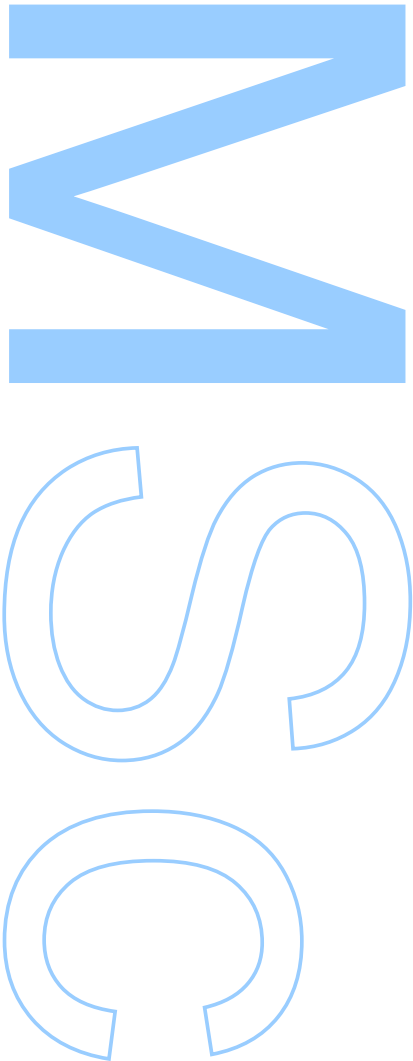
Isolated environments for threat detection and mitigation

Simão Francisco Oliveira da Silva
Mestrado em Segurança Informática
Departamento de Ciências de Computadores
2021

Orientador
Luís Filipe Coelho Antunes
Professor Catedrático
Faculdade de Ciências da Universidade do Porto

Coorientador
Patrícia Raquel Vieira Sousa
Doutora
Faculdade de Ciências da Universidade do Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

UNIVERSIDADE DO PORTO

MASTERS THESIS

Isolated environments for threat detection and mitigation

Author:
Simão SILVA

Supervisor:
Luís ANTUNES

Co-supervisor:
Patrícia SOUSA

A thesis submitted in fulfilment of the requirements
for the degree of MSc. Information Security

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

September 30, 2021

Acknowledgements

I would like to express my gratitude and appreciation to Professor Luís Antunes and Professor Patrícia Sousa for their support, guidance and the opportunity to develop this thesis.

I also would like to give a special thanks to Professor João Resende for his patience and dedication. Alongside Patrícia Sousa, they were tireless in supporting this work, sharing their suggestions and advice and for providing encouragement during the most difficult times, essential to complete this thesis. To them, my sincere and deepest gratitude.

To my colleagues at C3P, Inês and André, I am grateful for their support and warm welcome that made possible our great working environment. Thank you for being there when I need it.

To my family, there are no words that can express the thanks for the incredible support during this journey, especially during the bad times. Without you, I could never have gotten where I am today.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

MSc. Information Security

Isolated environments for threat detection and mitigation

by Simão SILVA

A honeypot is configured to detect, bypass or otherwise neutralise attempts at the unauthorised use of information systems. Honeypots are often used in a controlled environment with a specific network not listed on the web to identify potential threats and are designed to work away from the critical infrastructure environment. The increasing adoption and use of Internet of Things (IoT) devices make the task of configuring honeypots complex due to the highly dynamic environments. Some authors have proposed the notion of honeypots remotely using the same infrastructure, with the network requests from IoT devices being redirected to a remote cloud server, although it does not require a separate infrastructure to detect attacks and connections. However, with external servers, response delays make these systems discoverable and indicate to the attacker that they may be in the presence of a honeypot.

In this work, we focus on deploying honeypots virtually on IoT devices. With this technology, we can use endpoints to dispatch specific honeypots on recent known vulnerabilities in IoT devices to find and notify attacks within the network, as much of this information is verified and freely available by government entities. Unlike other approaches, the idea is not to have a fixed honeypot, but a set of devices that can be used at any time as a honeypot (adapted to the latest threat) to analyse the network for a potential problem and then report to the Threat Sharing Platform (TSP).

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Mestrado em Segurança Informática

Ambientes isolados para deteção e mitigação de ameaças

por Simão SILVA

Um *honeypot* é configurado para detetar, prevenir ou eliminar tentativas de acesso não autorizadas a sistemas de informação. *Honeypots* são geralmente usados num ambiente controlado e com uma rede específica segregada, para que seja possível identificar possíveis ameaças. Desta forma, este ambiente é criado de forma separada do ambiente crucial da infraestrutura. O aumento da adopção e uso de dispositivos da Internet das Coisas (IdC) tornam a tarefa de configuração de *honeypots* complexa devido aos ambientes altamente dinâmicos. Na literatura, temos vários exemplos de trabalhos que propõem a noção de *honeypots* remotos usando a mesma infraestrutura, com os pedidos dos dispositivos de IdC a serem reencaminhados para um servidor remoto na *cloud*, embora não seja requerida uma infraestrutura à parte para detetar ataques e conexões. No entanto, com servidores externos, os atrasos nas respostas tornam estes sistemas detectáveis e indicam ao atacante que pode estar na presença de um *honeypot*.

Neste trabalho, focámo-nos na implementação de *honeypots* virtuais em dispositivos que pertencem à tecnologia da IdC. Com este sistema, podemos usar dispositivos (nós) periféricos para lançar *honeypots* específicos de vulnerabilidades recentes e conhecidas em dispositivos de IdC para procurar e notificar ataques dentro da rede, usando informações de vulnerabilidades amplamente escrutinadas e disponibilizadas gratuitamente por entidades governamentais. Ao contrário de outras abordagens, a intenção não é disponibilizar um *honeypot* fixo num determinado dispositivo, mas antes ter um conjunto de dispositivos que possam ser usados em qualquer momento como um *honeypot* (adaptado à ameaça mais recente) com o intuito de analisar a rede de potenciais problemas e reportá-los à Plataforma de Partilha de Inteligência.

Contents

Acknowledgements i

Abstract ii

Resumo iii

Contents iv

List of Figures vii

List of Tables viii

1 Introduction 1
- 1.1 Motivation . . . 3
- 1.2 Proposed solution . . . 5
- 1.3 Contributions . . . 5
- 1.4 Outline . . . 6

2 Background 7
- 2.1 Honeypots . . . 7
- 2.2 Threat intelligence . . . 8
- 2.3 Cyberattacks in IoT . . . 9
- 2.4 Research Study on CVE and IDS . . . 10
  - 2.4.1 CVE . . . 10
  - 2.4.2 IDS . . . 11

3 Related work 13
- 3.1 Host-based Intrusion Detection Systems . . . 13
  - 3.1.1 AIDE . . . 13
  - 3.1.2 Fail2ban . . . 14
  - 3.1.3 OSSEC . . . 14
  - 3.1.4 Sagan . . . 14
  - 3.1.5 Samhain . . . 14
  - 3.1.6 Tripwire . . . 15
- 3.2 Threat Sharing Platforms . . . 15
  - 3.2.1 IBM X-Force Exchange . . . 16
  - 3.2.2 Anomali ThreatStream . . . 16

3.2.3 ThreatConnect . . . 16
3.2.4 CrowdStrike Falcon X . . . 16
3.2.5 ThreatQ . . . 17
3.2.6 Malware Information Sharing Platform . . . 17
3.3 IoT Honeypots . . . 18

4 Study on Host-based Intrusion Detection Systems 21
4.1 Experimental setup and methodology . . . 21
4.2 Results . . . 22

5 System design 26
5.1 Architecture . . . 26
5.2 Components . . . 28
5.2.1 Threat Sharing Platform . . . 28
5.2.1.1 Indicators of Compromise . . . 28
5.2.1.2 Common Vulnerabilities and Exposures . . . 28
5.2.2 Host Intrusion Detection System . . . 28
5.2.3 STDM Central Coordinator . . . 29
5.2.4 Watchdog . . . 29
5.2.5 Honeypot . . . 29

6 Implementation 31
6.1 Threat Sharing Platform . . . 31
6.2 STDM . . . 32
6.3 Host Intrusion Detection System . . . 33
6.4 Isolated environments . . . 34
6.4.1 Virtualisation . . . 34
6.4.2 Honeypot . . . 35
6.5 Watchdog . . . 36

7 Evaluation 37
7.1 Environment description . . . 37
7.2 Methodology . . . 38
7.3 Results . . . 39

8 Security Analysis 41
8.1 Attacker compromises the Sandbox system . . . 41
8.2 Unknown Zero-Day detected . . . 41
8.3 Physical access to the machine . . . 42
8.4 Denial-of-Service scenario . . . 42

9 Conclusion 43
9.1 Limitations . . . 43
9.2 Future Work . . . 44
9.3 Conclusions . . . 45

Acronyms 46

Bibliography **49**

List of Figures

5.1 System architecture 27

6.1 Example of a MISP event 32
6.2 Overview of the STDM workflow 33

7.1 Latency (ms) per number of requests 39

List of Tables

3.1 Summary of the HIDS features 15
3.2 State of the art's features summary 20

4.1 Experimental trials on HIDS 23

7.1 Latency results for the three setups 39

Chapter 1

Introduction

Internet usage is growing day by day, and our dependence on it is increasing. In an increasingly digital world, the way we shop or interact with objects is somewhat obsolete and is changing the environment around us. The speed of change is very high and will not slow down. New technologies - from smart homes to smart cities - are now indispensable for our daily lives, even if we are not aware of their presence. The smartphone introduction is a perfect example. Older generations, that are generally more reluctant to change, are using smartphones to connect with their family and friends, get the latest news or even remote medical appointments, moving away from phone lines and newspapers, and avoiding unnecessary travel.

The ubiquity of the Internet enabled the emergence of IoT, one of the most versatile technologies in existence today. Because it is scalable and adaptable, IoT has revolutionised industries and introduced new concepts such as Industry 4.0 [1, 2] and the Industrial Internet of Things [3], bringing new horizons to efficiencies in all types of manufacturing through data acquisition and analysis. The last few years have shown an increase in the development and use of devices and solutions based on IoT. Its proliferation has brought the IoT paradigm to a prominent role in our everyday lives, with devices gaining new features and changing the way we connect and interact. An example are intelligent audio systems such as Google Home and Amazon Echo. These devices are becoming our personal assistants, allowing us to listen to music, control video or photo playback, set alarms or receive news or traffic updates via voice commands. Its popularity has also increased its support for other features, including control of other network-connected devices - also known as home automation - which can include a variety of IoT sensors.

New technologies and their progression are changing the environment around us and, consequently, the society. With users spending more time staring at a screen than sleeping [4], it is important to alert users to the adversities they may face when using them. As the software is not unbreakable and new daily vulnerabilities are identified and made public, their devices and data can be compromised. In keeping with the digital age, where everything and everyone is connected, these vulnerabilities are additions to the intruder arsenal that, following previous examples, can be used to their advantage and possibly have serious consequences.

The growing number of cyber attacks and their extensive damage has changed the mindset of organisations, with more resources being applied to cyber security. According to statistics from AV-TEST, an institute in Germany, as of January 2021 [5] there were over a billion malicious executable scripts known to the security community. Digital transformation has also meant an increase in cybercrime, often associated with significant financial losses for individuals and organisations.

The past few years have seen a paradigm shift for attackers from pranks or entertainment to financial or military/political gain, explained by the increasing use of phishing and ransomware attacks. The latter has been a major concern and the one that attracted the most attention during the Covid-19 pandemic. The idea of the attack usually consists of threats to publish or block access to data or computer system, usually by encrypting it, until the victim pays the attacker a fee [6] and can have life-threatening consequences. Although there is no official death directly caused by theses attacks, the same cannot be said regarding normal functioning of services with many news reporting disturbance in the normal day-to-day operations. An example occurred during one of the spikes of Covid-19 in the United States [7] to Universal Health Services, a healthcare provider. The attack was able to lock computers and phones, and a message was displayed on computer screens reporting known ransomware, prompting Information Technology (IT) staff to instruct employees to turn off all computers and phones and switch to an offline process to avoid more damage. Still, it is unclear whether the patient data made it into the aggressors' hands. Another concern during the pandemic was state-sponsored attacks to steal knowledge, namely vaccine research. In the 2021 Global Threat Report recently released by CrowdStrike [8], the findings show that state-sponsored adversaries have engaged to steal valuable data related to vaccine research and government responses to Covid-19.

