FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Detection of Cyberattacks on a Multi-tenant Service

Ângelo Daniel Pereira Mendes Moura

Mestrado em Engenharia Informática e Computação

Supervisor: António Miguel Pontes Pimenta Monteiro

March 11, 2022

Detection of Cyberattacks on a Multi-tenant Service

Ângelo Daniel Pereira Mendes Moura

Mestrado em Engenharia Informática e Computação

March 11, 2022

Abstract

In today's technological world, cyberattacks and cybercrime are ever more prevalent. Because the world is becoming increasingly dependent on informatic systems, protecting these systems is becoming indispensable. As such, the purpose of this thesis is to research ways to detect cyberattacks, explore solutions that are capable of preventing them and validate their effectiveness when implemented into a multi-tenant service.

To accomplish the above, extensive research is made into the concept of monitoring and logging tools, as these tools are often cited as effective countermeasures against cyberattacks. After this research has been done, one such tool is implemented onto a multi-tenant service serving as a case study. After the implementation itself, multiple tests designed to validate the tool's effectiveness were performed: simulated cyberattacks, tests regarding information-gathering capabilities, and performance impact tests.

When faced with the tests performed, the monitoring and logging tool selected performed admirably, correctly detecting and identifying the attacks launched against the platform. It also demonstrated satisfactory levels of information gathering, a necessity when avoiding repeated attacks and evaluating and repairing damages after one. It could also respond automatically to an attack, significantly decreasing the response time. Furthermore, it demonstrated the qualifications to be adjusted to the case-study needs without affecting its performance in a significant way.

The positive results from the tests performed demonstrated that monitoring and logging tools are indeed a valuable line of defence against the ever more common cyberattacks and should be, whenever possible, integrated into any informatic system. However, as most things are in the realm of cyber security, these are merely one line of defence and should not be thought of as the end-all solution. Because as technology evolves, so do the tactics employed by attackers, and it is only with constant vigilance and awareness that systems can be kept safe.

Keywords: Cyberattacks, Monitoring, Logging, Vulnerabilities, Security

Resumo

No mundo tecnológico atual, os ciberataques e o cibercrime são cada vez mais prevalentes. Como o mundo está a tornar-se cada vez mais dependente dos sistemas informáticos, a proteção destes sistemas está a tornar-se indispensável. Como tal, o objetivo desta tese é investigar formas de detetar os ciberataques, explorar soluções capazes de os prevenir e validar a sua eficácia quando implementados num serviço multi-tenant.

Para tal, é feita uma extensa investigação sobre o conceito de ferramentas de monitorização e registo, uma vez que estas ferramentas são frequentemente citadas como contramedidas eficazes contra os ciberataques. Após esta investigação ter sido feita, uma dessas ferramentas é implementada num serviço multi-tenant, servindo este como um caso de estudo. Após a implementação, foram realizados múltiplos testes concebidos para validar a eficácia da ferramenta: ciberataques simulados, testes relativos a capacidades de recolha de informação, e testes de impacto de desempenho.

Quando confrontada com os testes realizados, a ferramenta de monitorização e registo selecionada comportou-se admiravelmente, detetando e identificando corretamente o ataque lançado contra a plataforma. Demonstrou também níveis satisfatórios de recolha de informação, uma necessidade para evitar ataques repetidos e para avaliar e reparar danos após um. Também demonstrou a capacidade de responder automaticamente a um ataque, diminuindo significativamente o tempo de resposta. Além disso, demonstrou ser ajustável às necessidades do caso de estudo e não afetou o desempenho deste de forma significativa.

Os resultados positivos dos testes realizados demonstram que as ferramentas de monitorização e registo são de facto uma valiosa linha de defesa contra os ciberataques, que são cada vez mais comuns, e devem ser, sempre que possível, integradas em qualquer sistema informático. No entanto, como a maioria das coisas no domínio da cyber segurança, estas ferramentas são apenas uma linha de defesa e não devem ser vistas como uma solução universal. Porque à medida que a tecnologia evolui, o mesmo acontece com as táticas utilizadas por atacantes, e é apenas com vigilância e estudo constante que os sistemas podem ser mantidos em segurança.

Keywords: Ataques Cibernético, Monitorização, Registo, Vulnerabilidades, Segurança

Acknowledgements

In creating this project, I have received a tremendous amount of support, and for it, I would like to extend my thanks and appreciation to the following people.

I want to thank my supervisor, professor Miguel Pimenta Monteiro, who allowed me to take this massive opportunity as my final thesis project.

I want to express immense gratitude to the people of Altice Labs for their support and willingness to help throughout every stage of this project. Especially Jose Melo for his continuous advice, guidance and general grammar corrections. And Paulo Sousa for his guidance, accommodation, and excellent team leadership.

I also want to express as much gratitude as possible to my family and friends. To my mom Rosa Pereira, for too much to put into words, so let us keep it at everything. To my brother Andre Moura for always supporting me no matter my choices. To my dad Ângelo Moura, for shaping the way I am today. I couldn't have done any of this without them. To my closest friends Daniel, Ricardo, Tiago and Simão for helping me relax and keeping me sane and humble. May we meet all again at the same place as always.

Ângelo Daniel Pereira Mendes Moura

"If you think you know-it-all about cybersecurity,
this discipline was probably ill-explained to you."

Stephane Nappo

Contents

List of Figures

List of Tables

Abbreviations

ACL	Access Control List
AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
CDN	Content Delivery Network
CIA	Confidentiality, Integrity and Availability
CI/CD	Continuous Integration and Continuous Deployment
CMS	Content Management System
CSP	Content Security Policy
DHCP	Dynamic Host Configuration Protocol
DLP	Data Loss Protection
DNS	Domain Name System
DoH	DNS-over-HTTPS
DOM	Document Object Model
DoS Attack	Denial-of-Service Attack
DDoS Attack	Distributed Denial-of-Service Attack
EDR	Endpoint Detection and Response
EPS	Events Per Second
FTP	File Transfer Protocol
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection Systems
IOC	Indicators of Compromise
IoT	Internet of Things
IP Address	Internet Protocol Address
IPS	Intrusion Prevention System
LotL	Living of the Land
MITM	Man-in-the-middle
NIDS	Network-Based Intrusion Detection System
NIPS	Network-Based Intrusion Prevention System
NIST	National Institute of Standards and Technology
PII	Personally Identifiable Information
OWASP	Open Web Application Security Project
SIEM	Security Information and Event Management
SOC	Security Operations Center
SSH	Secure Shell
SSRF	Server Side Request Forgery

SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locators
VPN	Virtual Private Network
XSS	Cross-Site Scripting

Chapter 1

Introduction

Cyberattacks are ever more common, and with today's ever depend world on technology, the consequences of these attacks can be disastrous. Should a cyberattack be successful on a healthcare system, a banking system or a news system, it could mean the loss of irreplaceable data and even goods or money. As such, the field of cyber security needs to be constantly evolving and improving. As the ways of attacking increase, so must the computer systems defence rise to match them.

