ryu for our system was motivated by its open-source nature, comprehensive documentation, and implementation in Python, which is in line with my experience. Moreover, Ryu's capacity to seamlessly incorporate with various platforms and tools through its REST API (northbound interface), along with its strong community backing and adaptable structure, made it a fine option for constructing a resilient SDN environment. In this implementation, the Ryu controller runs inside a Docker container. This method offers a stable and

isolated environment, guaranteeing that the controller works reliably regardless of the underlying system setup.

Switch

Open vSwitch (OVS) [110] is a high-quality, multi-layered switch that is licensed under the open-source Apache 2.0 license. The purpose of its design is to facilitate extensive network automation, while also maintaining compatibility with standard management interfaces and protocols. It is compatible with various virtualization platforms, including Docker and VMware. OVS is designed for use in virtualization deployments involving multiple servers. These environments are typically defined by their constantly changing endpoints, the preservation of logical abstractions, and the incorporation or transfer of tasks to specialized switching hardware. OVS version 3.1.0 was the one used for this implementation.

A simple network topology was also set up using Mininet [111], an emulator that generates a virtual network for testing and development. The Mininet 2.3.0 environment was initially used to carry out preliminary tests and verify the proper functionality of the tools installed with the configured topology. In addition, Mininet enabled the creation of an SDN router within one of the Mininet hosts, providing Internet access and facilitating switching between different subnets as needed. This virtual test environment was essential for simulating different network scenarios and evaluating the system's behavior under controlled conditions, before proceeding with implementation in more complex and real environments.

4.7. DHCP Server

The delivery of the DHCP service is accomplished through the implementation of the ISC DHCP server [112]. The ISC DHCP server version isc-dhcpd-4.4.3-P1 was chosen due to its stability and simplicity of use. It is appropriate for the implementation because it ensures reliable operation and has a straightforward configuration process that meets the system needs without unnecessary complexity. It is important to note that, at the time of publication of this dissertation, ISC DHCP, although still supported, will not be the subject of further maintenance releases and has been succeeded by Kea [113].

Following the installation, the DHCP server was configured by editing the /etc/dhcpd.conf file (see Listing 4.1). The configuration parameters and options specified in this file determine the DHCP server's Internet Protocol (IP) address assignment within existing subnets, the range of assignable IP addresses, and the specific network parameters (such as subnet mask, gateway and lease times) that clients will receive.

A limitation encountered in the implementation was that it was not possible to shut down the switch port when changing the device's VLAN, so the client's IP address remained unchanged and VMware could not accurately detect the status of the port for unknown reasons. To solve this problem, a configurable lease period was implemented allowing for shorter leases. This ensured that IP addresses were distributed more frequently, facilitating a more fast approach to managing IP address allocation. By adopting this method, customers can be sure of receiving new IP addresses for their new VLANs in shorter periods of time.

The two subnets specified in the file correspond to the production (192.168.13.0/24) and the quarantine (192.168.14.0/24) networks. The IP address of the SOAR Server has been configured statically to ensure reliable and consistent access to the SOAR Platform.

```
1  default-lease-time 60;
2  max-lease-time 60;
3  min-lease-time 60;
4
5  subnet 192.168.13.0 netmask 255.255.255.0 {
6     option routers 192.168.13.6;
7     range 192.168.13.6 192.168.13.254;
8          INTERFACES="rt-eth0";
9  }
10
11 subnet 192.168.14.0 netmask 255.255.255.0 {
12    option routers 192.168.14.6;
13    range 192.168.14.6 192.168.14.254;
14         INTERFACES="rt-eth1";
15 }
16
17 host soar_host {
18         hardware ethernet 00:0c:29:35:14:fc;
19         fixed-address 192.168.13.161;
20 }
```

LISTING 4.1. DHCP configuration.

4.8. Mitigation

In the context of this dissertation, the mitigation measure implemented was changing the VLAN to isolate vulnerable devices. To do this, it was necessary to change the behavior of the controller, i.e. its source code.

Regarding Figure 4.2, when a device is considered vulnerable, the SDN Controller is responsible for modifying the VLAN (11) for the device's interface and implementing specific flow rules on the switch to enforce network policies. The SDN Controller effectively isolates the susceptible device by modifying its network route, ensuring that it is relocated to a quarantine VLAN.

Listing 4.2 displays the algorithm executed in the SDN controller to enable the creation of VLANs in the network. This method offers some benefits, as the VLANs are dynamically adjusted based on the switch port, allowing for simple and straightforward modifications. By incorporating Ryu's REST API, VLANs changes can be automated and performed by the SOAR. The function uses two dictionaries to make this work: port_to_vlan (line 2) and ip_to_switch_port (line 3). The port_to_vlan dictionary helps keep track of which VLAN ID is

assigned to each port, so the controller can be aware of these assignments in real time. Meanwhile, ip_to_switch_port links IP addresses to their respective switch IDs and source ports. When the function is given an IP address (line 8) and a new VLAN ID (line 9), it starts by checking if the IP is already in the ip_to_switch_port dictionary. If it's there, it retrieves the switch ID and source port values associated with the given IP (line 14). If the IP isn't listed, the function returns a 404 error (line 16), indicating the IP address isn't found. Once the switch ID and port are known, the function updates the port_to_vlan dictionary with the new VLAN ID for that port (line 18). After that, it clears out any old flow rules for the port to prevent conflicts (line 20), ensuring the new VLAN configuration is applied correctly. Finally, it provides the updated port_to_vlan information (line 22), reflecting the latest VLAN assignments.

```
1  Dictionaries:
2  - port_to_vlan: Keeps track of which VLAN each port is assigned to
3  - ip_to_switch_port: Determines which port a given IP address is associated with
4
5  Function: Change VLAN ID
6
7  Input:
8  - ip_address: IP address of the device whose VLAN ID needs to be changed
9  - vlan_id: New VLAN ID to be assigned
10
11 Steps:
12 1. Check if the IP address is present in the ‘ip_to_switch_port‘ dictionary.
13    - If the IP address is found:
14      - Retrieve the switch ID and source port from the ‘ip_to_switch_port‘
            dictionary.
15    - If the IP address is not found:
16      - Return a 404 (Not Found) response.
17
18 2. Update the ‘port_to_vlan‘ dictionary with the new VLAN ID for the source
       port.
19
20 3. Clean all the flow rules associated with the source port using the
       ‘clean_flows‘ method.
21
22 4. Return the updated ‘port_to_vlan‘ dictionary as the response.
```

LISTING 4.2. Change VLAN algorithm.

CHAPTER 5

Results

This chapter presents and discusses the proposed system's results, focusing on the research questions outlined in this dissertation. Section 5.1 tests the system's ability to analyze the resources consumed by the vulnerability scanner and the impact on system scalability (RQ2) and Section 5.2 discusses how quickly mitigation measures are applied (RQ4). These tests are important to validate the system's operation in different scenarios, as well as to verify that the system can meet the established objectives successfully. In addition, the focus on resources consumed and speed of response ensures that the solution does not compromise the performance of the system or the network, evaluating its viability and scalability in more complex environments.

Tests were conducted in two different environments to comprehensively evaluate the system's performance in different scenarios. During the implementation of Virtual Local Area Networks (VLANs) in the Software-Defined Networking (SDN) controller, it was observed that, in physical networks, the association of VLANs with single physical ports, which connect the traditional switch to the DHCP server, affects the individual separation of devices by VLAN. Given this particularity, the virtual environment was used to test the implementation of the mitigation measure, where it was possible to adequately simulate the change of VLAN. On the other hand, the physical environment was essential for testing the vulnerability scanner and validating its operation in a real environment, where interaction with network devices, specifically identified as hosts in this context, could be verified more reliably. Using the physical environment, it was also possible to investigate my solution scalability in a more comprehensive way.

5.1. Performance and Scan Duration

The tests were carried out in the computer networks laboratories of Iscte - University Institute of Lisbon [114]. These are intended to show the performance of the vulnerability scanner incorporated into the developed system.

The laboratories had 32 machines at disposal, spread over 16 workbenches in a total of two rooms. Conducting tests involving different host numbers offered an extensive evaluation of the scanner's scalability and performance across diverse loads. By gradually increasing the number of hosts allowed an analysis of how resource use and scan duration correlate with the number of devices being scanned.

Each machine in the laboratory ran a Virtual Machine (VM) that was targeted to be scanned. Table 5.1 shows the specifications of the physical machines and the target VMs.

TABLE 5.1. Laboratory Machine and Target VM Specifications.

Specification	Laboratory Machine	Target VM
OS	Windows 10 Enterprise	Windows 2003
CPU	Intel i5-6600	8 CPU Cores
RAM	32 GB	1 GB

The Security Tools Server physical machine hosts a VM with Kali Linux, where the vulnerability scanner, GVM (Greenbone Vulnerability Manager), is running. This server is connected to the lab's switch to access the network. Within the VM, the DHCP server is responsible for assigning IP addresses to the laboratory's machines (see Figure 4.1). Table 5.2 shows the specifications of the server physical machine and the server VM settings.

TABLE 5.2. Server Machine and Server VM Specifications.