In a more detailed analysis, it is exposed the *big game hunting* phenomenon, where attackers targeted high-value persons of interest for extortion and blackmail. Along with techniques such as phishing, the reports also show that attackers were able to infect companies' infrastructure and profit from it as, due to the urgent nature of some products, paying the fee is often cheaper than paying the critical fee for loss of income.

As new daily vulnerabilities are identified and made public, attackers are constantly finding new ways to compromise devices and systems that, as mentioned, are additions to the attacker's arsenal, with previous examples showing that can generate serious consequences if we do not prepare for it. However, the increase in the number of cyber attacks and their extensive damage has changed the mindset of organisations, with more resources being applied to cyber security and the creation of specific teams and usage of dedicated tools to deal with this problem. The acronyms of the new teams became part of the vocabulary with defined objectives and methods, which, however, may overlap in some parts. An important one with a broader scope is Security Operations Centre (SOC), responsible for the prevention, incident response, compliance and risk management and is composed of analysts who need to know the constituent (e.g. business objectives, asset prioritisation, IT infrastructure and inter-dependencies), how it can be attacked and how it can be defended.

One of the most prominent methods used in security is a Intrusion Detection System (IDS) which are built to monitor systems and detect anomalies. This definition is often supported by analysis based on patterns or knowledge, leading to more sophisticated machine learning algorithms in search of better performance.

1.1 Motivation

Cybersecurity remains a major concern as its risks are a persistent threat to users and organisations. According to a survey published in December 2020 by Trend Micro [9], a cybersecurity company, nearly a quarter of respondents replied that they had suffered at least seven attacks and 83% responded that they expect an attack to be successful in the next year are "somewhat" to "very" likely.

Investment in cybersecurity is still questioned even in the current scenario, underestimate by decision-makers. Indifferent to this recurring discussion, cyberattacks are becoming smarter and more capable by the day, leaving security teams behind, left to analyse artefacts from the past to try to determine the future [10]. To bridge this gap and

as today's threats are known to be *evasive, resilient, and complex* [11], a new approach to security defences is needed with threat intelligence a key factor. Today's challenges call for collecting and sharing threat information to prevent attacks or at least perform disaster recovery promptly.

An approach to understanding and gathering intelligence about how our infrastructure might be compromised is changing the way we can defend ourselves. Given that an attacker only needs to find a single vulnerability and exploit it to gain unauthorised access, we need to learn their tactics and techniques. One way to accomplish that is using decoys, such as honeypot systems. These systems are a monitored network decoy that mimics the main system to lure adversaries into thinking they are exploiting a legitimate target and distract them from more valuable machines on a network, discovering new attacks and exploiting trends.

The use of honeypots has been an important tool to face the increase in cyber attacks. These systems are deployed with deliberately built-in security vulnerabilities and weak security measures such as weak login passwords or known vulnerable versions of installed services. Also, open ports can be left open to lure attackers to the honeypot system instead of a real system. These characteristics make honeypots the perfect bait and very attractive to attackers.

Honeypots are versatile and are not addressed exclusively to a specific problem like other common solutions (firewalls and antivirus). Given its functioning, it becomes an important information tool to spot existing threats and possibly new ones, which makes them an important asset especially relevant for SOC teams since they possess a major database of attacks' artefacts and, consequently, a major source of threat intelligence. By working together with threat intelligence platforms, honeypots systems are becoming a mandatory component for cybersecurity defence.

Although there is a large variety of honeypots aimed for different purposes, these systems have some constraints. Given the diversity of devices, there are still limitations in using these systems outside of x86 architectures, leaving out capable devices such as Raspberry Pi, which align computational power with low energy consumption. Another issue is regarding its use. With the proliferation of the cloud, we are witnessing the phenomenon of services migration to external servers, which, in the case of honeypots, the trend is to use remote machines as baits, in which traffic is redirected from the local network to the cloud network. However, resorting to this method allows the attacker to

easily detect that it is in a honeypot by measuring the latency times of Internet Control Message Protocol (ICMP) ECHO requests, thus making the honeypot ineffective.

1.2 Proposed solution

In this thesis, we explore the concept of honeypot lifecycle control and its application in heterogeneous environments. Given the growth in the number of devices connected and their computational power, honeypot systems can be mobile, not being restricted to the same device. This flexibility allows these systems to be resilient to reconnaissance techniques as the same instance can run on different devices with different specifications, thus rendering the reconnaissance information unusable that the attacker may have previously retrieved from their initial interaction. Thus, it is possible to confuse the attacker when trying to exploit a system.

Most solutions address the external point of view of the network. In this thesis, we are interested in protecting the internal infrastructure as well. As an example, IoT sensors are often placed in separate Virtual Local Area Networks due to the risk they lead to the remaining network (servers or critical machines). The main goal is to have a system that runs on top of an IoT infrastructure, but uses virtual honeypots inside of the devices to extract information regarding possible vulnerabilities in local services or possible intruders in the network.

1.3 Contributions

With our solution we pretend to reach the following goals:

- **Goal 1** - Provide an autonomous threat detection flow [1] able to analyse the diversity of devices in the network and detect and mitigate, if possible, its vulnerabilities;
- **Goal 2** - Creation of automatic honeypot instance deployment when a vulnerability is present in a device in the network, capable of running in a variety of devices including in low-power, performance-constraint devices;

[1] The threat detection flow is the set of tasks that will automatically gather the information of vulnerabilities and put it in the threat sharing platform that, in turn, will be used by the central coordinator to verify if a device in the network has a software affected by said vulnerability.

- **Goal 3** - Provide a mechanism that can discretely monitor honeypots instances to collect intelligence of tactics and techniques of intruders.

1.4 Outline

This thesis is structured in nine chapters describing our solution. The present chapter discusses the IoT concept and its devices, the emerging threats they are facing and how we intent to tackle this issue. Chapter 2 introduces the relevant concepts and technologies used in our solution that are explored in chapter 3, where we reunite some of the researches and examples of the current available solutions. Chapter 4 describes a study to evaluate the effectiveness of Host-based Intrusion Detection System (HIDS) when known flaws are being exploited. Chapter 5 presets an overview of our system with chapter 6 describing the technical aspects of implementation, concluding with an evaluation of its feasibility in chapter 7. Chapter 8 presets a security analysis of our proposal. Lastly, in chapter 9, we present the limitations our system and some ideas for future work as well as some final remarks.

Chapter 2

Background

The diversity of services and devices that accompany us and that we need requires us to consider the choice of tools and technologies to adopt for our security perimeter, as we will trust them with our devices and our information, not forgetting other aspects such as performance, reliability and maturity.

Our proposal aims to unite several technologies to create a robust system, providing benefits to users. With that in mind, this section revises the current state of technologies and tools such as honeypots, threat intelligence sharing platforms and IDS and respective concepts contextualisation, an overview of the cyberattacks affecting IoT devices, and, finally, a research study on Common Vulnerabilities and Exposures (CVE) and intrusion detection systems.

This background information will be important to understand our decisions in the design process.

2.1 Honeypots

Honeypots can be categorised based on their uses and purposes, depending on the different services they provide and the level of interaction they allow.

A low interaction honeypot will give the attacker limited access with some emulated services and protocols, that is, the attackers' interaction with the system is limited, and for a short period, just enough for deceiving attackers with little knowledge [12]. With this type of honeypot, the information about the attack and the attacker is small and serves the purpose of protecting the infrastructure, which makes them widely used in infrastructures where the interested party is protecting their system from the outside world.

In the opposite direction, high-interaction honeypots provide the most realistic infrastructure, making it much less likely that the attacker will discover that is being shunted or observed. They can gain full control of the system or even tamper with it. The goal is to get as much information about the attackers and learn their tactics and techniques [12].

2.2 Threat intelligence

As more and more attacks occur, it increases the likelihood that some organisation or group has seen such an attack before. Therefore, knowledge exists in some form, somewhere. However, it needs to be collected, validated and turned into actionable information. The goal of threat intelligence is to provide the ability to recognise and act on Indicators of Compromise (IoC) promptly, i.e., identify pieces of forensic data that signs that a system has been compromised by an attack or that it has been infected with a particular malicious software and apply prevent and mitigation measures [13]. Some examples of IoC include unusual Domain Name System (DNS) lookups, suspicious files and processes, an unusual large number of accesses to one file, data transfer over rarely used ports, file hash of a known piece of malware, unauthorised modification of configuration files or device settings and a large number of unsuccessful login attempts [14].

As cyber criminals become more sophisticated, IoC have become more difficult to detect. Common IoC - such as filenames, file hashes, IP addresses and registry keys - are constantly changing, thus increasing the difficulty of an already very complex and difficult task [15].

The underlying process of identifying an IoC is a crucial task. When an organisation is being targeted, the attacker will leave traces of their activity in the system and log files. The threat hunting team will gather this digital forensic data from these files and systems to determine if a security threat or data breach has occurred or if it is still ongoing. Given the amount of data, the team will make use of advanced solutions, such as artificial intelligence, machine learning and other forms of intelligent automation to detect anomalous activities and improve response time [16]. The need to exchange intelligence data between users and communities is critical to help protect against evolving threats. It helps to adapt more quickly and to encourage collaboration and sharing of best practices. The introduction of the concept of threat intelligence sharing platform in 2013 by Dandurand and Serrano [17] led to a variety of solutions available today. Threat Intelligence has become the security buzzword in the information security community and led to increasing

interest in the current environment of the Threat Intelligence industry, current research, and potential future use cases. In 2015, a study conducted by the SANS Institute [18] analysed a small selection of open source threat intelligence platforms and concluded that the market was still developing. A more recent comprehensive study, conducted in 2020 [19], revealed that there is still room to mature. After analysing 22 platforms, some of the authors key findings were the need for a clear definition of threat intelligence sharing and that current threat intelligence sharing is comparable to data warehousing and does not provide *real* intelligence.

2.3 Cyberattacks in IoT

The diversity of IoT devices has grown, being present at our home, at work, or on the street. Given its popularity, the manufacturers have increased their catalogue and provide countless appliances for customers. Given today's reality, this represents some setbacks given that security is not always a concern and may turn users' devices and networks vulnerable to attacks, with many users unaware of the implications of having a device exposed to the Internet. Then, attackers take advantage by exposing those devices' weaknesses and exploiting or triggered them, thus becoming the device and, consequently, the user vulnerable.

Cyberattacks intend to compromise a computer system, a network, a device or an infrastructure by intercept, steal, alter or destroy data or information systems, thus jeopardising the confidentially, integrity and availability principles. Exploiting has become increasingly easier and accessible with tools available for all users, whether beginners or experienced. Also, the advances in technology allowed the appearance of multiple types of attacks. Depending on the attacker's intentions, it is possible to distinguish two types of attacks: active and passive.

Active attacks represent an action that intentionally disrupts the system by altering itself or affect its operation. It involves masquerading (impersonation of user or system), message replay, message modification and system overload. Common examples of attacks are brute force attacks, Cross-site scripting (XSS), Denial-of-Service (DoS), phishing, Man-in-the-Middle (MitM), Remote Code Execution (RCE), ransomware and Structured Query Language (SQL) injection [20].

On the other hand, attackers do not modify messages in passive attacks. This way, there are no changes to the systems or network data. Passive attacks are associated with

the reconnaissance phase, where the attacker is only monitoring and scanning the system for vulnerabilities. In other words, the attacker eavesdrops (listening communications without consent) or performs other non-invasive actions such as port scanning, wiretapping or idle scanning (send spoofed packets to find out what services are available). In summary, active attacks compromise the integrity and availability of a system, while passive attacks endanger the confidentiality principle.

2.4 Research Study on CVE and IDS

Given the knowledge of vulnerabilities provided by the CVE catalogue, it is important to evaluate how current defence solutions, namely IDS, react when those vulnerabilities are being exploited.