The main goal of this dissertation is to explore ways to detect cyberattacks and find ways to prevent them and the damages they may cause to people. It aims to investigate the space and validate the effectiveness of the tools and techniques available and contribute to the ever raging war against cyberattacks and cybercrime.

This paper focuses on a set of tools collectively referred to as monitoring and logging tools. These tools are cited regularly as an effective way to detect and safeguard systems against cyberattacks. But in part due to their complexity and seemly specialised usages are often underused and not fully understood outside of the space of cybersecurity professionals.

This introductory section presents a breakdown and further specification of the main objective for this project, the steps taken in the pursuit of these goals, and a description of this document structure.

1.1 Objectives

As stated above, the main goal of this dissertation is to explore ways and mechanisms that detect and protect systems against cyberattacks. In pursuit of this goal, this dissertation explores the knowledge contained in the cybersecurity field while focusing on the different cyberattacks and what kinds of vulnerabilities are usually held within a computer system.

From this research, the concept of monitoring and logging tools was discovered being often referred to as countermeasures against cyberattacks and as a best practice inclusion when developing a system. When implemented onto a system, these tools are cited as capable of detecting when

a cybersecurity attack is targeting the system effectively. They also are stated to have the capabilities to support systems to prevent cyberattacks and provide invaluable information to perform system repairs and damage reports should one be successful.

Due to these claims, this project aimed to confirm the capabilities of logging and monitoring tools. And to do this, it implemented one such tool onto a multi-tenant service, serving as a case study. Then it proceeded to perform several tests designed to emulate real-world scenarios and use cases while recording the responses created by the implemented tool. These tests consist of different types of cyberattacks, information gathering and monitoring showcases, and performance impact checks.

In the end, the results were analysed to determine if monitoring and logging tools are indeed capable of detecting cyberattacks and if they can support systems in preventing and recovering from them.

1.2 Related Works

In this section, this dissertation will take a moment to highlight similar or related works to it. These works include research into monitoring and logging tools themselves or pieces that have used them to explore other avenues.

The work made by Vitsunee Teeraratchakarn and Yachai Limpiyakorn titled "Exploring Network Vulnerabilities for Corporate Security Operations" (Teeraratchakarn and Limpiyakorn, 2020). In it, they explored the capabilities of an open-source monitoring and logging product called Elastic Stack (Elastic NV, 2022) to discern the most common user names used by attackers during a brute-force attack. The ultimate goal was to advise organisations to regulate their internal identities better as an added safety measure (Teeraratchakarn and Limpiyakorn, 2020). Although their conclusion shares similarities with this dissertation's ultimate conclusion, their exploration of monitoring and logging tools was limited to brute-force attacks due to their focus on identities and user names.

The work made by Ferdy Mulyadi, Leela Anman, Ridnarong Promya and Chalermpol Charnsripinyo titled "Implementing Dockerized Elastic Stack for Security Information and Event Management" (Mulyadi et al., 2020). In it, they present an implementation of Elastic Stack (Elastic NV, 2022) utilising Docker (Docker Inc., 2022) technology. Although similar to this dissertation, this work focuses more on the impact on system performance such an installation has.

The work made by Wen-Lin Cheng, Ting-Che Chuang, Chien-Wen Yang, Yueh-Hsien Lin, Min Liu and Chuan Yin titled "An Integrated Security Monitoring System for Digital Service Network Devices" (Cheng et al., 2017). In it, they provide telecom operators with a solution to establish device security monitoring and centralized statistical analysis, a monitoring and logging tool (Cheng et al., 2017). Although similar to this dissertation, this work focuses on providing telecom operators with an adequate monitoring and logging solution rather than studying its capabilities.

The work made by various authors titled "On Vulnerability and Security Log analysis: A Systematic Literature Review on Recent Trends" (Svacina et al., 2020). In it, they compiled a summary of strategies to perform log analysis and discuss issues they have identified with the logging mechanism of software systems (Svacina et al., 2020).

The work made by Yong Sun and Haiwen Wang titled "Intelligent Computer Security Monitoring Information Network Analysis" (Wang et al., 2019). In it, they developed a system capable of effectively responding to network security attacks (Wang et al., 2019).

The work made by Rui Miguel Almeida Oliveira titled "Analysis of Intrusion Detection Log Data on a Scalable Environment" (Oliveira, 2020). They developed an application capable of capturing large amounts of network traffic logs, which then used to create a ground truth. After which, it uses machine learning techniques to classify traffic and detect intrusion attempts, with a general focus on detecting DDoS attacks (Oliveira, 2020).

The work made by Muhammad I.H. Sukmana, Kennedy A. Torkura, Feng Cheng, Christoph Meinel, and Hendrik Graupne titled "Unified Logging System for Monitoring Multiple Cloud Storage Providers in Cloud Storage Broke" (Sukmana et al., 2018). In it, they developed a logging system capable of unifying the different formatted logs of 3 cloud storage providers (Sukmana et al., 2018).

The work made by Yun Wang and Qianhuizhi Zheng titled "A Logging Overhead Optimization Method Based on Anomaly Detection Model" (Wang and Zheng, 2020). In it, they propose an optimisation method for the logging overhead problem by combining a new log parsing method with a deep learning-base anomaly detection method (Wang and Zheng, 2020). Tangential related to this dissertation demonstrates the work to ensure efficiency when processing logs.

1.3 Document Structure

Besides the introduction, this document has five more chapters.

Chapter 2 serves as a general introduction to the cyber security field. There is a broad definition of cyber security, a list of the most common vulnerabilities plaguing computer systems, and the most common types of cyberattacks and their recommended countermeasures.

Chapter 3 dives deep into the concept of monitoring and logging tools as a way to counteract cyberattacks. There the purpose and goals of these tools are highlighted. It also contains general guidelines for the correct implementation, usage and maintenance of these tools. Additionally, it includes a non-intensive list o currently available tools in the market.

Chapter 4 introduces the multi-tenant service used as a case study in this thesis, the DataPlaxe platform. It goes over its architecture and components and its most prevalent security needs.

Chapter 5 describes the approach taken for the implementation of a monitoring and logging tool on the DataPlaxe platform. It describes the selection process and highlights why the tool chosen was chosen, namely Wazuh. Then it describes the tests made to oversee the effectiveness of Wazuh, serving as a representative of the capabilities of monitoring and logging tools in general. These tests consist of simulated cyberattacks, information gathering and system monitoring trials,

and general performance and utility tests. These last ones are more directed to the specific needs of the DataPlaxe platform.

Chapter 6 serves as the conclusion of this thesis. The results obtained from the previous chapter are considered, and an evaluation of the effectiveness of monitoring and logging tools as a cyber security asset is made. It also discusses possible avenues to fool when thinking of future works within this space.