Specification	Server Machine	Server VM
OS	Windows 11 Home	Kali Linux 2022.2
CPU Model	AMD Ryzen 5 5600H	8 CPU Cores
RAM	24 GB	12 GB

Testing the impact of the vulnerability scanner in terms of resource consumption, like CPU and RAM, ensures that scanning operations do not impair performance or cause unexpected downtime. The network bandwidth used during the scanning process was also measured to ensure that the scanner does not overload the network, compromising communication between hosts or decreasing the efficiency of other services. CPU, RAM and bandwidth were the parameters monitored in the tests presented below. The intent was to see if there were any parameters that can be identified as limiting the solution performance, correlating with an increase in the duration of the scanning process.

5.1.1. Scan Duration

The tests conducted seek to measure the resources used by the GVM and their impact on the duration of the scans. The time taken to complete the scan has an impact on how often scans can be executed and how quickly vulnerabilities can be identified and mitigated. Large scan durations can delay the detection of vulnerabilities, increasing the exposure to external risks.

Tests were carried out with 1, 2, 4, 8, 16 and 32 hosts in the laboratory environment. Figure 5.1 illustrates the scanning duration with regard to the number of hosts scanned. The testing was conducted in a private network environment with no competing traffic. Only the traffic generated by the test itself was present, ensuring that the results exclusively reflect the performance

and behavior of the system under controlled conditions. The scan configuration used during the tests was "Full and Fast."

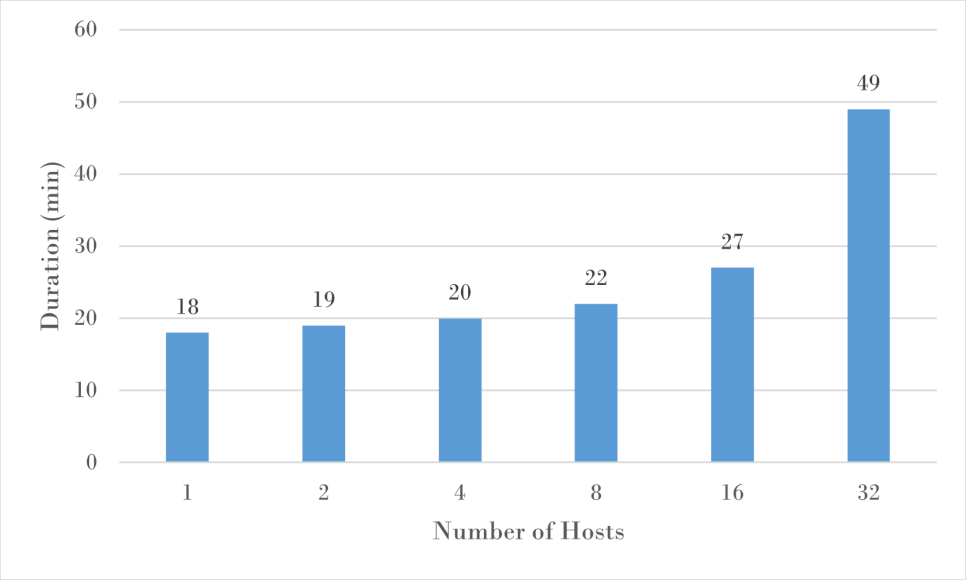

FIGURE 5.1. Duration of scanning as a function of the number of hosts scanned.

The results showed an approximately exponential increase in scan duration as the number of hosts increased mainly after 8 scanned hosts. This increase in duration can be explained by the fact that the GVM performs a series of complex Vulnerability Tests (VTs) on each host. As the number of hosts increases, the scanner must deal with a proportionally larger volume of data and tasks. These results will be clarified in the following subsections 5.1.2, 5.1.3 and 5.1.4 where the resources used by the scanner will be analyzed in a more comprehensive way. In this way, any external limiting factors that could cause a delay in the duration of the scans could be better identified.

5.1.2. CPU Usage

In this subsection the CPU usage during the different tests will be discussed. Table 5.3 shows the maximum, average and standard deviation CPU usage values during the tests carried out for different numbers of hosts scanned. The standard deviation reflects the dispersion of the usage values in relation to the average, indicating the consistency of the results over time. CPU utilization values were calculated over a representative sample. The CPU's initial state was 1%. The scan configuration used during the tests was "Full and Fast."

For a number of hosts equal to or less than 2, the maximum CPU usage remains moderate, while the average CPU usage is low. Although for 4 hosts the maximum CPU usage reaches almost 100%, the increase in the duration of the scanning is relatively low, adding only around two minutes compared to the test with a single host (see Figure 5.1). This behavior can be explained by the average CPU usage percentage, which has not yet reached critical values, allowing the system to continue operating with a certain margin of efficiency (see Table 5.3). However, from 8 hosts onwards, we observe the first complete saturation of the CPU, with a

substantial increase in average usage, indicating a significant overload on the server's resources. For 16 hosts, the impact on scan duration becomes more evident. The CPU not only reaches its maximum value, but also remains saturated for longer periods, compared to the previous tests. In the 32 hosts scenario, the situation worsens, since the average CPU value reaches almost 90%, which shows a critical overload. This high average utilization leads to a reduction in the scanner's efficiency, which is unable to serve all the machines optimally, resulting in a significantly longer time to complete the scan.

TABLE 5.3. CPU usage results.

	1 Host	**2 Hosts**	**4 Hosts**	**8 Hosts**	**16 Hosts**	**32 Hosts**
CPU Max (%)	27.5	58.6	99.0	100	100	100
CPU Avg (%)	14.3	24.3	47.8	78.0	82.1	87.7
STD Deviation (%)	6.7	13.9	29.0	35.2	32.9	27.0

Analyzing the standard deviation of the tests shown in Table 5.3, there is a significant variation in the consistency of CPU usage as the number of hosts increases. For 1 and 2 hosts, the standard deviation is relatively low, indicating that CPU usage over time has been stable and little dispersed around the mean. The stability of resource usage is more predictable in scenarios of lower load. However, for 4 and 8 hosts, there is a considerable increase in the standard deviation, reflecting greater variability in CPU consumption, which suggests that, in these scenarios, the system experienced peaks in resource utilization. This indicates that although average CPU usage is high, its variation over time is less significant, suggesting that the system remains consistently overloaded, with few moments of relief.

These results suggest an exponential growth in CPU demand as the number of hosts increases, highlighting the scalability challenges faced when running vulnerability scans on larger networks.

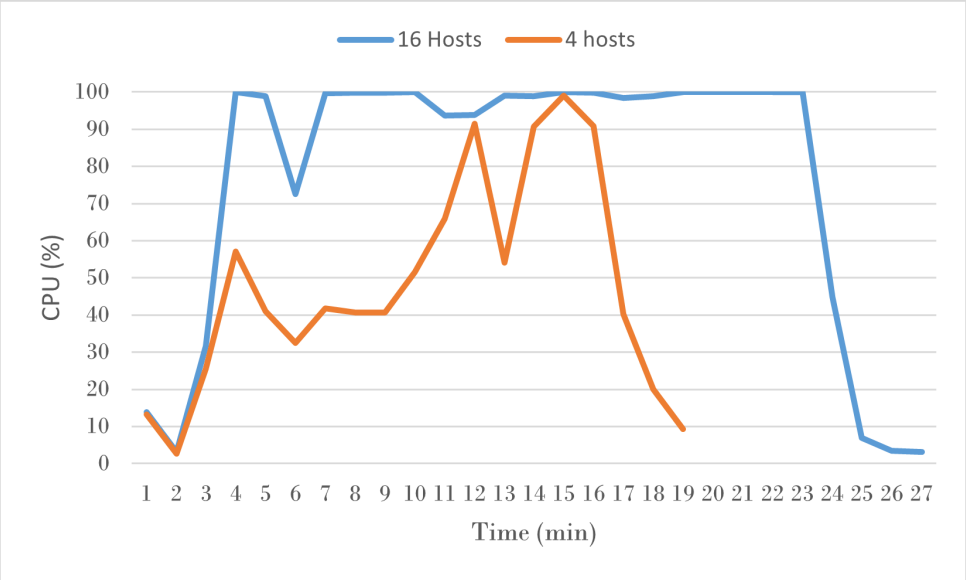

FIGURE 5.2. CPU usage for 4 and 16 hosts throughout the scanning process.

Figures 5.2 illustrates CPU usage over time for respectively the loads of 4 and 16 hosts. In both scenarios, initially, utilization peaks at approximately 13%, attributable to the task's creation in the GVM. Subsequently, regarding the plot for 16 hosts, CPU usage increases dramatically, particularly at the fourth minute, attaining 100%. This behavior may signify the rigorous execution of VTs. Subsequent to these peaks, a significant decline in CPU usage occurs at six minutes, with additional reductions evident at eleven minutes. At the end, utilization commences to diminish, as anticipated with the conclusion of the scan. On the other hand, in the 4 hosts plot, the CPU only saturated once at minute fifteen, managing to CPU usage quite well over time. The fluctuations in CPU usage are likely associated with the VTs executed at specific times, indicating the intensity and complexity of the operations conducted. Comparing both CPU more intense usage durations, one can identify the scan scenario of 16 hosts needs more time, around 7 minutes, to conclude than the other scenario with 4 hosts, which is consistent with the results in Figure 5.1.