2.4.1 CVE

Among the variety of sources of vulnerability information are the CVE program provided by MITRE, an American not-for-profit organisation. It consists of a list of records of publicly known cybersecurity vulnerabilities and contains their identifiers which are later used by databases that provide enhanced information for each record. The National Vulnerability Database (NVD) from National Institute of Standards and Technology (NIST), a non-regulatory government agency that develops technology, metrics and standards, is one of the most known and popular databases given that is highly maintained and is free to use by everyone. As it is important to know the software vulnerable, each CVE record in the database includes a list of vulnerable software and corresponding versions, in which the description of a product follows the specifications of the Common Platform Enumeration (CPE) standard [21]. As an example of its importance and coverage, the well-known security flaw *Heartbleed* that affected OpenSSL has the identifier CVE-2014-0160 [22] and the SMB vulnerability that allowed the WannaCry ransomware spread [23] has the identifier CVE-2017-0144 [24].

The CPE standard is a structured naming scheme for information technology systems, software and packages. It is based on the generic syntax for Uniform Resource Identifier (URI) and its specification includes the naming syntax, conventions for constructing CPE names from product information, a matching algorithm and an Extensible Markup Language (XML) schema for binding descriptive and tests information to a name [25]. It

was a response to the need for specific languages, such as CVE, OVAL [26] and XCCDF [27], to refer to IT products and platforms in a standardised way suitable for machine interpretation and processing [28].

2.4.2 IDS

Given today's demand for robust security solutions, one of the most prominent methods is IDS. These systems are built to monitor a network for malicious activity or policy violations and are responsible for preventing breaches of security incidents, monitoring and reacting to any unauthorised access that causes damage to the information stored, the information system, or the network [29].

Depending on how the examination is done, IDS can be classified in two categories:

- Network-based Intrusion Detection System (NIDS): a system that analyses incoming network traffic. It checks for attacks or irregular behaviour by inspecting the contents and header information of all the packets moving across the network;
- Host-based Intrusion Detection System (HIDS): a system that monitors important operating system files and processes.

Also, IDS evaluates network traffic in real-time against a signature policy, definition of acceptable/*normal* behaviour, or some other set of heuristics. In general, these are divided into two methods: signature-based and anomaly-based.

- Signature-based: detects possible threats by looking for specific signatures such as bit patterns, keywords, known malicious instruction sequences, or known threats. When a new threat is identified, a signature is generated and added to the list used by the IDS for future scans. This allows achieving a high threat detection rate with no false positives because the alerts are generated based upon the detection of known malicious content. However, this method is blind when face with a zero-day vulnerability.

  The main advantage of this approach is the ease to develop defences against these attacks if we know the behaviour of the attack. As an example, if an attack goal is to exploit a particular buffer overflow, the IDS can use pattern matching to look for particular strings. On the other hand, this approach has the constant necessity of keeping updated the signatures' database and could generate a high rate of false

negatives, given that a slight change in how the attack is carried out can lead to a failure in its identification;

- Anomaly-based: also known as behaviour-based, these IDS resort to artificial intelligence, namely machine learning, to build a model that defines the *normal* behaviour. All future interactions are, then, compared against that model in search for potential anomalies, each one treated as a potential threat.

  Although this method allows detecting previously unknown attacks, it can suffer from a high rate of false positives, i.e., previously unknown but legitimate activity can accidentally be classified as a malicious one. Also, it makes difficult the definition of custom rules since defining a rule requires predicting all behaviours. Even if possible, attacks like path traversal [30] can still go unnoticed given that most of them are going to fall on the normal usage pattern.

Chapter 3

Related work

As stated, our system comprises the use of tools and technologies with different levels of maturity in the cybersecurity world. In the following sections, we review some of the state-of-the-art host-based intrusion detection systems, threat sharing platforms, and honeypots systems for IoT devices with an overview of their features.

3.1 Host-based Intrusion Detection Systems

Host-based Intrusion Detection System (HIDS) [31] is a type of IDS that focus on analysing specific activities on the hosts by monitoring files modifications or memory usage, among others. Its monitoring capability relies heavily on audit trails and system logs by which it is determined if the system has been compromised. This approach runs on individual hosts monitoring from the device and detecting improper use of the available resources.

We present some of the most popular systems available on the market: AIDE, Fail2ban, OSSEC, Sagan, Samhain and Tripwire [32–34].

3.1.1 AIDE

The Advanced Intrusion Detection Environment (AIDE) [35] is a file and directory integrity checker. It works by creating a database from the current system state, and once initialised, it is used as a baseline for file integrity checking. Some of the file properties that can be checked against include inode, permissions, modification time, and file contents.

It has several digest algorithms and allows to define policies of which attributes of which files should be checked. It only does file integrity checks and does not check for rootkits or parse log files for suspicious activity like similar systems [36].

3.1.2 Fail2ban

Fail2ban [37] is a framework developed in Python design to scan log files to detect and ban IP addresses conducting too many failed login attempts by dynamically adding a firewall rule. It acts as an agent regularly monitoring services' logs in search of attempts of intrusion.

Although it usually associated as a type of Intrusion Prevention System (IPS), we also included in this section given that some literature also considers it a HIDS [38, 39].

3.1.3 OSSEC

Open Source Security (OSSEC) [40] is a scalable, multi-platform open-source HIDS that performs log analysis, integrity checking, rootkit detection, time-based alerting and active response. It has four operation modes - server, agent (or client), local and hybrid - that allows to be configured to send alarm messages upon a rule's instruction, to apply a security measure or to automatically answer to the threat or warning as convenient.

3.1.4 Sagan

Sagan [41] is a real-time log analysis and correlation engine written in C that uses multi-threaded architecture to deliver high-performance log and event analysis. It was designed and meant to analyse logs across many different platforms in many different locations. It also provides structures and rules similar to Snort-based tools enabling compatibility and log correlation events with multiple systems.

3.1.5 Samhain

Samhain [42] is a multi-platform, open-source host-based HIDS developed by Samhain Labs. It provides file integrity checking, log file monitoring and analysis, port monitoring, and detection of rogue SUID executables - a binary with set user ID permission that, when poorly written, can quickly and easily escalate a user's privileges) and hidden processes. It can be deployed in a centralised way or distributed way, where each node acts as an individual implementation.

3.1.6 Tripwire

Tripwire [43] is a security tool for monitoring and alerting file changes on the system. It continuously tracks critical system files and reports unauthorised changes to files and directories, including permissions, internal file changes, and timestamp details. All scanned files information is stored in a database, which is later used to compare the machine's current state and the stored values and, if there were differences, alert the user.

HIDS	Features
AIDE	File signature comparisons File and directory integrity File attributes database
Fail2ban	Log file monitoring Active response Prevention system
OSSEC	Log file processing Monitoring firewall and traffic log Policies for alerts
Sagan	Lightweight Log file monitoring Shares Snort syntax (NIDS)
Samhain	File integrity checking Port monitoring Stealth monitoring
Tripwire	Unauthorised file changes Cryptographic hashes Permissions, internal file changes

TABLE 3.1: Summary of the HIDS features

3.2 Threat Sharing Platforms

In a common strategy to protect against sophisticated cyberattacks faced daily, organisations have increased their willingness to exchange information and knowledge about incidents, vulnerabilities, and threats so that they can collectively be more robust in the future. The need to facilitate the trade of information drives software providers to platforms developed under the term *Threat Intelligence Sharing Platforms* introduced in 2013 by Dandurand and Serrano [17].

Following community trends, some of the key capabilities of these platforms include consolidating threat intelligence feeds from multiple sources, automated identification and containment of new attacks, security analysis, and integration with other security

tools like Security Information and Event Management (SIEM), next-generation firewalls, and endpoint threat detection and response solutions [44]. Keeping these characteristics in mind, we present some of the best solutions available in the field based on [19] and [45].

3.2.1 IBM X-Force Exchange

IBM X-Force Exchange is a cloud-based threat intelligence sharing platform that can be used to rapidly research the latest global security threats and aggregate actionable intelligence - human and machine-generated - in order to help security analysts speed up a time to incident response [46].

3.2.2 Anomali ThreatStream

Anomali ThreatStream is an operational threat intelligence stream that, with the help of artificial intelligence, automates the threat intelligence collection and management lifecycle to speed detection and increase analyst productivity. Intelligence data is collected and curated from hundreds of threat sources - premium and open-source - to deliver a single high-fidelity set of threat intelligence [47].

A differentiating factor is its highly accurate machine-learning algorithm that assigns scores to IoC, allowing security teams to prioritise mitigation tasks. It also allows its integration with popular SIEMs and orchestration platforms in order to strengthen threat identification and remediation workflows.

3.2.3 ThreatConnect

ThreatConnect is a platform that enables automated data collection from multiple sources and presents it to users in context. Security teams can then analyse the information manually or with automation assistance to look for evidence of cybersecurity dangers. The platform shows associations in the data, helping specialists identify meaningful connections [48]. It also uses an intelligence-driven orchestration feature called *Playbooks* that can be used to trigger a task on a given event [49].

3.2.4 CrowdStrike Falcon X

Available in three types - Falcon X, Falcon X Premium and Falcon X, it is a threat intelligence platform that comes with automated malware investigation features, thus reducing

the time required to identify threats and determine the associated severity. Users can also benefit from intelligence reports that give daily alerts and offer strategic insights. By combining with other family products, it provides a user-friendly endpoint integration that does not require new installations or deployments. Also, top premium users are provided a cybersecurity expert researching specific threats and giving a customised report of the findings [50].

3.2.5 ThreatQ

Developed by ThreatQuotient, this platform is specifically aimed at optimising cybersecurity operations focused on threats, and very specially designed to alleviate the difficulties that SOC and Computer Security Incident Response Team (CSIRT) teams face in making decisions based on information provided by cybersecurity systems and external sources. It focus on automate ingestion, correlation, normalisation, contextualisation and prioritisation of artefacts to decide its importance and relevance, thus increasing the efficiency and effectiveness of the operation and reducing costs [51].

3.2.6 Malware Information Sharing Platform

Malware Information Sharing Platform (MISP) is an open source software solution mainly developed by the Belgian Defense Computer Emergency Response Team (CERT) and the NATO Computer Incident Response Capability with the goal of gathering information about malware and attacks, storing data in a standardised format and distributing cybersecurity IoC and malware analysis to trusted parties [52]. By taking advantage of its collaborative feature, users can enhance the knowledge of existing malware and threats and implement counter-measures. It allows to automatically exchange and synchronise the data with other previously defined parties and trust-groups, thus maintaining an up-to-date knowledge.

It it a component of the *Hive project* [53], a project design for cyber threat intelligence and incident response for SOC teams, that also includes the tools TheHive, a Security Incident Response Platform, and Cortex, an analysis engine.

3.3 IoT Honeypots

In IoT-based networks, new devices entering the network are automatically configured due to their open nature, which leaves these networks subject to many attacks, as described in [54].

Depending on their purpose, honeypots are often designed for specific scenarios. An example is Honeyd [55] - a simulated honeypot environment intended to simulate the virtual network topology to filter network packets based on user preferences. Honeyd supports TCP, UDP and ICMP protocols, simulating the TCP/IP stack that allows responding to requests from the virtual honeypots according to the services configured for each virtual honeypot. When sending the response packet, the personality engine makes it match the network behaviour of the configured operating system personality [56]. As it can emulate real operating systems, Honeyd can appear to the attacker as a router, web server, or DNS server, allowing the honeypot to blend into existing networks. With this behaviour, Honeyd can spoof responses about active fingerprint measurements, such as those used by the NMAP tool.

The increasing need for honeypots has led to the concept of Honeypot-as-a-Service (HaaS), where, instead of being configured locally, they are made available to users by cloud providers, thus eliminating the need to worry about hardware, software, and manpower to create and maintain the honeypot. This frees the user to focus more on managing traffic and defining decoy services. A simple scenario of this concept is demonstrated in [57], which proposes the use of public cloud provider honeypots that are made available to end-users as a service in order to filter incoming traffic where traffic is only accepted on a local network if the request meets a set of restrictions defined for a particular service; otherwise, it is forwarded to the honeypot. Although it reduces costs, it still requires labor to apply and maintain the configurations. An evaluation of this solution [58] shows that it still requires manpower to apply and maintain the settings, and the traffic flow is in order.

An introduction to the idea of honeypots that adapts to the needs and changes of organisations is not new. In [59] this idea is explored using Honeyd's low-interaction honeypots - which make use of unassigned IP addresses to launch instances - as a front-end to high-interaction honeypots where attackers can engage in real systems. With traffic being diverted in the background from one honeypot to another, this makes Honeyd honeypots more reliable for intruders. However, this requires setting up physical machines dedicated exclusively to the honeypot network, and as we would need to mimic more rogue

systems, more IP addresses need to be available as well.