Chapter 2

Cyber Security

2.1 Overview

The field of cyber security is large and expansive. In today's digital world, as more of everybody's life is dependent on digital systems, the safety of these systems is even more relevant and essential. It is easy to understand security instinctively, but defining it by words or principles can be tricky. As such, this chapter introduces this field by offering a simple explanation of the most fundamental concepts at the core of this massive and ever-expanding field of study.

At the core of cyber security, there are two primordial questions: what makes a system secure, and how can we guarantee the security of a system. The answers to these two questions are constantly being debated, refined, and adapted to the ever-evolving world of technology.

Before it is possible to answer the first question, it is essential to define security properly. In the digital world, security is usually considered the result of the intersection of Confidentiality, Integrity and Availability (commonly abbreviated by the sigla CIA) (Correia, 2020). Confidentiality can be defined as "the property that information is not made available or disclosed to unauthorised individuals, entities, or processes" (Beckers, 2015). Integrity is "guarding against improper information modification or destruction" (Nieles et al., 2017). And Availability can be defined as "the property of being accessible and usable upon demand by an authorised entity" (Beckers, 2015).

From this interpretation of security, a secured system can be interpreted as a system that protects its information "from unauthorised access, use, disclosure, disruption, modification, or destruction in order to ensure confidentiality, integrity, and availability" (Nieles et al., 2017).

In the simplest terms, a secured system can be interpreted as giving complete control over its information to only authorised individuals. Although it is rare for any individual to have complete control over the entire system's data, it is the norm for any system's user to have ownership over parts of it; personal information is the typical example. (Intersoft Consulting, 2021)

As to how it is possible to guarantee the security of a system, that is a rather complex problem to solve. Providers and developers may use many techniques and tools to achieve the inherent goal present in the question.

Figure 2.1: Security as the result of the intersection of CIA (Correia, 2020)

There is the concept of Secure by Design, where a system is designed with security as aforethought. This simple but powerful approach to system development results in security tactics being enforced by the system's architecture. (Santos et al., 2017)

Developers can mitigate the risks and consequences of many vulnerabilities by upholding the Principle of Least Privilege. This principle enforces every program and user to have only the minimum necessary permissions to perform their tasks and only while performing them. (Saltzer and Schroeder, 1975)

And there is the concept of Defence in Depth, which consists of adding multiple layers of security, creating redundancy within the system, and having various checks in place for the system activity flow. (Groat et al., 2012)

And developers can also develop or implement auxiliary subsystems that complement the system's security. These can be an antivirus that performs a system analysis in search of compromising software, firewalls that enforce policy rules on information traffic, or the focus of this dissertation, logging and monitoring systems that monitor the system's internal workings in search of abnormal activity.

The concept of logging and monitoring systems is presented in a later chapter. For now, this study will go over the common threats to a system's security. This way, it will become more apparent how monitoring and logging systems can mitigate and solve a significant part of these threats.

2.2 Vulnerabilities of Computer Systems

As stated before, one of the core aspects of cyber security is guaranteeing the security of a system. In truth, no system is genuinely unbreakable. New forms of attack are constantly being invented, and these are then followed by a way to safeguard systems against them. Developers and attackers are always trying to get the upper hand on one another, and it is the job of developers to make sure their platforms are as vulnerability free as possible. As this is the only way to maintain a system secure, it is essential to clarify what a vulnerability is.

The National Institute of Standards and National Institute of Standards and Technology (NIST) defines a vulnerability as "a weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source" (Blank and Gallagher, 2012). The origin of most of these can usually be traced to a fault in some security control. They can surge due to insufficient testing, lack of validation of software sources, or simply by uninformed design. However, some vulnerabilities appear over time as the context of the system changes around it. These changes can happen due to the advent of new technologies, new attack techniques, software changes and updates, or even shifts in the companies' focus and environment. Over time, all existing security controls may become inadequate. Because of this tendency of security controls to become outdated and insufficient, systems must undergo regular security risk assessments and have a continuous monitoring program to help detect abnormal activity. (Blank and Gallagher, 2012)

Failing to fix a vulnerability can prove disastrous to any system, as it can be exploited to obtain sensitive information or compromise the system's internal workings. Because of this enormous risk, companies and developers have created and maintained lists with descriptions of the most common and worst vulnerabilities that can plague a system. These lists aim to educate their peers on what to look for in the developing phase or when managing a system.

One of the most recognised is the one maintained by the Open Web Application Security Project (OWASP) Foundation. Which every year compiles the OWASP Top 10, a "document outlining the 10 most critical security concerns for web application security" (OWASP Foundation, 2021a). The 2021's list is summarised below (OWASP Foundation, 2021b):

1. Broken Access Control

   A system that contains a broken or lacking Access Control cannot enforce its user to act within their permissions. Meaning that a standard user may view, alter, or even delete privileged information. (OWASP Foundation, 2021b)

2. Cryptographic Failures

   Cryptographic Failures occur whenever the required protections for data in transit and at rest are not met. (OWASP Foundation, 2021b)

3. Injection

   Code injection is usually the result of poor filtering of the input data provided by the users of a system. Because this data is not appropriately treated, an attacker can upload foreign code to the system. Allowing him to access private information, take over part or the entirety of the system, etc. (OWASP Foundation, 2021b)

4. Insecure Design

   Insecure design covers many weaknesses that originated from the lack of security awareness during the design phase. An error in the system's implementation may cause vulnerabilities, but a perfect implementation of a faulty design is sure to have security problems. This particular concern serves to provide ways to improve the design process of a system: promote the usage of secure design patterns and libraries, divide the system depending on exposure and protection needs, use unit and integration testing to validate critical flows of information, etc. (OWASP Foundation, 2021b)

5. Security Misconfiguration

   Security Misconfiguration covers many similar situations where system settings were not adequately cared for or set to secure values. Some examples are: "unnecessary features are enabled or installed (e.g., unnecessary ports, services, pages, accounts, or privileges)" (OWASP Foundation, 2021b), "Default accounts and their passwords are still enabled and unchanged" (OWASP Foundation, 2021b), upgraded systems have their latest security features disabled or not configured, etc. (OWASP Foundation, 2021b)

6. Vulnerable and Outdated Components

   The title of this point is self-explanatory. It warns developers to be aware of the version of the components used, directly or indirectly, on the system and perform regular maintenance to keep them as up to date as possible. (OWASP Foundation, 2021b)

7. Identification and Authentication Failures

   "Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks." (OWASP Foundation, 2021b) This point covers any possible vulnerability that allows a user to access the system without being correctly authenticated. Some examples are allowing automated attacks such as brute force or credential stuffing, allowing weak passwords such as "Password1" or "admin", exposing the session identifier in the URL, or a missing or ineffective multi-factor authentication. (OWASP Foundation, 2021b)