5.1.3. RAM usage

In this subsection the RAM usage during the different tests will be discussed. The results shown in Table 5.4 and Figure 5.3 indicate that the RAM usage does not vary significantly with the increase in the number of hosts being scanned. While there is a slight increase in RAM consumption as more hosts are added, the changes are relatively minor and do not exhibit the same exponential growth pattern observed in scanning time. This suggests that GVM is relatively efficient in managing memory, even as the workload scales up, implying that CPU is likely the primary factor contributing to the increased scanning duration. RAM usage values were calculated over a representative sample. The initial RAM Memory was 37% (4.3 GB). The scan configuration used during the tests was "Full and Fast."

TABLE 5.4. RAM usage results.

	1 Host	**2 Hosts**	**4 Hosts**	**8 Hosts**	**16 Hosts**	**32 Hosts**
RAM Max (%)	41.6	42.3	43.2	46.7	50.4	52.9
RAM Avg (%)	41.1	41.6	42.1	43.4	44.4	44.9
STD Deviation (%)	0.5	0.6	0.8	1.7	2.7	2.8

The analysis of the RAM usage over time, Figure 5.3, did not show any significant outliers, which proves the consistency of the results throughout the tests performed. This steady behavior allows us to conclude that no anomalous or unexpected phenomena occurred during the runs, thus guaranteeing the reliability of the tests. The absence of abrupt fluctuations in RAM usage suggests that the system maintained predictable performance, even in variable load scenarios.

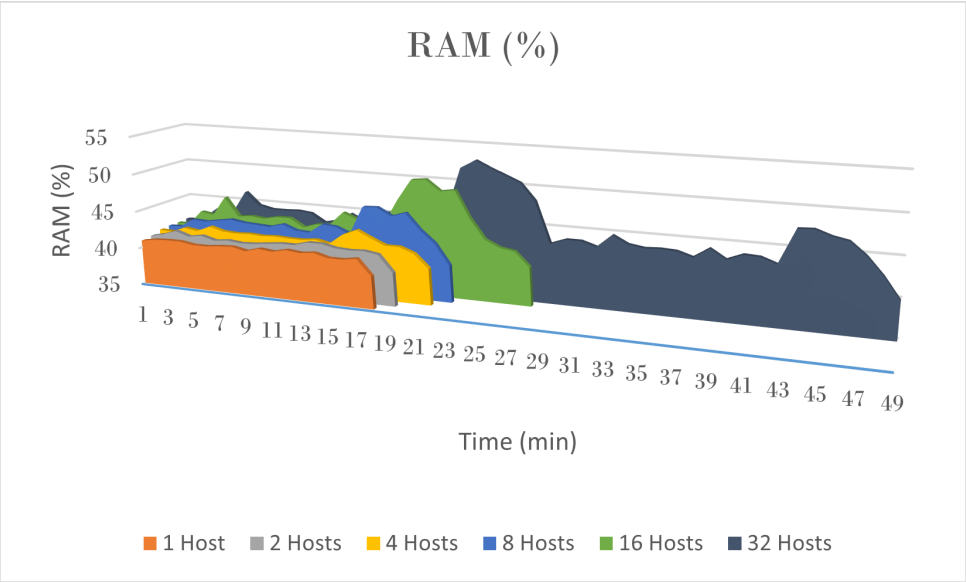

FIGURE 5.3. Comparison of RAM usage across different tests during the scanning process.

5.1.4. Bandwidth usage

In this subsection the network bandwidth spent during the different tests will be discussed. Figure 5.4 shows the bandwidth as a function of the number of hosts. The upload bandwidth is persistently lower to the download, indicating a higher volume of data received relative to the data transmitted. Subsequent to an initial rise, the bandwidth utilization growth curve begins to stabilize once the host count surpasses 8. The stabilization can be attributed to the constraint of 100% CPU demand, which affects the scanner's capacity to process and execute tests concurrently. The growth in bandwidth for uploads and downloads exhibits a correlation with the number of hosts, although downloads demonstrate a more pronounced increase. The average bandwidth usage was calculated based on a representative sample. The initial average bandwidth values were 0.7296 Kbps for upload and 1.0203 Kbps for download. The scan configuration used during the tests was "Full and Fast."

Although the values increase over time, they are not high enough for bandwidth to be considered a limiting factor when it comes to vulnerability scanning. I have used a LAN with a capacity of 100 Mbps which is significantly higher than the highest value of Figure 5.4, i.e. approx. 2 Mbps (2% of available network resources being used by the scanning process). Given the low traffic rates observed during these tests, it was not expected that the network, even with a different scan configuration, would become a bottleneck for the scanning process.

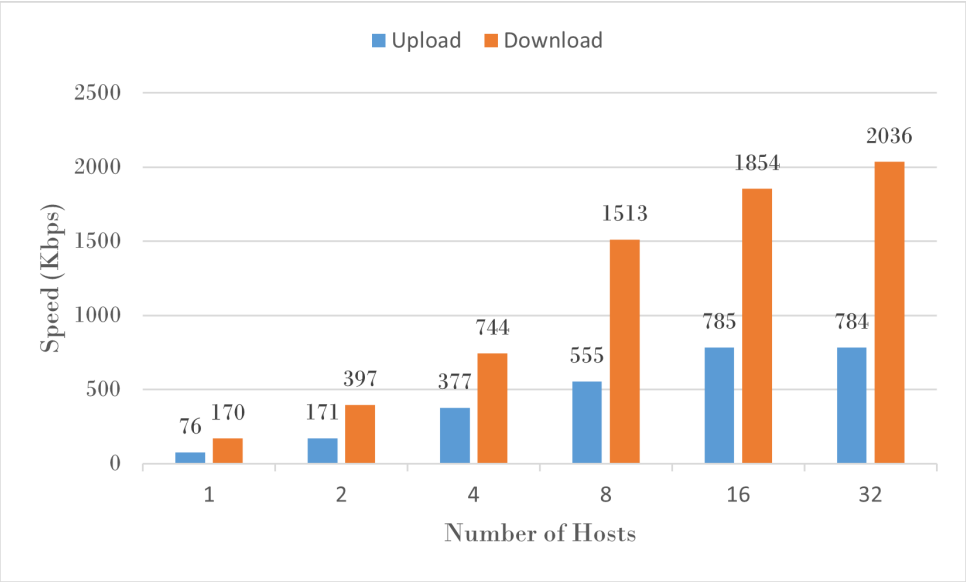

FIGURE 5.4. Comparison of average download and upload bandwidth usage as a function of the number of hosts scanned.

5.1.5. Conclusions

With regard to the tests conducted in the laboratory, it can be concluded that GVM consumes a lot of CPU resources when scanning more than eight hosts simultaneously, which can significantly impact the duration of the scanning process. The configuration used proved to be suitable for fixed computer networks, such as in office environments, but could prove problematic in more dynamic networks with many hosts joining and leaving, as happens in public networks. To mitigate these problems, a viable solution would be to limit the number of hosts to be analyzed simultaneously to a predefined value and put the rest on a job queue. Another option would be to modify the network architecture, restricting access to network assets to new devices only, or opt for a scanner with a lighter configuration, capable of carrying out an initial preliminary inspection of hosts before a more detailed examination. Or even add another instances of the scanner to share the load among them.

5.2. Task Execution Time Analysis

The tests performed in the virtual environment consisted of running vulnerability scans on one host with the Windows 2003 operating system (Table 5.1) inside VMware. The purpose of these tests is to measure the execution times of each task, using a logging system implemented both in SOAR Server and the Security Tools Server, allowing the exact moment at which the system advanced to the next step in the workflow to be tracked and verified (see Figure 4.2).

Table 5.5 shows the time spent on each of these steps. By analyzing these times in detail, it is possible to identify steps that may be consuming excessive time. The first measured time "Device Discovery" corresponds to the interval between the initial request to discover devices on the network and receiving a response. The second column "Prepare Vuln Scan" of the table refers to the time taken to prepare the vulnerability scan request, which includes tasks such as placing the device in the queue and creating an automation ticket, until the request is finally

submitted. After that, the vulnerability scan itself is performed, for a single host, as shown in Figure 5.1, which lasts 18 minutes and is not shown in the table. The “Request Report” column indicates the interval between sending the request for the vulnerability report and receiving it, while the “Parsing Report” column measures the time taken to process the report. Finally, the “Change VLAN” column represents the time it took the system to change the VLAN, which is described in Table 5.6.

TABLE 5.5. Measured steps times.

Device Discovery	**Prepare Vuln Scan**	**Request Report**	**Parsing Report**	**Change VLAN**
1,5 sec	0,98 sec	0,016 sec	0,003 sec	Table 5.6

The communication time between the SDN Controller and the Switch during the VLAN change process was also measured. To do this, the time interval between sending the FLOW_MOD message and receiving its ACK was analyzed. However, discrepancies were identified in the times recorded during message capture. In one scenario, the controller sent three FLOW_MOD messages, each corresponding to a specific command (ADD, DELETE and ADD), resulting in a time of 0.000075302 seconds. In another scenario, all the headers were encapsulated in a single message, resulting in a time of 0.000024276 seconds. In both cases, it was concluded that these times are insignificant and do not represent any kind of relevant delay in the system, as would be expected.