A comprehensive solution for honeypot systems focused on IoT environments is the YAKSHA project [60], where it is provided a platform that allows an organisation to perform continuous monitoring and penetration testing of its infrastructure without the need to maintain such infrastructure nor having dedicated personnel. It allows a way to create customised honeypots according to specific needs, enabling compatibility to any service installed. The honeypots are monitored by tools that gather logs and other pieces of information to extract the attacker's steps and patterns which, in turn, are catalogued and made available to users by reports or a dashboard. Although it is mentioned a mechanism for applying patches and IDS configurations on vulnerabilities detect, it is not clear how this process is achieved.

Given the current cybersecurity landscape and the constant need to stay up to date on news topics and intruder tactics and techniques, it is necessary to create a system capable of taking a range view of the system and being up to date with the intruder's threats, tactics, and techniques. From this perspective, we chose some keys factors that today's solutions must answer:

- Support honeypot deployment: given the increasing number of attacks and tools, we need to be aware of the tactics and techniques employed without having to compromise real systems;
- CVE scanning: IDS systems are, mostly, not up to date on current threats that can compromise our systems, thus giving a false sense of security. CVE databases are known to be a reliable source of threats and can be used to enhance the IDS knowledge of threats;
- TSP connection: the use of these platforms has increased its popularity, with communities being able to share, in real-time, IoC of attacks. The connection with these platforms allows to keep stronger security perimeter with devices being capable of sharing artefacts with each and prevent attacks;
- Running in internal nodes: honeypots must be able to be deployed and accessible from devices inside the network without jeopardising the device and the network;

- Honeypot undetected: honeypots' instances must be as close as possible to a local device, providing a fully-functional operating system and user interactivity;
- Honeypot interoperability: the decoy systems must be fully compatible to a system shifting, i.e., be capable to run on local devices as well as cloud instances, if necessary;
- Architecture interoperability: instances must be deployed in any capable device, namely Raspberry Pi, regardless of its processor's architecture.

Table 3.2 shows the comparison of current state-of-the-art against our proposed solution.

	[55]	[57]	[58]	[60]	**STDM**
Honeypot Deployment	X	X	X	X	**X**
CVE Scanning					**X**
TSP connection (IoC)					**X**
Run on Internal Network	X			X	**X**
Undetected from Intruders					**X**
Run in Real Nodes of Network					**X**
Compatible with ARM					**X**

TABLE 3.2: State of the art's features summary

Chapter 4

Study on Host-based Intrusion Detection Systems

The growth and diversity of devices and the consequent lack of investment in security mechanisms create environments susceptible to attacks, given the exposure of these devices to the Internet and, consequently, to attackers. From this exposure, vulnerabilities - software errors and security holes - are discovered, thus allowing attackers to take advantage of the device to enter the network. A significant part of these vulnerabilities is defined under a CVE record, which is catalogued as a list and maintained by the National Cybersecurity Federally Funded Research and Development Center, operated by MITRE [61].

In this section, we evaluate the response of HIDS to publicly disclosed computer security flaws that have a CVE ID number assigned. We initially describe how we experimented, followed by the methodology applied, concluding with the results obtained.

4.1 Experimental setup and methodology

Given the importance an IDS system has in the security perimeter, a relevant aspect to consider is the reliability of these systems, particularly when faced with a variety of attack vectors. To address this concern, we conducted a study where we selected a set of services with a version not older than three years with a CVE ID associated and a set of popular and actively maintained IDS compatible with x86 and Advanced RISC Machine (ARM) architectures: AIDE, Fail2ban, OSSEC, Sagan, Samhain and Tripwire. We

test the behaviour of each HIDS against every CVE to verify if our exploit was detected or blocked.

The test architecture was configured considering the availability and accessibility of hardware on the market, and an operating system actively maintained and available for different processor architectures. Therefore, the tests were conducted on a client-server basis with the services being installed on a Raspberry Pi 4 model B with 4 GB of RAM and a desktop with processor Intel Core I7 4770 with 16 GB of RAM, with both devices running Ubuntu 20.04 LTS of 64 bits and connected via Ethernet cable to the router TP-Link TL-WR841N.

4.2 Results

Table 4.1 shows the results of our tests, where each row specifies the CVE exploited against the selected HIDS, where each row specifies the CVE flaw exploited. When a security solution is not designed to detect a particular vulnerability and the required mechanisms are missing, the event is considered impractical.

At first glance, the results show a negative performance by Sagan since this IDS was unable to detect any of the exploits. In a more detailed analysis, we highlight the results obtained with SQL injection attacks, which is a very popular technique, and most of the IDS were able to detect it. The exceptions were Sagan, which did not detect any attack, and Samhain, which was unable to detect the exploiting of the vulnerability in *OrderBy* clause in version 1.45 of SoPlanning (CVE-2020-9268). Given that attackers make use of known tools when exploiting these vulnerabilities, such as *sqlmap*, our tests show that signature variant IDS were able to detect those attacks by monitoring the directories and files that *sqlmap* uploads to exploit the vulnerability. On the other hand, the anomaly-based variant - namely OSSEC and Fail2ban - were able to detect the abnormal number of requests made by *sqlmap* and block their following requests.

Regarding the RCE technique, when exposed, the general result was a successful detection. Although it did not detect with their installed baseline, OSSEC and Fail2ban were able to block the requests after we added our handmaid rules which, given the variety of existing software, is difficult for an IDS to keep up with every piece of software. In the anomaly-based variant, the exception was, once again, Sagan that was unable to detect any behaviour anomalies. Given how this technique is used, signature-based IDS have

Attack vector	CVE ID	HIDS					
		Anomaly-based			Signature-based		
		Sagan	Fail2ban	OSSEC	Tripwire	Samhain	AIDE
Access Control Bypass	CVE-2019-13188	○	○	○	○	○	○
Buffer overflow	CVE-2018-1000030	⊗	⊗	⊗	⊗	⊗	⊗
DoS	CVE-2020-9283	○	○	○	⊗	⊗	⊗
	CVE-2020-6060	○	○	○	⊗	⊗	⊗
	CVE-2019-17498	○	○	○	⊗	⊗	⊗
	CVE-2019-16279	○	●	●	⊗	⊗	⊗
	CVE-2019-13115	○	○	○	⊗	⊗	⊗
	CVE-2018-7182	○	○	○	⊗	⊗	⊗
	CVE-2018-8712	⊗	●	●	●	○	○
MitM	CVE-2019-6110 / CVE-2019-6111	○	○	○	⊗	○	⊗
Password Hash Disclosure	CVE-2019-13349	○	○	○	○	○	○
Path traversal	CVE-2018-12015	⊗	⊗	●	●	●	●
Privilege escalation	CVE-2019-14287	○	○	●	⊗	⊗	⊗
	CVE-2019-9891	⊗	⊗	⊗	⊗	⊗	⊗
	CVE-2019-9320	○	○	●	●	●	●
	CVE-2018-10933	○	○	●	●	●	●
	CVE-2019-5736	○	○	●	●	●	●
RCE	CVE-2020-7246	○	●	●	●	●	●
	CVE-2019-16278	○	●	●	●	●	●
	CVE-2019-15642	○	●	●	●	●	●
	CVE-2019-15107	○	●	●	●	●	●
	CVE-2019-12840	○	●	●	●	●	●
	CVE-2019-11043	○	●	●	●	●	●
	CVE-2019-9624	⊗	⊗	⊗	●	○	●
	CVE-2019-7731	○	●	●	●	●	●
SQL Injection	CVE-2020-9340	○	●	●	●	●	●
	CVE-2020-9268	○	●	●	●	○	●
Unauthorised file Read	CVE-2020-1938	○	○	●	⊗	⊗	⊗
User enumeration (brute-force)	CVE-2018-15473	○	●	●	⊗	⊗	⊗
XSS	CVE-2019-13189	○	●	●	⊗	⊗	⊗
XXE	CVE-2019-15641	○	○	○	⊗	⊗	⊗

● Detected ○ Undetected ⊗ Impractical

TABLE 4.1: Experimental trials on HIDS

difficulty in detecting these attacks, and its detection success resides in the attacker creating and/or modifying some file or directory within the IDS monitoring scope. Thus, OSSEC was the only one able to detect it on both approaches.

When looking at DoS attacks that cause serious harm and jeopardise the availability principle, only anomaly-based approaches were considered, given the specificity of this technique. In general, the HIDS were unsuccessful in detecting the exploitation. The exception was the case of CVE-2019-16279, where we caught the abnormality of requests using a handmaid rule, blocking the attacker. However, even with the blocking of malicious requests, our tests showed a mixed behaviour in the availability of the service: in some cases, the service remained available after blocking the requests, and in others, the service remained unavailable.

Looking into the remaining results, Knowage version 6.1.1 has associated CVE-2019-13188 and CVE-2019-13349, which were not detected. However, because the vulnerability is essentially the application's failure to verify user permissions, blocking access to these requests could disrupt the application and compromise its availability. The path traversal and MitM attacks also proved to be extremely difficult to detect, being possible to spot on the modification reports provided by the HIDS as our tests showed.

Another service under analysis was the Secure Shell (SSH), a service widely used to manage systems and applications remotely. Given the popularity and importance, using the baseline provided by OSSEC and Fail2ban, these IDS were able to block the user enumeration attack as exposed in CVE-2018-15473, being the only anomaly-based vulnerability that did not require any handmaid rule.

Lastly, we look into XSS and XML External Entities (XXE) attacks. The XSS vulnerabilities are typically found in web applications and consist of enabling attackers to inject client-side scripts into web pages viewed by other users. In case of CVE-2019-13189, the XSS was on the vulnerable parameters *start_url* and *user_id* on the *ChangePwdServlet* page. When exploited, this vulnerability was detected only by OSSEC and Fail2ban, again by using a custom rule. The same did not occur with CVE-2019-15641, that is a vulnerability on the *XML::Parser* service [62], the handler of XML messages. Because it is used without preventing the use of entities, a successfully logged attacker can exploit a XXE in order to retrieve a local file or discover internal networks with root rights. When tested, the anomaly-based IDSes were unable to detect the exploitation of this vulnerability.

The list of CVE flaws tested with a detailed description and reproduction scripts can be found in a GitHub directory [1].

[1] https://github.com/simao-silva/iot-cves

Chapter 5

System design

In this chapter, we will describe the technical implementation of our solution.

As mentioned in section 1.3, our solution aims to achieve the following goals:

- **Goal 1** - Provide an autonomous threat detection flow able to analyse the diversity of devices in the network and detect and mitigate, if possible, its vulnerabilities;
- **Goal 2** - Creation of automatic honeypot instance deployment when a vulnerability is present in a device in the network, capable of running in a variety of devices including in low-power, performance-constraint devices;
- **Goal 3** - Provide a mechanism that can discretely monitor honeypots instances to collect intelligence of tactics and techniques of intruders.

5.1 Architecture

The gather of threat knowledge is one of the most important and necessary tasks in today's cybersecurity. The increase of attacks led to changes in collecting knowledge of (potential) threats and how to take advantage of that knowledge.

Figure 5.1 shows the overview of the architecture of our system. The TSP component will update their database with the CVE entries (1). The *Sandboxing for Threat Detection and Mitigation (STDM) Central Coordinator* will act as a go-between managing the devices scanning history and honeypots. From time to time, the devices will send their list of software installed to the *STDM Central Coordinator* that, in turn, requests from the TSP the information about CVEs (2). For each CVE entry, the coordinator parses the CPE list and searches for a match in the list received from the devices (3-4). When a match is found,

the *STDM Central Coordinator* will select a random device and launch a honeypot with the same operating system and software version of the match (5). Inside the honeypot, an admin user can verify if the match is not a false positive and provide that information to the *STDM Central Coordinator* (6). From the honeypot logs (7), an administrator user can observe the malicious requests and can create, if possible, rules to be applied to the *HIDS* of all devices (8).