8. Software and Data Integrity Failures

   Software and Data Integrity Failures focuses on vulnerabilities that originate due to the use of unvalidated code or critical data without verifying its integrity. Unvalidated code may come from multiple different sources: the use of untrusted repositories or content delivery

networks (CDNs), malicious code introduced by unauthorised access to an insecure CI/CD pipeline and from applications auto-update functionalities. This point highlights techniques and procedures used to ensure the integrity of the used code and data. (OWASP Foundation, 2021b)

9. Security Logging and Monitoring Failures

   Security Logging and Monitoring Failures focus on ensuring the correct usage and implementation of monitoring and logging tools. It provides developers with warnings on not storing logs locally, having proper alert definitions and thresholds, ensuring real-time detection, escalation and alerting, and many more. The correct usage of monitoring and logging tools is crucial as, without them, active breaches cannot be detected. (OWASP Foundation, 2021b)

10. Server Side Request Forgery (SSRF)

    Server Side Request Forgery can happen when a web application uses a resource provided by an end-user without proper validation of the URL. An attacker can exploit this fact and coerce the application to send a crafted request to an unexpected destination bypassing firewalls, VPNs, or another type of network access control list (ACL). (OWASP Foundation, 2021b)

2.3 Cyberattacks

Now that the concept of vulnerabilities has been explained, it is also essential to understand the other side of Cyber Security: the attacking side. Cyberattacks are a constant threat to any system, and as attackers innovate and respond to technological evolutions, they become more varied. Because of this variety, it becomes necessary to categorise them to present them more efficiently. There are three types of cyberattacks: Reconnaissance Attacks, Network Intrusion Attacks and Denial of Service (DoS) Attacks. (Duarte, 2008)

In this section, this dissertation will go over the different types of attacks that can target a system. It will provide a short explanation for each one and highlight the recommended defences for each.

2.3.1 Reconnaissance Attacks

Reconnaissance Attacks are designed to gather information about the system instead of the usual goal of information contained within the system. They are usually a precursor to a more significant directed attack, as they allow an attacker to understand its target system and possibly find vulnerabilities that can be exploited. These types of attacks usually take the form of sweeping scans that look for open connection points, or ports as they are typically called, and try to discover the connection protocols used by the system. (Duarte, 2008)

A non-extensive list of the most common Reconnaissance Attacks a system might be targeted by can be seen next.

- Domain Name System (DNS) Request (Duarte, 2008)

  Description: Although not necessarily an attack, by making a DNS request, an attacker can gather a lot of crucial information about the target system. A DNS entry holds information such as the IP address, other hostnames used by the system and the mail exchanger. (Liu Cricket and Albitz Paul, 2006)

  Solution: The DNS has been an integral part of the Internet since 1985, and it wasn't designed with modern security concerns. Because of this, attackers have exploited it many times, and there isn't any definitive way to improve it. There have been some attempts to mitigate its outdated design. For example, in 2018, Google and Mozilla have started using DNS-over-HTTPS (DoH), a technology that uses the secure HTTPS protocol for DNS data transmission (Kaspersky, 2021). But the best way to avoid giving out too much information via the DNS system is not to use it. If the system uses a local area network or a site network, it may not need the DNS system and instead use an alternative technology for the same effect. (Liu Cricket and Albitz Paul, 2006)

- Ping sweeps or ICMP sweeps (Duarte, 2008)

  Description Although not necessarily an attack, an attacker might use what it's called a ping sweep to determine the live hosts in a range of IP addresses. (Duarte, 2008)

  Solution: There is no good way to avoid ping sweeps, it's possible to block ICMP packets using firewalls, but this compromises network capabilities (Duarte, 2008). Although most monitoring tools or intrusion prevention systems (IPS) can detect one is underway.

- Port Scanning (Duarte, 2008)

  Description: Port scanning can come in two different forms: horizontal, where the same port is scanned on other hosts, or vertical, where multiple ports of a single host are scanned. The goal of both is to check for vulnerable ports, as the scan can identify the port's status (open, closed, firewall-protected), what service is running on the port and the device type and operating system. (Duarte, 2008)

  Solution: Most monitoring tools, intrusion prevention systems (IPS) or firewalls can detect port scanning. And, upon detection, they can temporarily open all ports to feed false information to the attackers. Although the best solution is to only open the necessary ports for the system's functionalities and keep these as secure as possible behind a firewall. (Kaspersky, 2021)

2.3.2 Network Intrusion Attacks

Network Intrusion Attacks are what most people are referring to when they discuss cyberattacks, as these are by far the most damaging to any system. When successfully performed, these types

of attacks can expose sensible and private information contained in the system or, in some rarer cases, they can even allow complete system takeover. (Duarte, 2008)

As the name suggests, these attacks compromise the network of information of the system by performing unauthorised activities. This disruption can be done in several ways, and the exact results of the attack can also vary somewhat. In this section, this dissertation will go over the most common types of Network Intrusion Attacks, explain what the consequences of these attacks might be if left unchecked and highlight the most common defence mechanisms used against them. (Awake Security, 2021; Duarte, 2008)

- Living Off the Land (LotL) (Awake Security, 2021)

  Description: Living off the land (LotL) describes an attack where the attacker takes advantage of legitimate software present in the system to gather information. These legitimate software are usually operating system utilities, business productivity software and scripting languages; all commonly used and legitimate software widely used in many systems. Since the attackers use legitimate and already present software, this attack does not leave any physical trace, making detection challenging. (Awake Security, 2021; Kaspersky, 2021)

  Solution: Because this attack does not leave malicious files on the system, techniques such as comparing file signatures do not work. Additionally, some tools, such as the ones connected to the operating system, can be in the allowlist and therefore trusted, making detection even harder. Indeed, the best and possibly only way of detecting these types of attacks is to adopt solutions based on behavioural analysis. These technologies can see the abnormal program and user activity, indicating an ongoing intrusion attack. (Kaspersky, 2021)

- Man-in-the-middle (MITM) (Duarte, 2008)

  Description: A Man-in-the-middle attack involves an attacker intercepting data flow between two entities without stopping it. Making all information flow through the attacker, but both communicating parties are none the wiser as they do not perceive any change. This interception can be accomplished through multiple techniques at very different communication points. The most common is impersonating a trusted entity along the communication chain, entities like the Address Resolution Protocol (ARP) or a DNS, which is a technique called spoofing. (Duarte, 2008)

  Solution: There is no good way to detect a MITM attack, making prevention the best defence against these attacks. Ensuring a secure connection by having robust encryption mechanisms on wireless access points and having strong router credentials can prevent an attacker from brute-forcing its entry onto the network. Using a VPN or forcing the usage of the HTTPS protocol ensures that the data will be unreadable to the attacker. And having a public and private key authentication system is the best way of guaranteeing the authenticity of the entities involved in the communication chain. (Rapid7, 2021)

- Stack Smashing (Awake Security, 2021)