Looking at the Table 5.5, we can see that once the mitigation measure is known, it takes 19 ms to implement it. This means that in this case the host is protected almost instantly, which accomplishes the goal. Although there was secondary problem identified during the tests in the virtual environment was the time taken to change the VLAN at the host’s level so it could have connection. It was intended that, after changing the VLAN, the host’s network interface would be disconnected and reconnected. However, although the VMware virtual switch detected the interface as inactive, it didn’t take any further action. As a result, the DHCP server did not assign a new IP address to the host on the new VLAN. To mitigate this problem, the DHCP server’s lease time was adjusted and reduced to ensure that the IP was assigned more quickly on the new network. As noted in the previous paragraph the VLAN change was executed "instantly" at the SDN controller, isolating the vulnerable host after the controller applied OpenFlow rules to the topology switch. However, the host was unable to communicate until the new IP was assigned. This adjustment to the lease time made it possible to reduce the impact of this limitation.

Two tests were performed, with lease times of one and five minutes, each with 40 measurements to determine the average time spent in the lease renewal process. Theoretically, the average time is expected to be half of the total lease time, since the VLAN change request is made randomly throughout the lease cycle. According to the uniform distribution of the waiting time, the average time remaining until the lease expires should converge to this value. An important aspect to consider in the tests was to ensure that the VLAN change request was made at a random time within the lease interval. If the next request was made immediately after the first (and so on), knowing that the machine would only obtain a new IP after the lease time had

expired, the time measured would always be equivalent to the total lease time, which would distort the results. It was therefore necessary to ensure that the request was made at random times throughout the lease cycle, in order to ensure that the values obtained were consistent and representative of correct sampling. Table 5.6 shows the results obtained, where we can see that the theoretical value is within the standard deviation.

TABLE 5.6. Measured lease times.

Lease Time (sec)	**Average (sec)**	**STD Deviation (sec)**
60	38	8
300	212	62

5.2.1. Conclusions

The assessments conducted in the virtual environment demonstrated the system responsiveness, without delays throughout the workflow stages. The vulnerability detection and mitigation process was conducted with critical steps including device discovery, preparation for vulnerability scanning, and report generation demonstrating minimal execution times. The only stage that exhibited a notably extended duration was the vulnerability scan, lasting approximately 18 minutes for a single host, which is expected given the task's complexity. The mitigation measure, specifically a VLAN change, was executed almost instantaneously at the SDN controller level which means that the architecture can effectively isolate susceptible hosts in a timely manner.

CHAPTER 6

Conclusions and Future Work

This chapter presents the conclusions drawn from this dissertation and proposes future work.

6.1. Conclusions

This dissertation explored the proactive and automated detection and mitigation of vulnerabilities within network environments, managed by Software-Defined Networking (SDN) alongside various open-source tools to address these challenges. The research aimed to automate manual procedures to improve the efficiency and responsiveness of network security operations using a Security Orchestration, Automation and Response (SOAR) platform that orchestrates the different technologies involved. Furthermore, this dissertation assessed the effects of these strategies on network and device performance, ensuring their timely execution.

The current document has comprehensively revised the available literature for both detection and mitigation of system vulnerabilities, and their associated risks. From the Literature Review, the most prominent proposals were identified, classified, discussed, and compared. Resulting from the analysis made on the surveyed work, it was identified a large list of future interesting research directions. These are summarized: (1) a strong need on incorporating active probing tools into SDN to quickly identify vulnerabilities; (2) prioritizing mitigation efforts effectively, and adopt proactive instead reactive countermeasures; (3) the lack of risk indicators in data plane mitigation reveals the need for a more sophisticated method of evaluating and addressing security concerns. Considering the control plane, (4) although several studies mention the use of risk indicators, the existing literature lacks complete solutions that demonstrate measurable effects on network performance; (5) the literature also revealed a limited number of studies focusing on both vulnerability detection and mitigation; (6) while some automated processes were identified there was absence of integrating a SOAR to coordinate the various tools employed; (7) the developed systems were resistant to modifications or the incorporation of new detection or mitigation features, which is undesirable considering the continuous evolution of threats. Further developments are also needed in programmable systems supported by decision consensus among distributed agents for better protecting the system and its sensitive data against sophisticated security threats at the network periphery.

With the knowledge obtained from the Literature Review, it was possible to focus the research on answering some of the gaps presented by the studies related to the control plane. The main steps that the system would have to take in order to meet the proposed objectives were thus defined. The process begins with identifying active devices on the network, followed by assessing their vulnerabilities, analyzing the results to determine suitable mitigation measures,

and finally carrying out those measures. With the aim of guaranteeing the successful implementation of the system, it was ensured that the system met the quality attributes of interoperability, proactivity and adaptability. Interoperability: the system integrates effectively with tools such as Nmap and GVM through SOAR. SOAR orchestrates communication between these tools and enables workflow automation, ensuring that information on devices and vulnerabilities is collected and processed in a coordinated manner. To this end, the Security Service Adapter offers an interface that allows SOAR to communicate transparently and in an automated way with the tools in the Security Tools Server. Proactivity: the system is designed to perform automated and continuous scans to detect vulnerabilities in real time and carry out mitigation measures. For instance, when it identifies a vulnerable device, the system can automatically relocate that device to a quarantine VLAN before it can be exploited and compromises the network. Adaptability: a modular architecture has been implemented that allows for changes and expansions with minimal impact on its overall design. This flexibility makes it easier to introduce new tools, replace existing components (such as the vulnerability scanner), or implement new mitigation measures. The modularity of the system means it remains dynamic and responsive, ready to adapt to both new threats and changes in security policies or requirements without compromising its operation.

An architecture orchestrated by a SOAR was designed and implemented to detect and mitigate vulnerabilities in the network. Catalyst SOAR was the SOAR Platform chosen, demonstrating that its integration with the different security technologies was a flexible and customizable solution, allowing the creation of playbooks and automations in a simple fashion, despite some highlighted limitations that should be improved in future versions. Integration with SOAR has proved advantageous not only in terms of orchestration with tools such as the GVM vulnerability scanner, but also in centralizing decision-making regarding the mitigation measures to be applied. In the context of this dissertation, the mitigation measure implemented was to change the VLAN to isolate vulnerable devices, so when a device is considered vulnerable, the SDN controller is responsible for changing the VLAN for the device interface and implementing specific flow rules on the switch to apply network policies. The architecture has been designed so that the implementation of new mitigation measures can be done easily. Table 6.1 answers the research questions relating to methodology (RQ1 and RQ3).

Tests were performed to confirm the system's functionality in various scenarios and to ensure that it can successfully and proactively achieve its intended objectives. To achieve this, the tests were carried out in two different environments: laboratory and virtual. In the laboratory, it was observed that GVM requires substantial CPU resources when scanning more than 8 devices simultaneously. For instance, average CPU usage ranged between 14% and 48% for scans involving 1 to 4 devices, escalating from 78% to 88% when handling between 8 and 32 devices. RAM consumption remained consistent across 1 to 32 devices, with average values varying between 41% and 45%. Average bandwidth usage showed values of 76 Kbps (TX) and 170 Kbps (RX) for scans with 1 device, increasing to 784 Kbps (TX) and 2036 Kbps (RX) for

32 devices. The tests performed in the virtual environment exhibited the system's responsiveness, with no delays occurring during the workflow stages. The consumption of CPU resources was reflected in the increase in scan duration. It was observed that for a single device, the scan lasted 18 minutes, extending by just 4 minutes for the test with 8 devices. However, with the longer CPU saturation periods, the time increased to 27 minutes with 16 devices and rose to 49 minutes with 32 devices. The sole phase that demonstrated a significantly prolonged delay was the vulnerability scan, which is anticipated due to the task's complexity. The mitigation measure, specifically a VLAN change, was implemented almost immediately at the SDN controller level, lasting just 19 ms, indicating that the architecture can efficiently isolate vulnerable devices promptly. The configuration used proved to be suitable for stable networks, such as in networks for production environments, but could prove problematic in more dynamic networks with many devices joining and leaving, such as in developing environments. Table 6.1 answers the research questions relating to the tests (RQ2 and RQ4).

The dissertation achieved its objectives by investigating and implementing a system capable of detecting vulnerabilities, mitigating the associated risks and effectively orchestrating the tools involved. By integrating SDN, open-source tools and the SOAR Platform, it was possible to automate the entire detection and mitigation process, answering the research questions posed. The system developed proved capable of identifying vulnerabilities in a timely manner, applying one mitigation measure by isolating vulnerable hosts in a isolated VLAN, and ensuring the efficient orchestration of all components.

TABLE 6.1. Research Questions and Answers.