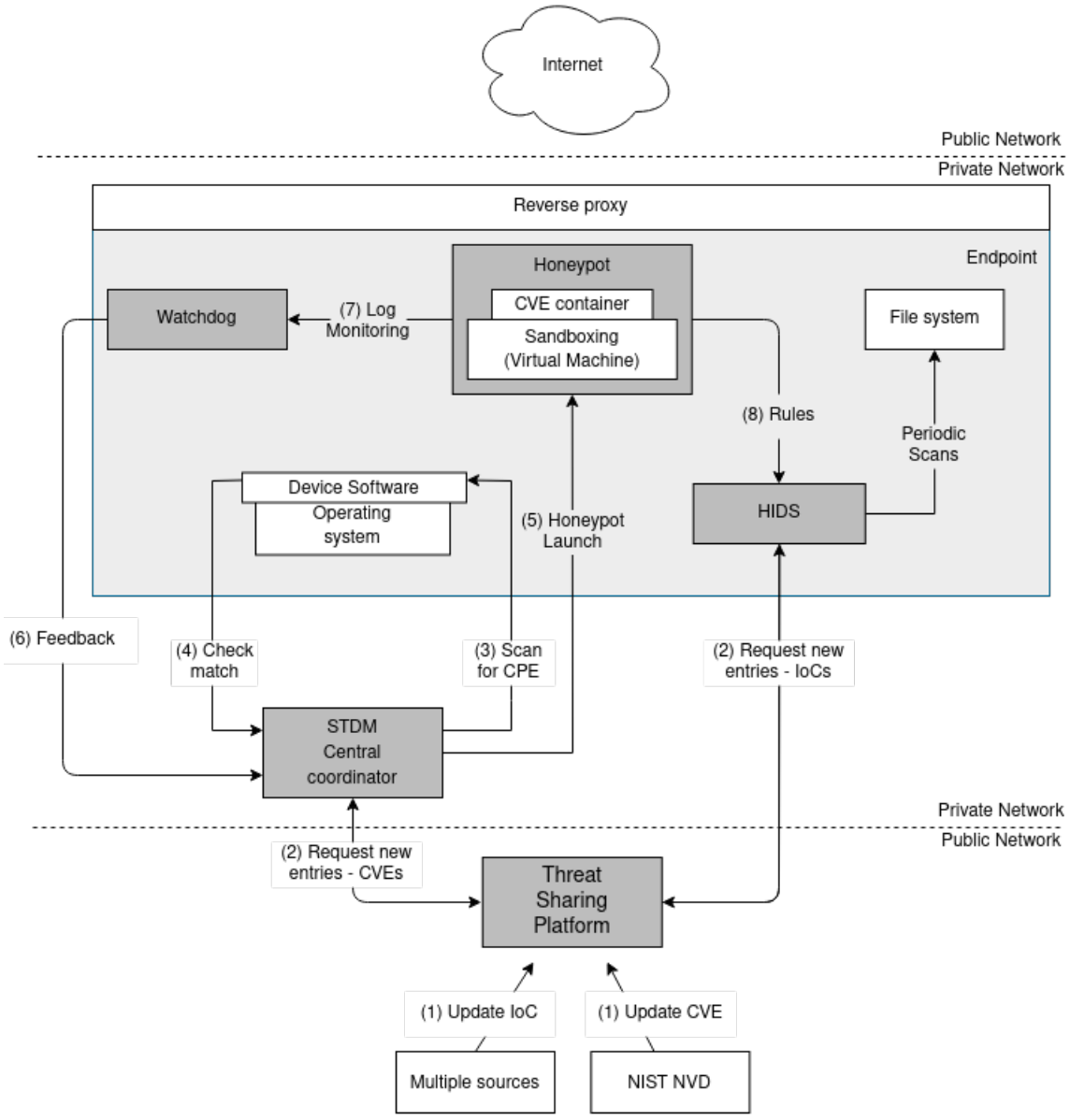


FIGURE 5.1: System architecture

5.2 Components

In this section we describe the highlighted components of our architecture.

5.2.1 Threat Sharing Platform

This component is responsible to store information regarding IoC and CVE for our setup. From time to time, the platform will retrieve IoC from public, trustworthy feeds and act as a local mirror database of *NIST*'s NVD database of CVE entries.

5.2.1.1 Indicators of Compromise

Even in cases of breaches, we can still learn with the shreds of evidence left by the intruder, known as IoC [13]. These IoC indicate the existence of security breaches in some devices or systems. Among the examples, it is possible to explore the accesses to malicious or phishing sites, e-mail issues used in spear-phishing campaigns, among many others. IoC are collected after suspicious activity and, on a regular basis, data is verified to detect vulnerabilities. In addition, the use of these indicators allows the creation of intelligent tools, which identify and isolated suspicious files, for example. We use these IoC to capture malicious activities on the network at its initial stage, preventing them from becoming bigger problems and compromise the security of the organisation's users.

5.2.1.2 Common Vulnerabilities and Exposures

Also, in our approach, we intend to enrich our knowledge through known vulnerabilities, namely CVEs [63]. CVE is a public service that catalogues vulnerabilities and related information about publicly known security flaws. When someone mentions a CVE, it is usually by referring to the ID number assigned to a security vulnerability. As they are public, we can recreate the environments that mimic that CVE and use it to gather IoC that can be later used to enhance our HIDS, as shown in steps 5, 7 and 8 of Figure 5.1. For that, those environments are launched as honeypots and are exposed to the Internet as decoy services.

5.2.2 Host Intrusion Detection System

To improve security in our devices, we deployed two HIDS systems - one that offers both anomaly and signature variants and the other that offers IPS capabilities - on each

device given that these systems are designed to enhance the protection against internal and external threats. Also, they can monitor network traffic to and from the machine, watch running processes, and inspect the system's logs in search for patterns that match known cyber attacks. Although they can be seen as limited, given their low visibility (limited to the host), decreasing the decision-making context, their deep visibility into the host's internals allows us to analyse activities with a high level of detail. Thus, unlike NIDS [64], they can directly access and monitor data files and system processes targeted by attacks.

5.2.3 STDM Central Coordinator

This component is the gateway between the threat-sharing platform and the devices in the network. It is responsible for checking if a device in our network has vulnerable software, keep records of those threats, and launch the honeypots containing those threats.

5.2.4 Watchdog

The *Watchdog* is an agent installed on the host's system to able users to access any file from the honeypot without having to directly interact with it or learn the backend command syntax to access it. From the *Watchdog*, a user can monitor the logs from the vulnerable service that is up and running on the honeypot instance. With that information, it can analyse logs and parse malicious queries that can be manually added later to the IoC database in the TSP. This enables the platform to be aware of inside attacks viewed or received by the honeypots allowing prioritisation of the threats in a global overview.

The *Watchdog* can be adapted to any sort of installation of service in the virtual machine.

5.2.5 Honeypot

The *Honeypot* instances are virtual machines with service(s) exposed to the Internet. The installation of those services is done directly on the file system or by using containers (to avoid external changes that can interfere with the service availability and expected behaviour). The instances are prepared to run in x86-64 and ARM architectures, thus enabling (virtually) the launching of an instance on any device on the network. Also, this mobility enables that a given instance running a given service can be easily shifted between devices.

The nature of these instances requires isolation techniques so that the attacker can not gain access to the host system and damage it. The common strategy used in the cybersecurity world is sandboxing, which is a security mechanism that tricks an application or program into thinking it is running on a regular computer [65]. This allows us to provide services to intruders in a tightly controlled environment without allowing the services and the intruders' actions to harm the host device.

Chapter 6

Implementation

Following the presentation of our architecture and its components in Section 5, this chapter describes the technical details of the implementation of our system, the decisions made, and problems that emerged during said implementation.

6.1 Threat Sharing Platform

For our architecture, we opt to use MISP as our threat (intelligence) sharing platform. It is an open-source software solution for collecting, storing, distributing, and sharing cybersecurity indicators, threats about cybersecurity incidents analysis, and malware analysis [52].

Interactions with MISP can be done by its Representational State Transfer (REST) interface. Using PyMISP, a Python library developed to access MISP platforms via their REST Application Programming Interface (API) [66], we can manage the platform and add new functionalities. In our case, we use it to keep our database of IoC and CVEs updated.

Regarding CVEs, we keep our database updated by a custom module-like script we developed, henceforth referred to as *CVE module*. That module will use NIST's REST service to periodically request new or recent modified entries from the CVE database since the last request. The service will answer with an empty response - meaning that we are up to date - or a list of entries with their respective details in JavaScript Object Notation (JSON) format.

For each element in that list, we must verify the contents of the *input* field. If the field is empty, the CVE is under analysis, and the information about this vulnerability is still preliminary. If not, we proceed to make a subsequent request to retrieve the CPE software

list affected by the vulnerability. Using the information from both requests, namely ID, description, reference links, and CPE list, we use the CVE module to create a MISP event with all these attributes and store it on the platform, making it available to the other components of our system. Figure 6.1 shows the result of an event created using the CVE module.

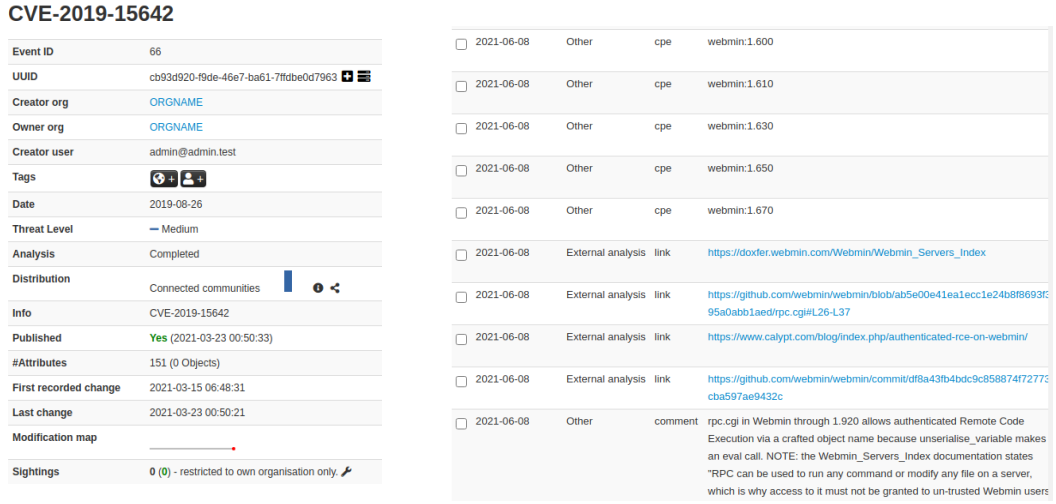


FIGURE 6.1: Example of a MISP event

6.2 STDM

This component is the bridge between the network devices and the TSP. Figure 6.2 shows the an overview of the component' workflow.

The component is composed of a server with a database storing the values of the device's last authentication timestamp (for debug) and several vulnerabilities found (for statistical purposes). All communications between devices and our server use SSL/TLS to provide confidentially and integrity. The devices will authenticate themselves into the component using a certificate-based authentication where, for each device, it is issued a certificate signed by our own Certificate Authority (CA), with the certificate being later installed on each device manually. This method allowed us easier management of the devices, given that only devices with a valid certificate installed can access our system. Then, they will periodically send to the server the names and versions of the software installed on the host (1). In turn, the server will request the TSP platform for the list of CVEs and respective CPEs (2-3) and verify if there is a match against the information received from the device (4). If a match is found (5), it increments the positive matches

statistics and launches a honeypot instance with the same operating system and informs the admin the vulnerable software version so it can later install it on the honeypot. The honeypot instance starts on a random device that has the necessary resources to run the instance, and it is available in the network at the time of the launch.

The software matching is not linear and requires a pre-processing step. Given that the names in NVD's database are in CPE syntax and differ from the syntax used by operating systems, we attempt to translate the CPE syntax closer to the names on the operating systems. However, in some situations, we notice a high rate of false positives, which we attribute to the versioning naming format. In these cases, a warning is displayed to an admin user that has to manually check if it is a match.