Description: Stack Smashing or Buffer Overwriting is a cyberattack that aims to execute foreign and malicious code. This execution is done by overwriting certain sections of the device's memory with commands and instructions designed by the attacker to gain control over a vulnerable application. Thus, allowing a malicious script to be run in the system with the permissions of the hijacked application. (Awake Security, 2021; Kaspersky, 2021)

Solution: The best way to prevent a stack smashing attack is to endow the system with a robust boundary-checking logic, preventing vulnerable applications from rewriting crucial memory sections. The system should also have as few vulnerable applications as possible, preferably none. And mechanisms that detect executable code or malicious strings before they are written to memory should also be implemented. (Awake Security, 2021)

- Malware Attacks (Rapid7, 2021)

  Description: Malware is a general term that encompasses many types of malicious software such as viruses, worms and trojan horses. The specific attacks these types of software can perform are numerous. They may encrypt a machine making it unusable until a ransom is paid, making the malware ransomware. They can watch, log, and transmit all activity performed on the device, making the malware spyware. Or they can even take complete control over the machine. (Rapid7, 2021)

  Solution: When it comes to preventing a malware attack, one should focus on eliminating the main entry points used by these types of software. There are two main ways malware can find its way onto a system: through user error and poor access control onto the system's network. User error, as in uniformed downloads, the insertion of unknown media onto machines and more, can be mitigated by continuous education, which should be provided to system collaborators. Ensuring a secure network is done using proven technologies such as firewalls, intrusion prevention systems (IPS), intrusion detection systems (IDS), and VPNs. (Rapid7, 2021)

  Besides these measures, the system should also have a suitable antivirus solution that performs regular analysis and constant monitoring to detect malware should it find its way onto the system. (Rapid7, 2021)

- SQL Injection (Rapid7, 2021)

  Description: Many commercial and open-source databases use Structured Query Language (SQL) to manipulate and manage data. A SQL injection attack targets these databases using maliciously crafted SQL statements to extract or modify data contained in the system's database. (Rapid7, 2021)

  Solution: Most SQL injection attacks happen due to poor sanitisation of user-provided inputs, which, as a rule, should always have any special characters replaced by harmless but equivalent ones. Additionally, a system should avoid using dynamic SQL and instead only use prepared statements and parameterised queries. The system should never leave sensitive

data in plaintext, and it should only give the bare minimum of privileges required to users, as it will limit what an attacker can do should they gain access. (Rapid7, 2021)

- Cross-Site Scripting (XSS) (Rapid7, 2021)

  Description: A cross-site scripting attack involves the injection of malicious client-side scripts into a user's web browser. These scripts can be injected in several ways and carry out undesirable actions from the user's browser. They could be transmitted through a malicious URL which an attacker tricked the user into clicking on (reflected XSS). Or they could be stored into a vulnerable website, either directly on its hosted content (persistent XSS) or in the vulnerable client-side scripts provided by the site/app (DOM-based XSS). (Rapid7, 2021)

  Solution: The best way to avoid an XSS attack is by sanitising all user inputs and encoding outputs to prevent malicious scripts from automatically loading and running. The system should also limit the use of user-provided data. Additionally, implementing a Content Security Policy (CSP) will provide additional levels of protection. (OWASP Foundation, 2021b)

- Brute-Force and Dictionary Attacks (Duarte, 2008)

  Description: Brute-force and dictionary attacks are designed to guess the authentication data of registered and trusted users. Brute-force attacks involve going through every possible combination of letters, numbers and symbols until a match is found. Dictionary attacks, instead of random combinations, try common ones used by many users, such as "password", "12345", "asdf", and many more. (Duarte, 2008)

  Solution: All passwords, given enough time, will eventually be discovered by brute-force and dictionary attacks. The best way to defend against them is by having a long and complex password, as it could take years for a brute-force attack to guess it, and a dictionary attack will not find it in its word database. (Rapid7, 2021)

  Aside from that user side safety measure, a system could enforce some delaying standards that activate in response to multiple failed login attempts. A system could artificially delay repeated logins; a legitimate user would barely notice a slight delay, but it would quickly pile up for an attacker. A system could force the completion of captchas as these have become difficult for computers to complete, or it could simply lock the account. (Rapid7, 2021)

  Besides those delaying measures, a system could also monitor user accounts, signalling any unusual behaviour such as logins from unrecognised locations or devices or multiple failed login attempts. These alerts could then be handled in real-time by a Security Operations Center (SOC). (Rapid7, 2021)

- Phishing Attacks (Duarte, 2008)

  Description: Phishing attacks are designed to trick normal users into divulging private information such as passwords, credit card numbers and banking information. They do this by employing social engineering techniques in messages sent in bulk to a high number of

users. These messages are designed to enlist an emotional and impulsive reaction from the receivers, with the usual goal of making them follow a link into a website controlled by the attacker. This website sometimes mimics known and trusted ones, but the attacker then receives all information placed here by the user. (Rapid7, 2021)

Large organisations need to be very cautious of phishing attacks. Due to the high number of employees and collaborators, there is a high chance of at least one of them falling victim to one of these attacks, mainly because attackers sometimes tailor these messages to contain company logos and other identifiable marks. And the capture of the credentials of a high position member could mean the loss of company secrets, company accounts and much more. (Rapid7, 2021)

Solution: The best solution against phishing attacks is continuous user education and exercise. By reminding users to be vigilant constantly, it is possible to diminish the chances of a phishing attack being successful drastically. A good measure is also to promote the usage of two-factor authentication. This measure would ensure that if the credentials were stolen, there would be an extra step to the login process, not easily performed by an attacker. (Rapid7, 2021)

Aside from that solution, some mail and messaging services also have mechanisms to detect and filter emails and messages with malicious URLs and suspicious attachments. (Rapid7, 2021)

2.3.3 Network Intrusion Cover-Up Methods

After a successful network intrusion attack, attackers will usually cover their tracks and avoid detection. Delaying the fixing of the exploited vulnerability allows the attacker to perform the same action. This practice can also hide the changes made to the system components or data. And it can also be used to avoid prosecution. (Awake Security, 2021)

Some network intrusion attacks, such as living off the land and using non-malware tools, have the advantage of blending into business justified usage, making them harder to detect. In addition, there are three practices used by attackers to circumvent security infrastructures. (Awake Security, 2021)

- Deleting logs (Awake Security, 2021)

  Description: Deleting logs is the action of deleting access logs after an attack has been accomplished. Deleting these logs makes it almost impossible to understand the attackers' actions while accessing the system. (Awake Security, 2021)

  Solution: The best way to combat this practice is to regularly review logs and centralise them into a separate secure location where the attackers can't tamper with them. (Awake Security, 2021)

- Using encryption on departing data (Awake Security, 2021)

Description: When stealing data from a system, attackers tend to encrypt it or, at least, try to pass the transmission as regular system traffic. This practice is used to obfuscate their movements from security infrastructures. (Awake Security, 2021)

Solution: There are multiple ways to combat this practice. Developers can set up firewalls to stop any unauthorised outgoing traffic. Most monitoring tools or intrusion prevention systems (IPS) detect abnormal outgoing traffic. And a robust logging system can keep track of the data accessed during the attack.