Research Question	Answer
RQ1 - How to automate device security vulnerabilities detection on networks?	The system automates vulnerability detection using Nmap for device discovery and GVM to detect security vulnerabilities. Nmap regularly scans the network to identify active devices and their IP addresses. GVM then periodically scans each device for vulnerabilities. The SOAR Platform orchestrates these stages, automating workflows to consistently manage both discovery and analysis. See Chapter 4.
RQ2 - What resources the automated vulnerability detection consumes and their impact on the system scalability?	The vulnerability scanner shows acceptable RAM and bandwidth utilization, but consumes significant CPU resources when scanning multiple devices simultaneously. In the tests, it was found that the scanner's performance degrades as the number of devices analyzed increases, especially when more than 8 devices are scanned at the same time. This increases the time required to complete the scan. See 5.1 results.
RQ3 - How the usage of the SDN controller capabilities can enhance the network security?	The SDN controller improves network security by enabling real-time management of network resources, such as isolating vulnerable devices by changing VLANs. Its integration with the SOAR Platform allows these actions to be automated, potentially improving vulnerability analysis and mitigation response times and enabling faster implementation of mitigation measures. See Chapter 4.
RQ4 - How timely can be the automated deployment of mitigation strategies?	The entire process is fast, with the vulnerability scanning being the longest delay, depending on the device and the number of devices being scanned. See 5.1.1 results. The VLAN change occurs almost instantly in the SDN Controller, isolating the vulnerable device. The IP lease renewal time caused delays, which can be reduced by adjusting the DHCP lease time. Nevertheless this doesn't affect security. See 5.2 results.

6.2. Future Work

Given that this dissertation is exploratory in nature and taking into account the limited time and obstacles encountered during its development, below are presented some suggestions for future work. Appendix A discusses in more detail the Open Issues left by the Literature Review with regard to proactive security measures in next-generation networked systems.

Applying more mitigation measures: The application of more mitigation measures is necessary to guarantee a robust defense. The system developed in this dissertation was designed to be flexible, allowing new mitigation measures to be integrated easily. Deep packet inspection (DPI) is a complementary measure to VLAN isolation, especially for devices with low-risk vulnerabilities. Through DPI, the network traffic of these devices can be monitored in real time until they are fixed, allowing malicious activity to be detected without completely blocking access to network resources. Implementing Moving Target Defense (MTD) as a mitigation measure could be used in situations where a vulnerable device does not need to be isolated immediately. The device could be moved to a Virtual LAN where MTD is applied, the IP address

of the device could be changed periodically, and the architecture would provide a less intense intermediate level of quarantine. Another possible approach is to analyze the Common Vulnerability Scoring System (CVSS) vector in detail and interpret its metrics to provide valuable information that enables the application of more appropriate mitigation measures for vulnerable devices.

Node Classification and Scanning: The SDN controller can be enhanced to classify nodes based on the context of the network topology, based on their proximity to critical assets or the service they provide, would make it possible to prioritize scanning on devices that perform critical functions on the network, such as servers or devices with privileged access. In addition, the implementation of customizable configuration options for vulnerability scans could be explored. For example, perform less aggressive scans during periods of high activity, reducing the impact on device and network performance, and reserve more aggressive scans for periods of lower activity. Another use would be to use the information gathered about the device's location on the network and the number of connections it has and adjust the aggressiveness and periodicity of the scans accordingly.

References

[1] A. Fleck. "Infographic: Cybercrime expected to skyrocket in coming years." (2024), [Online]. Available: https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/. (Accessed Dec. 14, 2024).

[2] J. Polónio, J. Moura, and R. Neto Marinheiro, "On the road to proactive vulnerability analysis and mitigation leveraged by software defined networks: A systematic review," *IEEE Access*, vol. 12, pp. 98 546–98 566, 2024. DOI: 10.1109/ACCESS.2024.3429269.

[3] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *Journal of Network and Computer Applications*, vol. 67, pp. 1–25, May 2016, ISSN: 10958592. DOI: 10.1016/j.jnca.2016.03.016.

[4] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, pp. 45–77, 3 Dec. 2007, ISSN: 07421222. DOI: 10.2753/MIS0742-1222240302.

[5] J. vom Brocke, A. Hevner, and A. Maedche, "Introduction to design science research," pp. 1–13, 2020. DOI: 10.1007/978-3-030-46781-4_1.

[6] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of systems and software*, vol. 80, no. 4, pp. 571–583, 2007, ISSN: 0164-1212. DOI: 10.1016/j.jss.2006.07.009.

[7] Parsifal. "Perform systematic literature reviews." (), [Online]. Available: https://parsif.al/. (Accessed Dec. 14, 2024).

[8] H. Sabino, R. V. Almeida, L. B. de Moraes, *et al.*, "A systematic literature review on the main factors for public acceptance of drones," *Technology in Society*, vol. 71, p. 102 097, 2022, ISSN: 0160-791X. DOI: 10.1016/j.techsoc.2022.102097.

[9] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1505–1521, 2022. DOI: 10.1109/TSC.2022.3174475.

[10] M. Kaur and R. Aron, "A systematic study of load balancing approaches in the fog computing environment," *The Journal of supercomputing*, vol. 77, no. 8, pp. 9202–9247, 2021. DOI: 10.1007/s11227-020-03600-8.

[11] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. Mounir, "A comprehensive survey on sdn security: Threats, mitigations, and future directions," *Journal of Reliable*

Intelligent Environments, vol. 9, no. 2, pp. 201–239, 2023. DOI: 10.1007/s40860-022-00171-8.

[12] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang, "Data-driven cybersecurity incident prediction: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1744–1772, 2019. DOI: 10.1109/COMST.2018.2885561.

[13] N. Sun, M. Ding, J. Jiang, *et al.*, "Cyber threat intelligence mining for proactive cybersecurity defense: A survey and new perspectives," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1748–1774, 2023. DOI: 10.1109/COMST.2023.3273282.

[14] M. Ghaznavi, E. Jalalpour, M. A. Salahuddin, R. Boutaba, D. Migault, and S. Preda, "Content delivery network security: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2166–2190, 2021. DOI: 10.1109/COMST.2021.3093492.

[15] E. Bardhi, M. Conti, R. Lazzeretti, and E. Losiouk, "Security and privacy of ip-icn coexistence: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2427–2455, 2023. DOI: 10.1109/COMST.2023.3295182.

[16] W. Li, W. Meng, and L.-f. Kwok, "A survey on openflow-based software defined networks: Security challenges and countermeasures," *J. Netw. Comput. Appl.*, vol. 68, pp. 126–139, 2016. DOI: 10.1016/j.jnca.2016.04.011.

[17] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015. DOI: 10.1109/COMST.2015.2474118.

[18] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015. DOI: 10.1109/JPROC.2014.2371999.

[19] P. Bosshart, D. Daly, G. Gibb, *et al.*, "P4: Programming protocol-independent packet processors," *ArXiv*, vol. abs/1312.1719, 2013. DOI: 10.1145/2656877.2656890.

[20] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing sdn from openflow to p4: A survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–37, 2023, ISSN: 0360-0300. DOI: 10.1145/3556973.

[21] FIRST. "Cvss v3.1 specification document." (), [Online]. Available: https://www.first.org/cvss/v3.1/specification-document. (Accessed Dec. 14, 2024).

[22] MITRE. "Cve website." (), [Online]. Available: https://www.cve.org/. (Accessed Feb. 10, 2024).

[23] V. Pham and T. Dang, "Cvexplorer: Multidimensional visualization for common vulnerabilities and exposures," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1296–1301. DOI: 10.1109/BigData.2018.8622092.

[24] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Understanding passive and active service discovery," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 57–70. DOI: 10.1145/1298306.1298314.

[25] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack graph-based moving target defense in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1653–1668, 2020. DOI: 10.1109/TNSM.2020.2987085.

[26] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1909–1941, 2020. DOI: 10.1109/COMST.2020.2982955.

[27] A. Chowdhary, S. Pisharody, and D. Huang, "Sdn based scalable mtd solution in cloud network," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, 2016, pp. 27–36. DOI: 10.1145/2995272.2995274.

[28] L. Spitzner, "The honeynet project: Trapping the hackers," *IEEE Security & Privacy*, vol. 1, no. 2, pp. 15–23, 2003. DOI: 10.1109/MSECP.2003.1193207.

[29] W. Fan, Z. Du, D. Fernández, and V. A. Villagra, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3906–3919, 2017. DOI: 10.1109/JSYST.2017.2762161.

[30] R. T. El-Maghraby, N. M. Abd Elazim, and A. M. Bahaa-Eldin, "A survey on deep packet inspection," in *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, IEEE, 2017, pp. 188–197. DOI: 10.1109/ICCES.2017.8275301.

[31] P. Krongbaramee and Y. Somchit, "Implementation of sdn stateful firewall on data plane using open vswitch," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, IEEE, 2018, pp. 1–5. DOI: 10.1109/JCSSE.2018.8457354.

[32] P. Mehran, E. A. Reza, and B. Laleh, "Spkt: Secure port knock-tunneling, an enhanced port security authentication mechanism," in *2012 IEEE Symposium on Computers & Informatics (ISCI)*, IEEE, 2012, pp. 145–149. DOI: 10.1109/ISCI.2012.6222683.

[33] Z. Niu, Q. Li, C. Ma, H. Li, H. Shan, and F. Yang, "Identification of critical nodes for enhanced network defense in manet-iot networks," *IEEE Access*, vol. 8, pp. 183 571–183 582, 2020. DOI: 10.1109/ACCESS.2020.3029736.