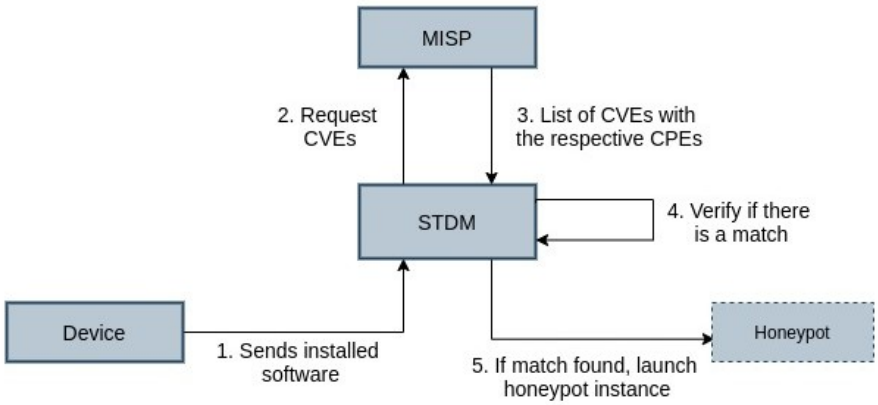


FIGURE 6.2: Overview of the STDM workflow

6.3 Host Intrusion Detection System

For our solution, we opt to use a combined solution of HIDS by leveraging the Fail2ban's IPS characteristics. As HIDS are *passive* in its essence, meaning they identify but do not prevent suspicious activity, we chose to add an IPS system as they are active in preventing those suspicious activities. Thus, to take advantage of all of their features, we used OSSEC (HIDS) combined with Fail2ban (HIDS/IPS).

OSSEC is a comprehensive free and open-source HIDS solution that performs log analysis, file integrity checking, policy monitoring, rootkit detection, and active response using both signature and anomaly detection methods. It can connect to MISP, our threat sharing platform, to keep up with IoC and improve its detection motor. Also, it comes with out-of-the-box support to a comprehensive set of services, thus decreasing the need

to add custom services profiles. It was installed in each device using its *local installation* method with devices performing all operations on the device level.

With Fail2ban, we have a tool that helps to stop, prevent or slow down attacks in (near) real-time. It offers a simple interaction and user-customisation. Users can define their filters (malicious patterns) and then reference them when adding the service profile to the *jail* [67], a concept that Fail2ban uses to apply rules to any given application or log file. In *jail*, the user configures various settings such as the service port, the log file path to apply filters, and the ban time, among others, that are needed so that the filters know what to do when capturing an abusive IP address.

6.4 Isolated environments

Honeypot security creates a vulnerability in the host's that is deliberately highlighted for attackers to use. Also, there is a need to deploy these systems in secure environments so that activities in the honeypot systems do not affect the host. With that goal in mind, in this section we describe the technologies and methods applied to achieve that goal.

6.4.1 Virtualisation

Virtualisation was chosen technology for our honeypots because the environments are not shared between host and guest's systems. For security reasons, unlike containers, the kernel and libraries are not shared between host and guest to prevent an attacker from accessing the host machine. From the functionality point of view, it allows the creation of several isolated environments from single hardware, thus allowing better usage of resources. While containers are highly supported for our architectures (x86 and ARM), the isolation provided by virtual machines was the key factor to consider.

Virtualisation technology has many solutions for x86 architectures, but the same does not apply to devices with ARM architectures. However, continuous improvements in boards, such as the Raspberry Pi, have made it feasible to use Kernel-based Virtual Machine (KVM), a Linux kernel module that allows the Linux kernel to act as a hypervisor, which has become more accessible for users with Linux distributions with the precompiled Linux KVM kernel module.

The chosen virtual machine provider was Multipass [68], a lightweight virtual machine manager. Maintained by Canonical, it offers a simpler and easier command-line

interface with booting from a virtual machine that only requires a single command line. Its choice was due to the support of different architectures and operating systems, allowing it to scale its use to any system.

For the management of virtualisation, we had to do some changes in Multipass settings. By default, it uses the QEMU hypervisor [69] that, although it runs perfectly on x86-based processors, presented some incompatibility issues when running on ARM processors making Multipass was unable to run. After reading documentation and community forums, we change the hypervisor to LXD [70], a REST API that connects to libxlc, the Linux Containers (LXC) library - a solution for virtualising software at the operating system level within the Linux kernel [71]. Initially developed for containers only, LXD supports the launching of virtual machines since version 3.19 and offers an experience on syntax and performance very similar to containers. With the usage of this hypervisor, it created the opportunity to monitor the virtual machines from the host's filesystem, as discussed in section 6.5. Since on the operating system level the interaction is done using *lxc* command, we will refer to them as *LXC/LXD*.

Other virtualisation hypervisor solutions were considered and tested. Popular solutions like VirtualBox, VMware Workstation, and VMware Player were soon discarded since they require an x86-64 host so incompatible with our flexible honeypot solution. Although not ideal for our architecture, we also tested hypervisors that run directly on the host's physical hardware (also known as *type-1 hypervisors*), namely VMware ESXi and Xen Project, that were discarded given its need for external devices to store the virtual machines and performance issues.

6.4.2 Honeypot

The honeypot system is a Multipass virtual machine that is deployed by the *STDM Central Coordinator* component. Currently, only Ubuntu cloud images can be deployed automatically. However, the usage of custom images is also supported.

The system's image is very similar to a non-graphical version of Ubuntu Server with no visible distinguishability that can hint the intruder that it is a decoy. Once up and running, an admin user sets up the vulnerable service, including the additional required software and the necessary port-forwarding with IPtables. The services are installed preferably from the source code of the corresponding vulnerable version and, when possible,

deployed on containers to take advantage of its portability so that installation is environment independent and the service easily deployed (it can start in just a few seconds) with minimal overhead.

6.5 Watchdog

We have two ways to get access to the honeypot: by using the Multipass interface or the LXC/LXD interface. Although both options allow us to access the honeypot and monitor the services' logs, they came with setbacks that can reveal to the attacker that he/she is in a decoy system. Using the Multipass interface, since it uses the SSH service as a backend, an attacker can monitor the SSH log file and discover a user with *Sudo* privileges monitoring the log files, which can be an admin user and thus jeopardise the decoy system. Instead, if the LXC/LXD interface is used, the attacker can be suspicious of the system if it discovers that the *lxd-agent* is running (given this is necessary for the host-honeypot communication) or if a process monitoring some log file own by the root is running. The resolution of these issues were two: the first one is mounting the */proc* directory with *hidepid=2* flag, thus deny users access to processes besides theirs; the seconds one is by unmounting and mounting the system image file (.img) used to support the virtual machine on the host's file system. This second option comes with the inconvenience of needing to repeat the unmount / mount routine due to a limitation on the image file that supports the virtual machine file system. When mounted, we notice that later changes made within the virtual machine's file system are not reflected in the mount point and vice versa. However, if we repeat the unmount/mount routine on the image file, we can see the changes we made earlier. This limitation forces us to redo the unmount/mount routine whenever we need to get new data from the log files and makes any real-time log monitoring completely impossible.

Given the options above, we opt to use the LXC/LXD interface. Even though the flag usage can indicate the presence of a decoy system, it can also be seen as a default security measure employed by an admin. Also, considering the usability trade-off between the two resolutions, the usage of the LXC/LXD interface provides a better user interaction.

To avoid the need of admin users to learn and remember the syntax of the command to access the instance, we developed a script that abstracts the backend command and that also accepts user input so that user can execute commands as if it were inside the instance.

Chapter 7

Evaluation

In this section, we aim to evaluate the performance of our system. Our evaluation is centred on the measurement of delay between the time attack is detected (i.e., the malicious request is detected) on the honeypot and the time the central coordinator receives that information.

The goal of our evaluation is to demonstrate the feasibility of the solution.

For the performance evaluation, the main challenge was to measure and compare with a standard service in the literature. An example of a type of this service is Apache2 [72], which is used in research articles to detect and demonstrate performance overhead associated with new implementations. *André Brandão et al.* [73] is an example of the usage of Apache2 with Intel Software Guard Extensions (SGX) to encrypt the HTTPS private key inside an enclave. As it is one of the most used web servers, we use a vulnerability of this service to demonstrate our prototype. For this, we exploit the flaw CVE-2019-10092 in the vulnerable Apache HTTPd 2.4.38 server.

7.1 Environment description

For our scenario, we used a Raspberry Pi 4 Model B with 4GB of RAM with a Broadcom BCM2711 1.5 GHz Quad-Core 64-bits processor connected through Ethernet cable to our router Thomson TG784n. Also, we acquire a virtual machine from Microsoft Azure, a known cloud provider, with 2 GB of RAM and an Intel Xeon E5-2673 v3 2.40 GHz processor truncated at 2 cores. Both used Ubuntu 20.04 LTS as the operating system.

7.2 Methodology

We designed our testing scenarios to consider different honeypot installations. We tested our system in the following scenarios:

- Setup 1 (HaaS with physical machines): cloud virtual machine exclusively dedicated to be a honeypot ;
- Setup 2 (HaaS with virtual machines and using LXD): using Multipass instances as honeypots on the cloud virtual machine to evaluate the impact of virtualisation overhead;
- Setup 3 (*STDM*): our solution running in a local network with a Raspberry Pi running Multipass instances as honeypots.

The tests were performed to explore the differences against the HaaS concept and usage of cloud machines to compare the delay times between on-premise and off-premise solutions (State-of-the-Art). Given the usage of virtualisation, we also tested its impact on the overall results. Scenarios from setup 1 and setup 2 allow us to test and compare the State-of-the-Art solutions with our solution (setup 3).

Regarding deployment, we launch the vulnerable Apache Server from a container available on a Docker Registry to simplify the installation process in all setups. For setups 1 and 2, to allow queries from cloud machines to reach the central coordinator, we had to configure IPTables rules and delete the default Azure forwarding rules that were causing the requests to be dropped. For setups 2 and 3, we deployed Multipass' virtual machine, installed the Docker software, and configured the necessary redirects from the host to the virtual machine, so that we could reach the server from the outside.

The attack scenario was performed on a client-server basis where the attacker performs a XSS attack in *mod_proxy* error page. In a server setup with proxy module enabled, but misconfigured in such a way that shows a proxy error page, an attacker can take advantage of the vulnerability present when the path in the Uniform Resource Locator (URL) is being parsed and create a custom anchor tag. By leveraging URL encoding of the backslash ("\") character ("%5c"), the attacker can make the anchor tag point to any site and launch further attacks from there.

We measure the delay times between the moment the malicious request is detected and the moment the information is recognised by the *STDM Central Coordinator*. For this,

a specific tool was designed, located precisely in the *STDM Central Coordinator*, which provides the network latency generated (in milliseconds) by our solution. The exploitation was performed using a bash script containing a cURL request to the server's IP address.

7.3 Results

In this section, we aim to show the results and comparison with state-of-the-art.

Table 7.1 presents the results for each of the three scenarios network throughput. For gathering the network information, we used *iPerf* [74], which is software used to test the network bandwidth. We collected two-time samples to measure the setup performance only in terms of network bandwidth. The results presented in the table are the mean and standard deviation for the three setups and both measurements (TCP bandwidth and UDP jitter).

	Setup 1	Setup 2	Setup 3
TCP bandwidth (Mbits/sec ± sd)	1,22 ± 0,89	1,17 ± 1,06	30,29 ± 2,91
UDP jitter (ms ± sd)	4,02 ± 1,12	4,32 ± 2,88	0 ± 0

TABLE 7.1: Latency results for the three setups

Figure 7.1 represents the delay associated with each of the implementation scenarios. The latency of the process between the detection of the malicious request in the honeypot instance and the moment when the *STDM Central Coordinator* receives this information. To do this, we collect three samples for each number of requests.