- Installing rootkits (Awake Security, 2021)

  Description: After a successful intrusion attack, attackers will usually install software that allows them to access and control the system without being detected. These software are traditionally called rootkits, and attackers use them to analyse a system calmly and exploit it over long periods. (Awake Security, 2021)

  Solution: The detections of rootkits can be tricky as they operate on the same level as the operating system. Some antivirus solutions have a rootkit scan specifically designed to detect rootkit infection on the device, which should be run regularly. It is also possible to detect the presence of the rootkit using behavioural analysis performed by a monitoring system. (Kaspersky, 2021)

2.3.4 DoS Attacks

Denial-of-Service (DoS) Attacks have a different goal than the ones presented before. The purpose of these attacks is not to steal information or gain control over the system. These attacks wish to interrupt a system's normal functions and block access to regular users. They do this by flooding the system with useless requests or meaningless data to process, congesting the system so that valid requests and data cannot be processed. (Vyncke and Paggen, 2007)

Due to the massive improvements in the processing power of machines, especially the ones used by commercial services, simple DoS Attacks are rare. Because a typical device can not generate enough data to block a target completely, attackers usually coerce multiple machines to act in coordination to perform the attack. This coordinated attack performed by various devices is called a Distributed Denial-of-Service Attack (DDoS). (Duarte, 2008; Kaspersky, 2021)

There are multiple techniques to perform a DDoS attack. Next in this section, 3 of these techniques are highlighted as they are the most used.

- DNS Amplification (Kaspersky, 2021)

  Description: This DDoS attack abuses the DNS technology to generate massive amounts of data to flood the target device. An attacker makes multiple small-sized requests to a DNS service while impersonating the target device by placing the victim's IP address as the source. The DNS's responses, which are much larger than the requests, are then sent to the target device, flooding him with data. (Kaspersky, 2021)

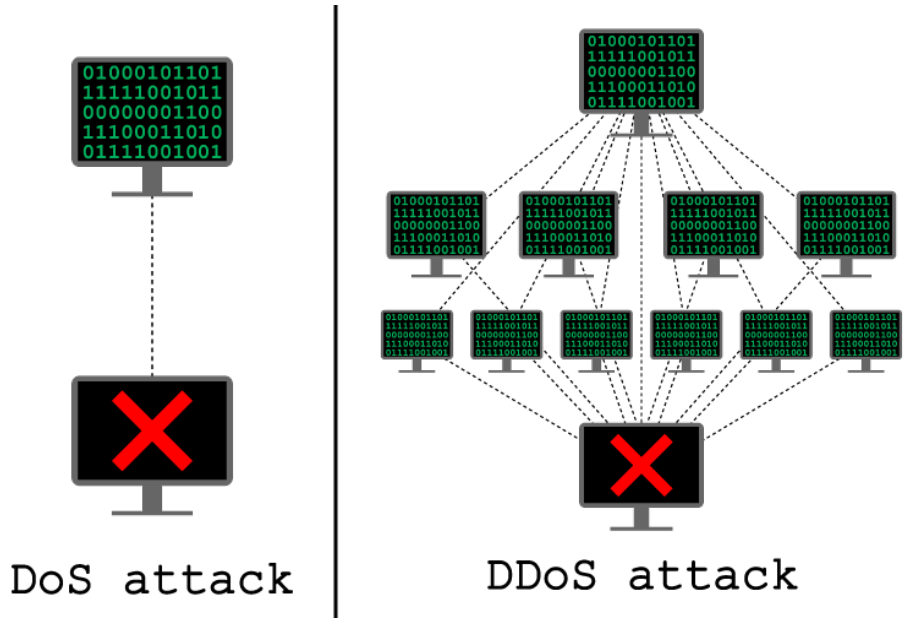

Figure 2.2: DoS Attacks compared to DDoS Attacks (EC-Council, 2021)

- Smurf Attack (Kaspersky, 2021)

  Description: To perform this DDoS attack, an attacker broadcasts network requests (ICMP) while impersonating the victim's device, using a technique called IP spoofing. The victim is then flooded with the responses from all network nodes.

- Trojan-DDoS (Kaspersky, 2021)

  Description: A trojan-DDoS is a malicious program that attackers place on multiple vulnerable devices. After a substantial number of devices are infected with this program, the attacker can then coordinate and control them to send numerous requests to a target device, flooding and blocking him.

It is tricky to defend systems against DDoS attacks. Some paid services provide additional processing power to systems temporarily. There is also the technique of Sinkholing, which consists of redirecting traffic to a sinkhole server that analyses and blocks it. (Kaspersky, 2021) Monitoring tools can detect and block abnormal traffic before reaching the system. But all these techniques merely defend against the attack. Any system will inevitably fall against a big enough DDoS attack. The best defence against these attacks is for the system to be efficient and have enough processing power to withstand it.

Chapter 3

Monitoring and Logging Tools

3.1 Overview

Previously in this dissertation, monitoring and logging tools have been said to detect and defend systems against cyberattacks. The OWASP Foundation has gone so far as to consider failures in these systems as one of the top vulnerabilities plaguing systems in 2021 (OWASP Foundation, 2021a; Rivera-Ortiz and Pasquale, 2020). In this chapter, this dissertation will better explain what these tools are, how they function, the challenges they face, and the advantages they bring when implemented into a system.

The simplest definition of a monitoring and logging tool is a tool that is capable and responsible for detecting every action performed within a system and logging it for future consulting. Besides detecting and logging, such tools can also be programmed to respond automatically to specific types of actions, such as those that violate internal policies. (Creasey, 2015)

Besides detecting and responding to outright policy violations, these tools can define a "normal" state for the system using the log history. If that "normal" state is compromised, either with abnormally high network flow, uncommon access locations, or other common signs of an outgoing attack, it can detect that abnormal state and respond accordingly. This response can be to alert security personnel or perform a more specialised automated response. (Oliveira, 2020; Qin et al., 2018)

In this section, this dissertation will go through a detailed look at the inner workings of these tools. It will go from a more in-depth look at their purpose and goals to the abstract phases of monitoring a system. It will also present guidelines for effectively creating, storing, and handling logs to effectively use a monitoring and logging tool.

3.2 Purpose and Goals

This dissertation has quickly gone through the essential functions of monitoring and logging tools. But this section will further break down their purpose and what they are used for, both in cyber security and support functions for the system or business.

Regarding cyber security, monitoring and logging tools are used for the identification of security incidents, to monitor policy violations, and help defend against vulnerability identification and exploitation; in short, monitoring and logging tools help to avoid and detect cyberattacks (Creasey, 2015; Wang et al., 2019). They also serve to establish system baselines, such as network traffic and system access patterns, which, when deviated, can signify an ongoing attack (Oliveira, 2020; Qin et al., 2018). They can also provide insightful information about problems and unusual conditions and supply additional application-specific data for incident investigation (OWASP Foundation, 2021c).