[34] A. N. Alhaj and N. Dutta, "Analysis of security attacks in sdn network: A comprehensive survey," *Contemporary Issues in Communication, Cloud and Big Data Analytics: Proceedings of CCB 2020*, pp. 27–37, 2022. DOI: 10.1007/978-981-16-4244-9_3.

[35] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2015. DOI: 10.1109/COMST.2015.2487361.

[36] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (ddos) attacks in sdn and cloud computing environments," *IEEE Access*, vol. 7, pp. 80 813–80 828, 2019. DOI: 10.1109/ACCESS.2019.2922196.

[37] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2015. DOI: 10.1109/COMST.2015.2453114.

[38] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in sdn: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102 595, 2020. DOI: 10.1016/j.jnca.2020.102595.

[39] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Computers & security*, vol. 53, pp. 79–108, 2015. DOI: 10.1016/j.cose.2015.05.006.

[40] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Pérez-Díaz, and M. Zareei, "Towards security automation in software defined networks," *Computer Communications*, vol. 183, pp. 64–82, 2022. DOI: 10.1016/j.comcom.2021.11.014.

[41] M. Yu, J. Zhuge, M. Cao, Z. Shi, and L. Jiang, "A survey of security vulnerability analysis, discovery, detection, and mitigation on iot devices," *Future Internet*, vol. 12, no. 2, p. 27, 2020. DOI: 10.3390/fi12020027.

[42] R. M. Ogunnaike and B. Lagesse, "Toward consumer-friendly security in smart environments," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops*, 2017, pp. 612–617. DOI: 10.1109/PERCOMW.2017.7917633.

[43] T. Eom, J. B. Hong, S. An, J. S. Park, and D. S. Kim, "A systematic approach to threat modeling and security analysis for software defined networking," *Ieee Access*, vol. 7, pp. 137 432–137 445, 2019. DOI: 10.1109/ACCESS.2019.2940039.

[44] Y. Nikoloudakis, E. Pallis, G. Mastorakis, C. X. Mavromoustakis, C. Skianis, and E. K. Markakis, "Vulnerability assessment as a service for fog-centric ict ecosystems: A healthcare use case," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 1216–1224, 2019. DOI: 10.1007/s12083-019-0716-y.

[45] S. Kim, S. Yoon, J.-H. Cho, *et al.*, "Divergence: Deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4834–4846, 2022. DOI: 10.1109/TNSM.2021.3139928.

[46] M. Zolotukhin, T. Hämaläinen, and R. Immonen, "Curious sdn for network attack mitigation," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, 2021, pp. 630–635. DOI: 10.1109/QRS-C55045.2021.00096.

[47] R. Sood and S. S. Kang, "Detection and mitigation of network vulnerability through scheduling framework in software defined networks," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, 2019, pp. 1118–1123.

[48] H. Do Hoang, V.-H. Pham, *et al.*, "Empirical study on reconnaissance attacks in sdn-aware network for evaluating cyber deception," in *2021 RIVF International Conference*

on Computing and Communication Technologies (RIVF), IEEE, 2021, pp. 1–6. DOI: 10.1109/RIVF51545.2021.9642134.

[49] L. M. Almutairi and S. Shetty, "Generalized stochastic petri net model based security risk assessment of software defined networks," in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, IEEE, 2017, pp. 545–550. DOI: 10.1109/MILCOM.2017.8170813.

[50] Q. Qin, K. Poularakis, and L. Tassiulas, "A learning approach with programmable data plane towards iot security," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2020, pp. 410–420. DOI: 10.1109/ICDCS47774.2020.00064.

[51] S.-Y. Chang, Y. Park, and A. Muralidharan, "Fast address hopping at the switches: Securing access for packet forwarding in sdn," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2016, pp. 454–460. DOI: 10.1109/NOMS.2016.7502843.

[52] S.-Y. Chang, Y. Park, and B. B. A. Babu, "Fast ip hopping randomization to secure hop-by-hop access in sdn," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 308–320, 2018. DOI: 10.1109/TNSM.2018.2889842.

[53] E. O. Zaballa, D. Franco, Z. Zhou, and M. S. Berger, "P4knocking: Offloading host-based firewall functionalities to the network," in *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, 2020, pp. 7–12. DOI: 10.1109/ICIN48450.2020.9059298.

[54] I. Pali and R. Amin, "Portsec: Securing port knocking system using sequence mechanism in sdn environment," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, IEEE, 2022, pp. 1009–1014. DOI: 10.1109/IWCMC55113.2022.9824343.

[55] A. Saxena, R. Muttreja, S. Upadhyay, K. S. Kumar, and U. Venkanna, "P4filter: A two level defensive mechanism against attacks in sdn using p4," in *2021 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, IEEE, 2021, pp. 113–118. DOI: 10.1109/ANTS52808.2021.9937010.

[56] J. Li, H. Jiang, W. Jiang, J. Wu, and W. Du, "Sdn-based stateful firewall for cloud," in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, IEEE, 2020, pp. 157–161. DOI: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00037.

[57] A. Javadpour, F. Ja'fari, T. Taleb, and M. Shojafar, "A cost-effective mtd approach for ddos attacks in software-defined networks," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, IEEE, 2022, pp. 4173–4178. DOI: 10.1109/GLOBECOM48099.2022.10000603.

[58] H. Galadima, A. Seeam, and V. Ramsurrun, "Cyber deception against ddos attack using moving target defence framework in sdn iot-edge networks," in *2022 3rd International Conference on Next Generation Computing Applications (NextComp)*, IEEE, 2022, pp. 1–6. DOI: 10.1109/NextComp55567.2022.9932172.

[59] K. Guo, Y. Gao, D. Wang, H. Zhi, T. Zhang, and Y. Lu, "A hybrid routing mutation mechanism based on mutation cost and resource trustworthiness in network moving target defense," in *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*, IEEE, 2022, pp. 426–431. DOI: 10.1109/DSC55868.2022.00065.

[60] T. Moghaddam, M. Kim, J.-H. Cho, *et al.*, "A practical security evaluation of a moving target defence against multi-phase cyberattacks," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2022, pp. 103–110. DOI: 10.1109/DSN-W54100.2022.00026.

[61] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Nelson, and H. Lim, "Poster: Address shuffling based moving target defense for in-vehicle software-defined networks," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–3. DOI: 10.1145/3300061.3343392.

[62] M. F. Hyder and M. A. Ismail, "Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches," *IEEE Access*, vol. 9, pp. 21 881–21 894, 2021. DOI: 10.1109/ACCESS.2021.3055577.

[63] F. S. D. Silva, T. Pascoal, E. P. Neto, R. S. Nunes, C. H. Souza, and A. Neto, "An adaptive moving target defense approach for software-defined networking protection," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2023, pp. 1–6. DOI: 10.1109/NOMS56928.2023.10154278.

[64] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J.-H. Cho, and T. J. Moore, "Frvm: Flexible random virtual ip multiplexing in software-defined networks," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, 2018, pp. 579–587. DOI: 10.1109/TrustCom/BigDataSE.2018.00088.

[65] J. Narantuya, S. Yoon, H. Lim, *et al.*, "Sdn-based ip shuffling moving target defense with multiple sdn controllers," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*, IEEE, 2019, pp. 15–16. DOI: 10.1109/DSN-S.2019.00013.

[66] S. Chiba, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "Design of a network scan defense method by combining an sdn-based mtd and ips," in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2021, pp. 273–278. DOI: 10.23919/APNOMS52696.2021.9562686.

[67] B. Zhang, L. Han, and S. Sun, "Dynamic random route mutation mechanism for moving target defense in sdn," in *2021 6th International Symposium on Computer and Information Processing Technology (ISCIPT)*, IEEE, 2021, pp. 536–541. DOI: 10.1109/ISCIPT53667.2021.00114.

[68] M. Kim, J.-H. Cho, H. Lim, *et al.*, "Evaluating performance and security of a hybrid moving target defense in sdn environments," in *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2022, pp. 276–286. DOI: 10.1109/QRS57517.2022.00037.

[69] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev, "Frequency-minimal moving target defense using software-defined networking," in *2016 international conference on computing, networking and communications (ICNC)*, IEEE, 2016, pp. 1–6. DOI: 10.1109/ICCNC.2016.7440635.

[70] C. Gudla and A. H. Sung, "Moving target defense application and analysis in software-defined networking," in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, 2020, pp. 0641–0646. DOI: 10.1109/IEMCON51383.2020.9284847.

[71] C. Gudla and A. H. Sung, "Moving target defense discrete host address mutation and analysis in sdn," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2020, pp. 55–61. DOI: 10.1109/CSCI51800.2020.00017.

[72] S. Macwan and C.-H. Lung, "Investigation of moving target defense technique to prevent poisoning attacks in sdn," in *2019 IEEE World Congress on Services (SERVICES)*, IEEE, vol. 2642, 2019, pp. 178–183. DOI: 10.1109/SERVICES.2019.00050.

[73] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim, "Optimal network reconfiguration for software defined networks using shuffle-based online mtd," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, IEEE, 2017, pp. 234–243. DOI: 10.1109/SRDS.2017.32.