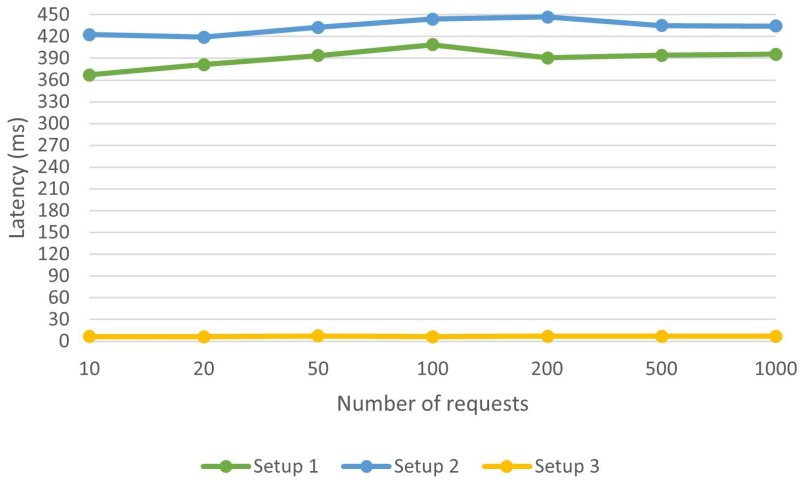


FIGURE 7.1: Latency (ms) per number of requests

The results of Setup 1 (HaaS with physical machines) show a delay in the order of the 390ms with minimal oscillations. Then, Setup 2 (HaaS with virtual machines and using LXD) presents small delay increments compared to Setup 1. This shows that using virtual machines has a residual overhead on performance and is, therefore, a good choice when looking for multiple deployments and host isolation. On the other hand, the results of Setup 3 (STDM - our implementation) show the main advantage of having Honeypots on the network endpoints because the delay associated with this task is reduced to close 0 with on-premises honeypots. Similar behaviour could be achieved with Honeypots running on a separate Local Area Network (LAN), but these solutions raise concerns about the actual usability of the honeypot network (as shown in related works).

Chapter 8

Security Analysis

In this section, we provide a security/threat analysis of the proposed system. We identify potential threats of the implementation and define how an attacker may attempt to exploit the system and the limitations we introduce to block that threat.

8.1 Attacker compromises the Sandbox system

When an attacker exploits the target CVE, it could find a vulnerability in STDM that allows the execution of unexpected commands in the Sandbox. These flaws can leave the network vulnerable as the attacker may try to gain root access or other privileges that allow access to other machines.

To mitigate these types of threats, we set up the information extraction process from the Sandbox always in reading mode and directly from the file system, providing an extra layer of security. Thus, there are no writes or manipulations of information inside the Sandbox, but always from the host machine.

Also, the *Watchdog* component has rules for the detection of misbehaving communications (locally or remote).

8.2 Unknown Zero-Day detected

Zero days can occur within the local network because if an attack is detected, a report is generated with an alert containing the network traffic/log files manipulated to be analysed by a SOC team.

8.3 Physical access to the machine

If an attacker can compromise the local machine where the honeypot is running, the attacker can modify the STDM code to read and try to write information to impersonate it, and the associated communication to the *STDM central coordinator*.

In this new scenario, the attacker will attempt to upload misleading information to the central server, but the information will not match the information provided by other STDM instances running on other devices, making the attack detectable.

8.4 Denial-of-Service scenario

The DoS attack limits the machine or network resource unavailable to its real users, temporarily disrupting the services of a network component connected to the Internet.

In this scenario, our implementation may suffer from a DoS to the service being tested by the STDM, but STDM will immediately detect it by the *Watchdog* component to stop the running virtual machine and report the behaviour to the local machine. If the local machine experiences a general DoS to the host, it can cause temporary failure in the ability to communicate with the central node. There are two main solutions: HIDS can detect this DoS and block the attempt; or if it is a Distributed Denial-of-Service (DDoS), it may not handle the request, so it stores all the information until the attack stops.

Chapter 9

Conclusion

The IoT concept is growing at an incredible rate with more and more devices available every day, thus making security an urgent need and challenge.

Traditionally, maintaining network security has involved acting vigilantly by using network-based defence techniques such as firewalls, intrusion detection systems, and encryption, but the current situation calls for more proactive techniques to detect, deflect and neutralise attempts to illegally use information systems. In this scenario, the use of honeypots is a proactive and promising approach to combat threats to network security.

In this work, we studied and created a solution to autonomously detect and prevent threats based on public information of known threats shared by government entities and the community and connecting it to specialised software that parses and analyses that information so that users can have a perspective of what needs to be improved. In addition, using known and effective ways to gather intelligence and attacker tactics and techniques, we can leverage this knowledge and improve our defence mechanisms to strengthen our security perimeter and make our systems more resistant to future attacks.

9.1 Limitations

Multipass proved to be an effective tool for launching virtual machines on devices with few resources, expanding its capabilities and opening horizons for new usage scenarios. However, it requires a minimum of 10GB of free space to launch an instance, which can be a setback for many of these devices. Also, due to the way Multipass handles changes within its instances, there is no current mechanism that allows complete monitoring of an instance without limitations or privileged internal access.

The proliferation of single-board computers - namely Raspberry Pis - alongside its lower power consumption and lower price when looking to comparable alternatives has increased its presence and usage at home, on industries, or in concepts like smart cities. However, as we notice in our tests, these devices faced some performance issues, especially when dealing with virtualisation technologies. Upon the initialisation of Multipass' instances, these devices take some extra time to complete that process which can be, in some scenarios, a discouraging factor.

Given the solutions available on the market, we noticed the difficulty in creating an automatic mechanism to add custom rules in HIDS. In our research, we noticed the use of different formats and syntaxes, which makes a standardisation process impossible and makes interoperability difficult.

With Multipass being developed by Canonical, currently, only Ubuntu-based images are provided out-of-the-box, thus limiting the access to other operating systems.

9.2 Future Work

As future work and applications, we have some paths that we want to follow. We want to enhance the system with human-in-the-loop mechanisms because, in addition to the supervision performed automatically by the system, a user must be able to add their domain knowledge that could identify points of failure and new rules for improving the detection of abnormal patterns, also reducing false alarm rates and adversarial attacks. As misclassification can have serious consequences, human-in-the-loop should be used to confirm all patterns and decisions. In this sense, human-in-the-loop in IPS and IDS systems can validate the abnormal events and generate rules to feed the first defence layer. On our current implementation, with Fail2ban an administrator can add rules, but we want to make more steps on it for the overall system, giving the possibility to launch honeypots with new rules.

If an attacker gains access to the system, they may attack the system (as explained in the security analysis section), but one solution to blocking this type of attack is to use multiple software vendors to perform this process. In combination with Byzantine Fault Tolerance (BFT), it will allow the mitigation and detection of another vector of attacks to the host machine, allowing to identify wrong answers from the system. BFT replication can be used as a solution to handle Byzantine faults of both accidental and malicious nature, reaching consensus (agreement on the same value) even when some of the nodes

in the network fail to respond or respond with incorrect information (as an example, an attacker to the physical machine with misleading information). The goal of a BFT mechanism is to safeguard against system failures by employing collective decision-making (both – correct and faulty nodes), which aims to reduce to influence of the faulty nodes.

Given the popularity of containers, thanks to their lightweight and portability proprieties, we intent to add a local registry container, with containers of vulnerable software versions found in our devices or other flawed software that we want to collect artefacts. This opens the possibility of increasing the deployment process of a honeypot instance since it can be done exclusively by automation.

9.3 Conclusions

We offer a new security approach to deploying dynamic local honeypots capable of running on devices not previously used for this purpose. As far as we know, this is the first proposal for a dynamic honeypot that can be installed on any network device in order to test network devices. Thus, it takes the opportunity to confuse the attacker, as he/she never knows if it is a network device or a honeypot placed for this purpose. IoT environments benefit from this solution as sensors are often the most vulnerable devices on a network.

It combines IDS and IPS systems to detect and automate the honeypot initialisation process according to what is happening in the network (environment). This process can be used as the first line of defence for active intrusion detection and prevention (or, in the worst case, logs so admins can do manual prevention), as it also allows for gaining insights into new attacks. The implemented system focuses on the early discovery of viruses on the network and malicious network activities.

Contrary to previous work, the idea is not to have a fixed honeypot but to have a set of devices that can be used at any time as a honeypot (adapted to the most recent threat) to test the network for these possible threats and then report to the MISP. Also, we developed a honeypot solution that can run on multiple devices of different architectures without having to allocate specific hardware.

We have experimentally demonstrated that it is possible and feasible to deploy decoy systems in on-premises infrastructure, even on power-constrained devices. Regarding latency and comparing to standard HaaS, we achieved a reduction from 360ms to closer to 0ms.

Acronyms

AIDE Advanced Intrusion Detection Environment 13, 15, 21

API Application Programming Interface 31, 35

ARM Advanced RISC Machine 21, 29, 34, 35

BFT Byzantine Fault Tolerance 44, 45

CA Certificate Authority 32

CERT Computer Emergency Response Team 17

CPE Common Platform Enumeration 10, 26, 31–33

CSIRT Computer Security Incident Response Team 17

CVE Common Vulnerabilities and Exposures 7, 10, 19, 21, 22, 25, 26, 28, 31, 32, 41

DDoS Distributed Denial-of-Service 42

DNS Domain Name System 8, 18

DoS Denial-of-Service 9, 23, 24, 42

HaaS Honeypot-as-a-Service 18, 38, 40, 45

HIDS Host-based Intrusion Detection System viii, 6, 11, 13, 14, 21–24, 27, 28, 33, 42, 44

HTTPS Hyper Text Transfer Protocol Secure 37

ICMP Internet Control Message Protocol 5, 18

IDS Intrusion Detection System 3, 7, 10–13, 19, 21, 22, 24, 44, 45

IoC Indicators of Compromise 8, 16, 17, 19, 28, 29, 31, 33

IoT Internet of Things ii, 1, 5–7, 9, 13, 18, 19, 43, 45

IP Internet Protocol 8, 18, 19, 39

IPS Intrusion Prevention System 14, 28, 33, 44, 45

IT Information Technology 2, 3, 11

JSON JavaScript Object Notation 31

KVM Kernel-based Virtual Machine 34

LAN Local Area Network 40

LXC Linux Containers 35, 36

MISP Malware Information Sharing Platform 17, 31–33, 45

MitM Man-in-the-Middle 9, 23, 24

NIDS Network-based Intrusion Detection System 11, 29

NIST National Institute of Standards and Technology 10, 28, 31

NVD National Vulnerability Database 10, 28, 33

OSSEC Open Source Security 13–15, 21–24, 33

RCE Remote Code Execution 9, 22, 23

REST Representational State Transfer 31, 35

SGX Intel Software Guard Extensions 37

SIEM Security Information and Event Management 16

SOC Security Operations Centre 3, 4, 17, 41

SQL Structured Query Language 9, 22, 23

SSH Secure Shell 24, 36

STDM Sandboxing for Threat Detection and Mitigation 26

TCP Transmission Control Protocol 18, 39

TSP Threat Sharing Platform ii, 19, 20, 26, 29, 32

UDP User Datagram Protocol 18, 39

URI Uniform Resource Identifier 10

URL Uniform Resource Locator 38

VLAN Virtual Local Area Network 5

XML Extensible Markup Language 10, 24

XSS Cross-site scripting 9, 23, 24, 38

XXE XML External Entities 23, 24

Bibliography

[1] E. Manavalan and K. Jayakrishna, "A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements," *Computers & Industrial Engineering*, vol. 127, pp. 925–953, 2019. [Cited on page 1.]

[2] C. Zhang and Y. Chen, "A review of research relevant to the emerging industry trends: Industry 4.0, IoT, blockchain, and business analytics," *Journal of Industrial Integration and Management*, vol. 5, no. 01, pp. 165–180, 2020. [Cited on page 1.]

[3] J. Cheng, W. Chen, F. Tao, and C.-L. Lin, "Industrial IoT in 5G environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, pp. 10–19, 2018. [Cited on page 1.]

[4] I. I. Editors, "US adults added 1 hour of digital time in 2020," https://www.emarketer.com/content/us-adults-added-1-hour-of-digital-time-2020, 01 2021, (Accessed on 19/08/2021). [Cited on page 2.]

[5] "Malware Statistics & Trends Report," https://www.av-test.org/en/statistics/malware/, (Accessed on 26/01/2021). [Cited on page 2.]

[6] R. Richardson and M. M. North, "Ransomware: Evolution, mitigation and prevention," *International Management Review*, vol. 13, no. 1, p. 10, 2017. [Cited on page 2.]

[7] O. Filipec and D. Plasil, "The cybersecurity of healthcare: The Case of the Benegov Hospital Hit by Ryuk Ransomware and Lessons Learned," *Obrana a Strategie-Defence & Strategy*, pp. 27–51, 2021. [Cited on page 2.]

[8] D. Miller, "Industrial Cybersecurity Concerns Heat Up in The Era of COVID-19," https://www.automationworld.com/cybersecurity/article/21354953/crowdstrike-releases-2021-cybersecurity-global-threat-report, (Accessed on 21/09/2021). [Cited on page 2.]