Besides security-related functions, monitoring and logging tools can also provide valuable business information, sometimes neglected. These include sales data, audit trails (e.g., data addition, modification and deletion, and data exports) and system performance information such as data load time and page timeouts. Additionally, they can also be used for the legally sanctioned interception of data (application-layer wiretapping). As well as supply data for subsequent requests for information (e.g., litigation information and data relevant to police and other regulatory investigations) and other business-specific requirements. (AppDynamics, 2021; OWASP Foundation, 2021c)

3.3 Logging Guidelines

As stated previously, a core part of these systems is creating and handling logs. These logs contain vast amounts of information, and by analysing them, it is possible to discern user behavioural patterns and system performance and even detect and track cyberattacks. Due to their content, logs should be stored centrally and be protected against unauthorised access and analysed regularly. (OWASP Foundation, 2021c)

Ideally, one would record and store every action performed by and within the system, but that is usually not possible. And even if it was possible, it might not be desirable. When dealing with a platform that serves millions of users, storing every action that every user performs can quickly pile up (Rivera-Ortiz and Pasquale, 2020; Wang and Zheng, 2020). And even if enough storage space is available, these logs still need to be appropriately and effectively processed and presented to be of any use (Safdar et al., 2018).

Therefore, these logging systems should be configured to log suitable cybersecurity-related events and should be regularly tuned to reduce the number of false alerts to acceptable levels. These security-related logs should also be normalised into a standard and suitable format (Sukmana et al., 2018), and their timestamps should be synchronised to a common trusted source (OWASP Foundation, 2021c). They should also be adequately analysed because sometimes seemingly innocent and smaller logs can be correlated and aggregated to paint a much larger security incident (Rivera-Ortiz and Pasquale, 2020).

Additionally, the employer of the logging system should also establish cybersecurity-related logging standards and procedures such as log retention and rotation periods. And, of course, take

fast and effective actions to remediate any issues identified and respond to any cyber security incidents that happen. (Creasey, 2015)

Because of the challenges of storing and processing massive amounts of logs, developers need to prioritise what should and shouldn't be logged (Wang and Zheng, 2020). And although the specific selection should be tailored for the individual system or business, some standard guidelines exist as a basis for this process (Creasey, 2015; OWASP Foundation, 2021c; Rivera-Ortiz and Pasquale, 2020). These data selection guidelines and some regarding the format of the singular logs are presented in the following subsections.

3.3.1 Data Selection

As stated previously, it is essential to prioritise what data should and shouldn't be logged. This prioritisation is needed because logs can quickly pile up, mainly if the business or system has limited dedicated storage due to budget constraints (Wang and Zheng, 2020). Ideally, any company should perform an internal analysis and select the relevant data it wants and needs to log. But general guidelines exist on what to and don't log and which sources are most important from a security standpoint (Creasey, 2015; OWASP Foundation, 2021c; Rivera-Ortiz and Pasquale, 2020). These guidelines are presented in this section.

The following table contains a list that complies experts' opinions on which types of logs are typically most important for identifying possible cyber security attacks. It also includes their relative cost to obtain, i.e., purchase costs and ongoing use of any relevant tools and services. Technically all logs have an associated cost (e.g., acquiring, storing, and analysing), but in the table, log sources marked as "Free" are already available in systems, networks, tools, or services the organisation should already have. It is important to note that the priority stated in this table is not valid for every business or system, and log management should be evaluated for each case. (Creasey, 2015)

Table 3.1: Log Sources (Creasey, 2015)

Log Source	**Priority**	**Relative Cost**
Email	Essential	Free
HTTP proxy	Essential	Free
Malware protection logs	Essential	€
NIDS	Essential	€€
NIPS	Essential	€€
System activity logs (e.g., Admin)	Important	Free
Firewall	Important	Free
DNS	Important	Free
DHCP	Important	Free
Web Server Logs	Important	Free
SQL Server Logs	Important	Free
Sandboxing techniques (including virtual execution engines)	Important	€€€
Endpoint (and agent-based) logs	Useful	€
Authentication logs (e.g., Windows)	Useful	Free
Physical	Useful	€€
VPN	Useful	Free
Netflow	Useful	Free
FTP	Useful	Free
Appflow	Useful	Free
Data loss protection (DLP)	Useful	€€

Besides considering the source of logs, one should also investigate individual events and ponder which should be logged. These are again highly dependent on the system or business, and a blind checklist approach can lead to real problems being undetected amidst the clutter (OWASP Foundation, 2021c; Wang and Zheng, 2020). But some events are highly recommended to be logged.

When possible, one should always log validation failures: input, e.g., protocol violations, unacceptable encodings, invalid parameter names and values, or output, e.g., database recordset mismatch and invalid data encoding. Authentication successes and failures and session management failures, e.g., the modification of the cookie session identification value. Authorisation failures issue by the access control system. Application and systems start-ups, shutdowns, logging initialisations, and any errors these might encounter, e.g., syntax and runtime errors, connectivity problems, performance issues, third party service error messages, file system errors, file upload virus detection and configuration changes. And legal and other opt-in options information such as permissions for mobile phone capabilities, terms of use, terms and conditions, personal data usage consent and authorisation to receive marketing communications. (OWASP Foundation, 2021c)

Additionally, if possible, one should always log any use of higher-risk functionalities. These

are the establishment of network connections. The addition or deletion of users, changes to privileges, assigning users to tokens, adding or deleting tokens. The use of systems administrative rights, access by application administrators, all actions by users with administrative privileges. Access to payment cardholder data, the use of data encrypting keys, key changes, creation, and deletion of system-level objects. And data imports and exports, including screen-based reports and submission of user-generated content, especially file uploads. (OWASP Foundation, 2021c)

Optionally, if possible and desirable for the system or business, it is valuable also to log sequencing failures, excessive use of system elements and changes to data, configurations and application's code files and memory. And it could also prove helpful to log any incident of fraud and other criminal activities and suspicious, unacceptable, or unexpected behaviour. (OWASP Foundation, 2021c)

Parallel to the previous list, there is also a list of what should be excluded from the logs. This list mainly comprises data not legally sanctioned to be logged and stored. Therefore, the elements of the following list should not usually be recorded directly in the logs. Preferably they should be removed, but if a business deems them necessary, they should take measures to mask, sanitise, hash, or encrypt the data. (OWASP Foundation, 2021c)

There should not be a direct record of applications' source code, access tokens and session identification values (if these are needed to track session-specific events, they should be replaced with a hashed value). Also, systems shouldn't record sensitive personal data and some forms of personally identifiable information (PII), e.g., health, government identifiers and vulnerable people identifiers. They should never directly store authentication passwords, database connection strings, encryption keys and other master secrets, bank account or payment cardholder data and commercially sensitive information. And they should never log data of a higher security classification than them, information that is illegal to collect and that the user has opted out or not consented to collect. (OWASP Foundation, 2021c)

Additionally to those elements, one should also have unique care when logging file paths, internal network names and addresses and non-sensitive personal data, e.g., personal names, telephone numbers and email addresses. (OWASP Foundation, 2021c)

A good rule of thumb when handling personal data when the individual's identity is not required is to implement de-identification techniques such as deletion, scrambling or pseudonymisation of direct and indirect identifiers. (OWASP Foundation, 2021c)

3.3.2 Formatting Guidelines

Having looked at what should and shouldn't be logged, this dissertation will now focus on the individual logs' format. As stated previously, records should be normalised and consistent, meaning they should have a defined form (Sukmana et al., 2018). The specific structure is dependent on the system's needs and goals, but there are some required fields that a log should have (OWASP Foundation, 2021c).