[74] T. Moghaddam, G. Yang, C. Thapa, S. Camtepe, and D. D. Kim, "Poster: Toward intelligent cyber attacks for moving target defense techniques in software-defined networking," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023, pp. 1022–1024. DOI: 10.1145/3579856.3592825.

[75] D. P. Sharma, J.-H. Cho, T. J. Moore, F. F. Nelson, H. Lim, and D. S. Kim, "Random host and service multiplexing for moving target defense in software-defined networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761496.

[76] M. M. Islam, Q. Duan, and E. Al-Shaer, "Specification-driven moving target defense synthesis," in *Proceedings of the 6th ACM Workshop on Moving Target Defense*, 2019, pp. 13–24. DOI: 10.1145/3338468.3356830.

[77] A. Chowdhary, A. Alshamrani, D. Huang, and H. Liang, "Mtd analysis and evaluation framework in software defined network (mason)," in *Proceedings of the 2018 ACM*

International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2018, pp. 43–48, ISBN: 9781450356350. DOI: 10.1145/3180465.3180473.

[78] H. Kim, E. Hwang, D. Kim, *et al.*, "Time-based moving target defense using bayesian attack graph analysis," *IEEE Access*, vol. 11, pp. 40 511–40 524, 2023. DOI: 10.1109/ACCESS.2023.3269018.

[79] M. Ge, J.-H. Cho, D. Kim, G. Dixit, and I.-R. Chen, "Proactive defense for internet-of-things: Moving target defense with cyberdeception," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 1, pp. 1–31, 2021. DOI: 10.1145/3467021.

[80] Z. Rehman, I. Gondal, M. Ge, H. Dong, M. Gregory, and Z. Tari, "Proactive defense mechanism: Enhancing iot security through diversity-based moving target defense and cyber deception," *Computers & Security*, vol. 139, p. 103 685, 2024. DOI: 10.1016/j.cose.2023.103685.

[81] F. Shi, Z. Zhou, W. Yang, S. Li, Q. Liu, and X. Bao, "Ahip: An adaptive ip hopping method for moving target defense to thwart network attacks," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, 2023, pp. 1300–1305. DOI: 10.1109/CSCWD57460.2023.10152746.

[82] L. Zhang, Q. Wei, K. Gu, and H. Yuwen, "Path hopping based sdn network defense technology," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, IEEE, 2016, pp. 2058–2063. DOI: 10.1109/FSKD.2016.7603498.

[83] J. H. Anajemba, N. Ababneh, Y. Hamid, A. Chowhan, O. Obinna, and E. Vajzovic, "Sdn-based port hopping technique for mitigating network attacks," in *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, IEEE, 2023, pp. 1–6. DOI: 10.23919/SoftCOM58365.2023.10271570.

[84] Z. Minjiao, M. Yufeng, W. Bo, and Q. Zhang, "A dynamic deceptive honeynet system with a hybrid of virtual and real devices," in *2022 5th International Conference on Computing and Big Data (ICCBD)*, IEEE, 2022, pp. 113–117. DOI: 10.1109/ICCBD56965.2022.10080304.

[85] W. Huang, Y. Sun, W. Ou, and Y. Wang, "A flow scheduling model for sdn honeypot using multi-layer attack graphs and signaling game," in *2021 7th International Conference on Computer and Communications (ICCC)*, IEEE, 2021, pp. 2012–2020. DOI: 10.1109/ICCC54389.2021.9674241.

[86] B. Park, S. P. Dang, S. Noh, J. Yi, and M. Park, "Dynamic virtual network honeypot," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2019, pp. 375–377. DOI: 10.1109/ICTC46691.2019.8939791.

[87] S. Kyung, W. Han, N. Tiwari, *et al.*, "Honeyproxy: Design and implementation of next-generation honeynet via sdn," in *2017 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2017, pp. 1–9. DOI: 10.1109/CNS.2017.8228653.

[88] X. Luo, Q. Yan, M. Wang, and W. Huang, "Using mtd and sdn-based honeypots to defend ddos attacks in iot," in *2019 Computing, Communications and IoT Applications (ComComAp)*, IEEE, 2019, pp. 392–395. DOI: 10.1109/ComComAp46287.2019.9018775.

[89] C. Gao, Y. Wang, X. Xiong, and W. Zhao, "Mtdcd: An mtd enhanced cyber deception defense system," in *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, IEEE, vol. 4, 2021, pp. 1412–1417. DOI: 10.1109/IMCEC51613.2021.9482133.

[90] B. Rashidi, C. Fung, K. W. Hamlen, and A. Kamisinski, "Honeyv: A virtualized honeynet system based on network softwarization," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–5. DOI: 10.1109/NOMS.2018.8406205.

[91] C. S. Bontaș, I.-M. Stan, and R. Rughiniș, "Honeypot generator using software defined networks and recursively defined topologies," in *2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet)*, IEEE, 2022, pp. 1–5. DOI: 10.1109/RoEduNet57163.2022.9921097.

[92] J. Franco, A. Aris, L. Babun, and A. S. Uluagac, "S-pot: A smart honeypot framework with dynamic rule configuration for sdn," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, IEEE, 2022, pp. 2818–2824. DOI: 10.1109/GLOBECOM48099.2022.10000682.

[93] R. Li, M. Zheng, D. Bai, and Z. Chen, "Sdn based intelligent honeynet network model design and verification," in *2021 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, IEEE, 2021, pp. 59–64. DOI: 10.1109/MLISE54096.2021.00019.

[94] H. Wang and B. Wu, "Sdn-based hybrid honeypot for attack capture," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, IEEE, 2019, pp. 1602–1606. DOI: 10.1109/ITNEC.2019.8729425.

[95] M. Karakate, H. Esaki, and H. Ochiai, "Sdnhive: A proof-of-concept sdn and honeypot system for defending against internal threats," in *2021 the 11th International Conference on Communication and Network Security*, 2021, pp. 9–20. DOI: 10.1145/3507509.3507511.

[96] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE transactions on dependable and secure computing*, vol. 10, no. 4, pp. 198–211, 2013. DOI: 10.1109/TDSC.2013.8.

[97] "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model," International Organization for Standardization (ISO), Geneva, Switzerland, Technical Report ISO/IEC TR 25010:2023, 2023.

[98] "Proactive vulnerability detection and mitigation leveraged by sdn." (2024), [Online]. Available: https://github.com/linuxer1337/sdn-vuln. (Accessed Dec. 14, 2024).

[99] "Thehive: A scalable, open source and free security incident response platform." (2016), [Online]. Available: https://github.com/TheHive-Project/TheHive. (Accessed Sep. 7, 2024).

[100] "Cortex: A powerful observable analysis and active response engine." (2017), [Online]. Available: https://github.com/TheHive-Project/Cortex. (Accessed Sep. 7, 2024).

[101] Catalyst. "Catalyst - speed up your reactions." (2021), [Online]. Available: https://catalyst.security-brewery.com/. (Accessed Sep. 7, 2024).

[102] FastAPI. "Fastapi framework, high performance, easy to learn, fast to code, ready for production." (), [Online]. Available: https://fastapi.tiangolo.com/. (Accessed Sep. 4, 2024).

[103] T. Muhammad and M. Munir, "Network automation," *European Journal of Technology*, vol. 7, no. 2, pp. 23–42, 2023. DOI: 10.47672/ejt.1547.

[104] Redis. "Redis - the real-time data platform." (), [Online]. Available: https://redis.io/. (Accessed Oct. 3, 2024).

[105] Nmap. "Nmap: Discover your network." (), [Online]. Available: https://nmap.org/. (Accessed Sep. 29, 2024).

[106] G. Lyon. "Nmap network scanning." (2009), [Online]. Available: https://nmap.org/book/. (Accessed May. 20, 2024).

[107] Nmap. "Nmap(1) - linux man page." (), [Online]. Available: https://linux.die.net/man/1/nmap. (Accessed May. 20, 2024).

[108] GVM. "Vulnerability management: Open source and gdpr-compliant." (), [Online]. Available: https://www.greenbone.net/en/. (Accessed Sep. 3, 2024).

[109] Ryu. "Build sdn agilely." (), [Online]. Available: https://ryu-sdn.org/. (Accessed Jul. 10, 2024).

[110] OpenvSwitch. "Open vswitch - production quality, multilayer open virtual switch." (), [Online]. Available: https://www.openvswitch.org/. (Accessed Jul. 16, 2024).

[111] Mininet. "Mininet - an instant virtual network on your laptop (or other pc)." (), [Online]. Available: https://mininet.org/. (Accessed Aug. 29, 2024).

[112] ISC. "Isc dhcp - enterprise-grade solution for ip address-configuration needs." (), [Online]. Available: https://www.isc.org/dhcp/. (Accessed Sep. 28, 2024).

[113] ISC. "Kea dhcp - modern, open source dhcpv4 and dhcpv6 server." (), [Online]. Available: https://www.isc.org/kea/. (Accessed Sep. 28, 2024).

[114] Iscte. "Iscte - laboratories." (), [Online]. Available: https://www.iscte-iul.pt/conteudos/research/1002/laboratories. (Accessed Oct. 8, 2024).

[115] C. F. Torres, R. Camino, *et al.*, "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1343–1359.