[9] N. Strother, "Cybersecurity Threats Remain A Concern For Global Corporations," https://www.forbes.com/sites/guidehouse/2021/01/08/cybersecurity-threats-remain-a-concern-for-global-corporations/, 01 2021, (Accessed on 28/08/2021). [Cited on page 3.]

[10] M. Bromiley, "Threat intelligence: What it is, and how to use it effectively," *SANS Institute InfoSec Reading Room*, vol. 15, p. 172, 2016. [Cited on page 3.]

[11] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Computers & security*, vol. 72, pp. 212–233, 2018. [Cited on page 4.]

[12] "Low, Medium and High Interaction Honeypot Security," https://www.guardicore.com/blog/high-interaction-honeypot-versus-low-interaction-honeypot-comparison/, (Accessed on 04/05/2021). [Cited on pages 7 and 8.]

[13] O. Catakoglu, M. Balduzzi, and D. Balzarotti, "Automatic extraction of indicators of compromise for web applications," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 333–343. [Cited on pages 8 and 28.]

[14] "What is an indicator of compromise (IoC)? — Kaspersky IT Encyclopedia," https://encyclopedia.kaspersky.com/glossary/indicator-of-compromise-ioc/, (Accessed on 08/08/2021). [Cited on page 8.]

[15] "What are Indicators of Compromise (IOCs)," https://securityscorecard.com/blog/what-are-indicators-of-a-compromise, (Accessed on 08/08/2021). [Cited on page 8.]

[16] "Indicators of Compromise (IOC) Security Explained," https://www.crowdstrike.com/cybersecurity-101/indicators-of-compromise/, (Accessed on 27/06/2021). [Cited on page 8.]

[17] L. Dandurand and O. S. Serrano, "Towards improved cyber security information sharing," in *2013 5th International Conference on Cyber Conflict (CYCON 2013)*. IEEE, 2013, pp. 1–16. [Cited on pages 8 and 15.]

[18] "Automated Defense - Using Threat Intelligence to Augment," https://www.sans.org/white-papers/35692/, (Accessed on 27/06/2021). [Cited on page 9.]

[19] C. Sauerwein, C. Sillaber, A. Mussmann, and R. Breu, "Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives," 2017. [Cited on pages 9 and 16.]

[20] "What is a Cyber Attack?" https://www.upguard.com/blog/cyber-attack, (Accessed on 22/09/2021). [Cited on page 9.]

[21] C. NIST, "Official Common Platform Enumeration (CPE) Dictionary," 2021. [Cited on page 10.]

[22] "NVD - CVE-2014-0160," https://nvd.nist.gov/vuln/detail/CVE-2014-0160, (Accessed on 22/09/2021). [Cited on page 10.]

[23] "WannaCry Ransomware: A Detailed Analysis of the Attack," https://techspective.net/2017/09/26/wannacry-ransomware-detailed-analysis-attack/, (Accessed on 30/06/2021). [Cited on page 10.]

[24] "NVD - CVE-2017-0144," https://nvd.nist.gov/vuln/detail/CVE-2017-0144, (Accessed on 22/09/2021). [Cited on page 10.]

[25] L. A. B. Sanguino and R. Uetz, "Software vulnerability analysis using CPE and CVE," *arXiv preprint arXiv:1705.05347*, 2017. [Cited on page 10.]

[26] "OVAL - Open Vulnerability and Assessment Language," https://oval.mitre.org/, (Accessed on 16/06/2021). [Cited on page 11.]

[27] "Security Content Automation Protocol," https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/xccdf, (Accessed on 16/06/2021). [Cited on page 11.]

[28] "About CPE," https://cpe.mitre.org/about/, (Accessed on 16/06/2021). [Cited on page 11.]

[29] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013. [Cited on page 11.]

[30] E. E. Han and T. N. Phyu, "Classification of SQL injection, XSS and Path Traversal for Web Application Attack Detection." Fourteenth International Conference On Computer Applications (ICCA 2016), 2016. [Cited on page 12.]

[31] T. Wilhelm and J. Andress, "Chapter 8 - Use of Timing to Enter an Area," in *Ninja Hacking*, T. Wilhelm and J. Andress, Eds. Boston: Syngress, 2011, pp. 119–134. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597495882000081 [Cited on page 13.]

[32] C. Lai, A. R. Chavez, C. B. Jones, N. Jacobs, S. Hossain-McKenzie, J. B. Johnson, and A. Summers, "Review of intrusion detection methods and tools for distributed energy resources." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2021. [Cited on page 13.]

[33] "Top 5 open-source HIDS systems," https://logz.io/blog/open-source-hids/, (Accessed on 28/05/2021).

[34] "8 Best HIDS Tools," https://www.dnsstuff.com/host-based-intrusion-detection-systems, (Accessed on 28/05/2021). [Cited on page 13.]

[35] C. L. Smith, "AIDE - Advanced Intrusion Detection Environment," Pacific Northwest National Lab.(PNNL), Richland, WA (United States), Tech. Rep., 2013. [Cited on page 13.]

[36] "AIDE - ArchWiki," https://wiki.archlinux.org/title/AIDE, (Accessed on 30/04/2021). [Cited on page 14.]

[37] G. fail2ban, "fail2ban: Daemon to ban hosts that cause multiple authentication errors," https://github.com/fail2ban/fail2ban, (Accessed on 30/04/2021). [Cited on page 14.]

[38] D. Teixeira, L. Assunção, T. Pereira, S. Malta, and P. Pinto, "Ossec ids extension to improve log analysis and override false positive or negative detections," *Journal of Sensor and Actuator Networks*, vol. 8, no. 3, p. 46, 2019. [Cited on page 14.]

[39] "Fail2ban Setup," https://www.cyberpunk.rs/fail2ban-setup-intrusion-prevention-framework, (Accessed on 30/04/2021). [Cited on page 14.]

[40] R. Bray, D. Cid, and A. Hay, *OSSEC host-based intrusion detection guide*. Syngress, 2008. [Cited on page 14.]

[41] "Open Source - Quadrant Information Security," https://quadrantsec.com/sagan_log_analysis_engine/, (Accessed on 30/04/2021). [Cited on page 14.]

[42] R. Wichmann, "The Samhain HIDS," *fact sheet*, 2011. [Cited on page 14.]

[43] G. H. Kim and E. H. Spafford, "The design and implementation of tripwire: A file system integrity checker," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, 1994, pp. 18–29. [Cited on page 15.]

[44] "What Is Endpoint Detection and Response?" https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint/what-is-endpoint-detection-and-response.html, (Accessed on 03/05/2021). [Cited on page 16.]

[45] "Top Threat Intelligence Platforms [2021]," https://www.esecurityplanet.com/products/threat-intelligence-platforms/, (Accessed on 26/06/2021). [Cited on page 16.]

[46] "IBM X-Force Exchange: FAQ," https://exchange.xforce.ibmcloud.com/faq, (Accessed on 03/05/2021). [Cited on page 16.]

[47] "ThreatStream - Threat Intelligence Platform," https://www.anomali.com/products/threatstream, (Accessed on 03/05/2021). [Cited on page 16.]

[48] "ThreatConnect - What is Threat Intelligence," https://threatconnect.com/solution/threat-intelligence/, (Accessed on 03/05/2021). [Cited on page 16.]

[49] A. W. Mir and R. K. Ramachandran, "Implementation of Security Orchestration, Automation and Response (SOAR) in Smart Grid-Based SCADA Systems," in *Sixth International Conference on Intelligent Computing and Applications*. Springer, 2021, pp. 157–169. [Cited on page 16.]

[50] "Falcon X - Cyber Threat Intelligence & Automation," https://www.crowdstrike.com/endpoint-security-products/falcon-x-threat-intelligence/, (Accessed on 03/05/2021). [Cited on page 17.]

[51] E. Fernández, "ThreatQ, plataforma de inteligencia sobre amenazas abierta para acelerar las operaciones de ciberseguridad," *Revista SIC: ciberseguridad, seguridad de la información y privacidad*, vol. 30, no. 144, pp. 166–166, 2021. [Cited on page 17.]

[52] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 2016, pp. 49–56. [Cited on pages 17 and 31.]

[53] "TheHive Project," http://thehive-project.org/, (Accessed on 26/07/2021). [Cited on page 17.]

[54] Q. D. La, T. Q. Quek, J. Lee, S. Jin, and H. Zhu, "Deceptive attack and defense game in honeypot-enabled networks for the internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1025–1035, 2016. [Cited on page 18.]

[55] N. Provos, "Honeyd - A virtual honeypot daemon," in *10th DFN-CERT Workshop, Hamburg, Germany*, vol. 2, 2003, p. 4. [Cited on pages 18 and 20.]

[56] N. Provos *et al.*, "A Virtual Honeypot Framework," in *USENIX Security Symposium*, vol. 173, 01 2004, pp. 1–14. [Cited on page 18.]

[57] N. F. Khan and M. M. Mohan, "Honey pot as a service in cloud," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 20, pp. 2883–2888, 2018. [Cited on pages 18 and 20.]

[58] J. Jafarian and A. Niakanlahiji, "Delivering Honeypots as a Service," 01 2020. [Cited on pages 18 and 20.]

[59] H. Artail, H. Safa, M. Sraj, I. Kuwatly, and Z. Al-Masri, "A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks, journal = Computers & Security," vol. 25, no. 4, pp. 274–288, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404806000587 [Cited on page 18.]

[60] A. Kostopoulos, I. P. Chochliouros, C. Patsakis, M. Anastasiadis, and A. Guarino, "Protocol Deployment for Employing Honeypot-as-a-Service," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2020, pp. 105–115. [Cited on pages 19 and 20.]

[61] "CVE - Frequently Asked Questions," https://cve.mitre.org/about/faqs.html#who_owns_cve, (Accessed on 16/06/2021). [Cited on page 21.]

[62] "XML::Parser - A perl module for parsing XML documents," https://metacpan.org/pod/XML::Parser, (Accessed on 20/09/2021). [Cited on page 24.]

[63] M. Guo and J. A. Wang, "An ontology-based approach to model common vulnerabilities and exposures in information security," in *ASEE Southest Section Conference*, 2009. [Cited on page 28.]

[64] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," *Eai Endorsed Transactions on Security and Safety*, vol. 3, no. 9, p. e2, 2016. [Cited on page 29.]

[65] "Virtualization-Based Sandboxes are Vulnerable to Advanced Malware," https://www.lastline.com/blog/virtualization-based-sandboxes/, (Accessed on 28/05/2021). [Cited on page 30.]

[66] G. PyMISP, "PyMISP - Python Library to access MISP," *PyMISP (accessed on 20 February 2021)*. [Online]. Available: https://github.com/MISP/PyMISP [Cited on page 31.]

[67] "Fail2ban Manual 0.8," https://www.fail2ban.org/wiki/index.php/MANUAL_0_8#Jails, (Accessed on 30/04/2021). [Cited on page 34.]

[68] Canonical, "Multipass orchestrates virtual Ubuntu instances," available at: https://github.com/canonical/multipass (Accessed 20 July 2021), 2015. [Cited on page 34.]

[69] F. Bellard, "QEMU, a fast and portable dynamic translator." in *USENIX annual technical conference, FREENIX Track*, vol. 41. Califor-nia, USA, 2005, p. 46. [Cited on page 35.]

[70] "LXD - Introduction," https://linuxcontainers.org/lxd/introduction/, (Accessed on 26/07/2021). [Cited on page 35.]

[71] S. Senthil Kumaran, *Practical LXC and LXD: linux containers for virtualization and orchestration*. Springer, 2017. [Cited on page 35.]

[72] Y. Hu, A. Nanda, and Q. Yang, "Measurement, analysis and performance improvement of the Apache web server," in *1999 IEEE International Performance, Computing and Communications Conference (Cat. No. 99CH36305)*. IEEE, 1999, pp. 261–267. [Cited on page 37.]

[73] A. Brandão, J. S. Resende, and R. Martins, "Hardening cryptographic operations through the use of secure enclaves," *Computers & Security*, vol. 108, p. 102327, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821001516 [Cited on page 37.]

[74] A. Tirumala, "Iperf: The TCP/UDP bandwidth measurement tool," *http://dast.nlanr.net/Projects/Iperf/*, 1999. [Cited on page 39.]