[116] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: How dark is the forest?" In *2022 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2022, pp. 198–214.

[117] S. Rouhani and R. Deters, "Security, performance, and applications of smart contracts: A systematic survey," *IEEE Access*, vol. 7, pp. 50 759–50 779, 2019.

[118] M. Eckhart and A. Ekelhart, "Digital twins for cyber-physical systems security: State of the art and outlook," *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*, pp. 383–412, 2019.

[119] B. Lewis, M. Broadbent, and N. Race, "P4id: P4 enhanced intrusion detection," *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–4, 2019. DOI: 10.1109/NFV-SDN47374.2019.9040044.

[120] H. Harkous, M. Jarschel, M. He, R. Pries, and W. Kellerer, "P8: P4 with predictable packet processing performance," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2846–2859, 2021. DOI: 10.1109/TNSM.2020.3030102.

[121] A. Siddiqui, B. P. Rimal, M. Reisslein, and Y. Wang, "Survey on unified threat management (utm) systems for home networks," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024. DOI: 10.1109/COMST.2024.3382470.

[122] J. A. Ordonez-Lucena, P. Ameigeiras, D. López, J. Ramos-Muñoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, pp. 80–87, 2017. DOI: 10.1109/MCOM.2017.1600935.

[123] X. Li, M. Samaka, H. A. Chan, *et al.*, "Network slicing for 5g: Challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017. DOI: 10.1109/MIC.2017.3481355.

[124] H. Zhang, N. Liu, X. Chu, K. Long, H. Aghvami, and V. C. M. Leung, "Network slicing based 5g and future mobile networks: Mobility, resource management, and challenges," *IEEE Communications Magazine*, vol. 55, pp. 138–145, 2017. DOI: 10.1109/MCOM.2017.1600940.

[125] J.-f. Xie, F. Yu, T. Huang, *et al.*, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 393–430, 2019. DOI: 10.1109/COMST.2018.2866942.

[126] A. Silva, P. Smith, A. Mauthe, and A. E. S. Filho, "Resilience support in software-defined networking: A survey," *Comput. Networks*, vol. 92, pp. 189–207, 2015. DOI: 10.1016/j.comnet.2015.09.012.

APPENDIX A

Open Issues

In continuation of exploring proactive approaches to enhance cybersecurity, the discussion extends to emerging research areas such as programmable smart contracts integrated with distributed ledger databases. While this innovation promises decentralized and more scalable transaction systems, it faces challenges such as frontrunning attacks due to the reliance on public mining phases for ledger updates. These attacks allow to extract some amount of the victim initial transaction's expected outcome towards directly the attacker(s) profit [115]. Other attacks are discussed in [116]. Recent efforts propose incentivizing good behavior among network participants to preemptively mitigate those attacks. However, to fully realize the potential of these solutions, further investigation is needed. Specifically, there's a need to delve into enhancing smart programmable systems with capabilities for proactive discovery and mitigation of security vulnerabilities at the network edge. This involves leveraging secure smart contracts [117] and trustful mining [117] mechanisms to facilitate the seamless sharing of system assets among stakeholders at the network periphery.

The Digital Twin (DT) can be very useful to elect a set of system management policies to detect and mitigate system vulnerabilities inside the network infrastructure of any organization. The main concept behind DT is representing the real system in a virtual model where future security actions are simulated, tested, selected, and transposed back to the real system. Thus, DT can be fundamentally effective in scenarios associated to high-secure cyber-physical systems [118], using IoT devices, smart agents, and the network edge.

The underneath discussion outlines a roadmap for advancing smart programmable systems towards proactive security measures in next-generation networked systems.

A.1. Data Plane

Leveraging Data Plane programmable technologies, such as the case of P4 [19], [20], can advance the state-of-the-art for the proactive discovery and mitigation of security vulnerabilities in next-generation networks with IoT devices. In fact, the P4 adoption allows the major processing associated to both vulnerability detection and reaction could be done at the data plane level, reducing the necessary system time to react against threats and diminishing the SDN channel control workload. We envision the next P4 data plane open issues:

P4-Based Intrusion Detection Systems (IDS): Explore the development of P4-based IDS [119] tailored for next-generation networks with IoT devices. These systems should be capable of analyzing network traffic in real-time, identifying potential security threats, and triggering proactive mitigation actions.

Dynamic Security Policy Enforcement: Investigate methods to dynamically adapt security policies in P4-enabled network devices based on detected vulnerabilities and network conditions. This involves designing flexible P4 programs that can be updated on-the-fly to address emerging security risks.

Vulnerability-aware Routing and Traffic Engineering: Develop P4-based mechanisms for integrating vulnerability information into routing and traffic engineering decisions. This could involve leveraging vulnerability data to optimize traffic flows, minimize attack surface, and enhance network resilience against security threats.

Fine-grained Access Control and Segmentation: Explore the use of P4 to implement fine-grained access control and segmentation strategies in next-generation networks. This includes defining P4-based policies to isolate IoT devices, enforce least privilege principles, and prevent lateral movement of attackers within the network.

Behavioral Anomaly Detection: Investigate the use of P4 for implementing behavioral anomaly detection techniques tailored for IoT environments. Develop P4 programs capable of profiling normal device behavior, detecting deviations indicative of security breaches, and triggering proactive responses.

Integration with Machine Learning: Explore synergies between P4-based network programmability and machine learning techniques for security. Investigate approaches to integrate ML models into P4 pipelines for enhancing the accuracy of security threat detection and mitigation.

Resilience and Fail-Safe Mechanisms: Design P4-based resilience mechanisms to ensure the reliability and fail-safe operation of security features in next-generation networks. This involves implementing redundancy, failover, and recovery mechanisms within P4 programs to withstand attacks and hardware failures.

Scalability and Performance Optimization: Investigate techniques to optimize the scalability and performance of P4-based security solutions in large-scale IoT deployments. This includes designing efficient and predictable packet processing pipelines [120].

Using DPI in P4-enabled SDN systems: Enable innovative solutions for dealing with scenarios involving hosts security vulnerabilities and cyber threats trying to explore those. One possible application of DPI's capabilities is that after detecting that a machine has low-risk vulnerabilities, instead of quarantining the machine, by moving it to a VLAN with restricted access, it can be protected by inspecting and eventually discarding malicious traffic destined to that machine. In this way, the machine remains attached in the normal way to the network and it continues available to perform its tasks, until the minor machine vulnerabilities could be solved.

A.2. Control

Considering now using the upper layers of programmable systems with the intent of advancing the state-of-the-art for the proactive discovery and mitigation of security vulnerabilities in next-generation networks with IoT devices, we envision the next open issues:

Dynamic Security Policy Enforcement: Develop mechanisms within SDN controllers to dynamically enforce security policies based on real-time threat intelligence and network conditions. This involves creating flexible policy management frameworks that can adapt to evolving security threats.

Vulnerability-aware Network Management: Investigate methods to integrate vulnerability assessment data into SDN controllers for proactive network management. Develop algorithms to prioritize and mitigate vulnerabilities in IoT devices based on their criticality and impact on network security.

Adaptive Access Control: Explore the use of SDN controllers to implement adaptive access control mechanisms for IoT devices. Develop policies that dynamically adjust device access privileges based on contextual information, such as device behavior and security goals.

Behavioral Anomaly Detection: Research techniques for implementing behavioral anomaly detection within SDN controllers. Develop algorithms to analyze network traffic patterns, detect abnormal behavior indicative of security threats, and trigger appropriate mitigation actions.

Threat Intelligence Integration: Investigate how to enhance the SDN controllers' intelligence for dealing with system menaces. A possible way is developing novel mechanisms to automatically update security policies based on emerging threats and known vulnerabilities in IoT devices. Easily to configure and Unified threat management systems are seen as very important to keep protected the operation of networked systems [121].

Software-defined Segmentation: Explore the use of SDN controllers to implement software-defined segmentation in next-generation networks, using segmentation technologies such as VLAN, segment routing or network slicing [122]–[124]. Develop policies to isolate IoT devices into secure segments based on their security requirements and communication patterns.

Machine Learning Integration: Investigate synergies between SDN controllers and machine learning techniques for security enhancement [125]. Develop algorithms to analyze network data, detect anomalies, and predict potential security threats, leveraging the programmability of SDN controllers.

Resilience Mechanisms: Design resilience mechanisms within SDN controllers to ensure the reliability of system security features. Develop robust recovery mechanisms to system threats such as cyber attacks and system failures [126].

Scalability and Performance Optimization: Optimize the scalability and performance of SDN controllers in large-scale IoT deployments. Develop algorithms and data structures to efficiently handle the increased volume of security-related data and policy updates.

Validation and Testing Frameworks: Develop comprehensive validation and testing frameworks for SDN-based security solutions, including the P4 usage at the data plane. This involves creating realistic testbeds, generating attack scenarios, and evaluating the effectiveness of proactive security measures. Then, the more efficient solutions can be applied in the production network.

As a final conclusion of this section, to tackle the long list of unresolved problems discussed above, we think a strategic cooperation is also necessary among scholars, industry participants, and policymakers to foster innovation, develop best practices, and establish standards for proactively addressing key security vulnerabilities in edge computing scenarios before these system weaknesses be explored by cyber attackers.