Logs should have a date and time in a standardised format that is consistent for all records and synchronised with a standard and trusted source. They should identify the precise location where

the logged event occurred. This location identifier could be the application or service name and version, and application address (e.g., cluster/hostname or server IPv4 or IPv6 address and port number, workstation identity, local device identifier), geolocation, a window/form/page or even a location in the code. Additionally, they should also have who performed the event being logged, be it human or machine, e.g., user identity or source address. And a description of the event being logged. (OWASP Foundation, 2021c)

Besides those fields, logs could also contain additional information such as the original intended action, the affected objects, the result status, and the result's reason. They could also have the HTTP codes, headers, or user agents. As well as the user type (e.g., public, authenticated user, CMS user, search engine, authorised penetration tester, uptime monitor), the event responses and other extended details such as stack trace, system error messages, debug information, HTTP request body, HTTP response headers and body. (OWASP Foundation, 2021c)

3.4 The Monitoring Process

After investigating the logging process of these systems, this dissertation will now focus on the monitoring process. The main goal behind monitoring a system is to detect potential cyber security incidents. This detection is done by looking for indicators of compromise (IOC). (Creasey, 2015)

Indicators of compromise (IOC) are identified by a combination of event identification, cyber security intelligence and the individual system context. The diligent keeping of logs does the event identification. And cyber security intelligence encompasses the knowledge of threat agents, their sources and motives, of attack methodologies, as well as attack types and tools. (Wang et al., 2019)

The monitoring process can be divided into four core phases performed in a continuous loop that combine these three elements. These phases are the collection phase, the fusion phase, the analysis phase, and the action phase. (Creasey, 2015)

The **collection phase** is mainly automated, and, as the name suggests, it is in this phase that logs are collected. Additionally, records are normalised, filtered, and stored in this phase. (Creasey, 2015)

The **fusion phase** is also automated. In this phase, mechanical systems fuse seemingly benign logs into larger, more complete pictures, revealing possible indicators of compromise (IOC). (Creasey, 2015)

The **analysis phase** is where human intervention and cyber security intelligence is most required. In this phase, professionals, armed with the knowledge of current attack patterns and vulnerability trends, analyse the most pertinent logs of the last step to determine if any of them indicate a compromise, an actual IOC. (Creasey, 2015)

The **action phase** is the last phase of the loop, and although it mostly requires human intervention, some of it can be automated. In this phase, a specific response to the detected IOC is

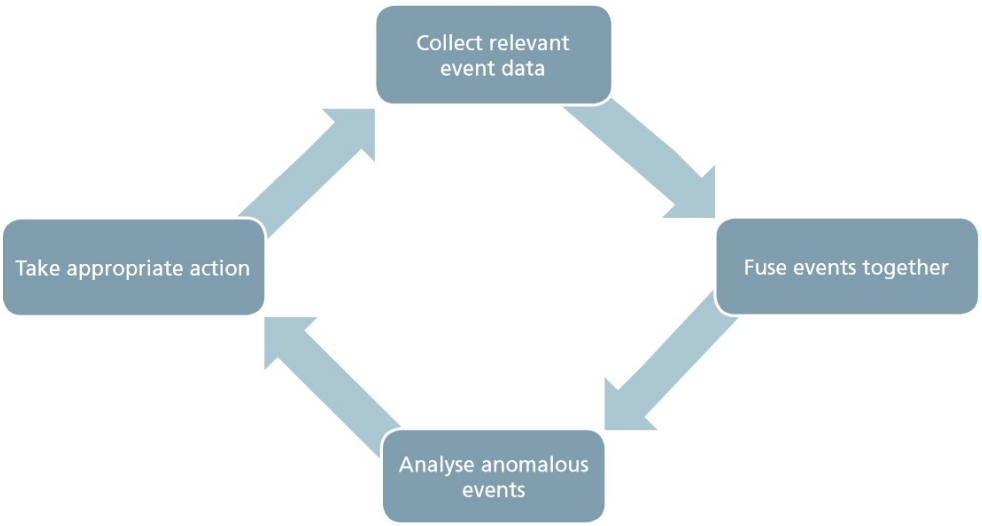

Figure 3.1: Cyber Security Monitoring Phases (Creasey, 2015)

performed. Some examples of actions performed in this phase are remediation and recovery, escalation both internally or externally, further investigation, reporting or any other kind of incident management. (Creasey, 2015)

3.5 Market

Now that the concept of monitoring and logging tools has been established. This dissertation will give a short breakdown of tools in this space currently available in the market. This breakdown includes relevant security information and event managers (SIEM), which focus on providing customers with real-time analysis of event logs and processing security alerts and endpoint security tools, focusing on maintaining and ensuring the security of machines within a network. This list is by no means intensive, nor is it a deep dive into the characteristics of these tools. It is simply a starting point for people who need such tools for their own companies or systems.

- IBM QRadar (IBM Security, 2022)

  QRadar by IBM Security is a Security Information and Event Manager (SIEM) to help security teams detect, prioritise, and respond to threats across an enterprise. Its focus is primarily on serving businesses. Its most significant appeal is his capability to monitor thousands of devices, endpoints, and apps across a network.

- LogRhythm NextGen SIEM (LogRhythm, 2022)

  NextGen SIEM by LogRhythm is a SIEM tool whose focus is primarily on automation, machine learning and artificial Intelligence. It is award-winning software that minimises the time needed to respond to any attack or threat.

- OSSIM (AT&T Business, 2022)

  The Open-Source Security Information and Event Management, abbreviated to OSSIM by AlienVault, is an open-source SIEM solution created by security engineers to fill the gap of open-source products in the market. It provides the essential security capabilities needed within a unified platform while allowing users to contribute and receive real-time information about malicious hosts by leveraging AlienVault's Open Threat Exchange.

- RSA NetWitness (RSA Security LLC, 2022)

  NetWitness by RSA is a SIEM solution that also utilises Open XDR to have an AI-powered and accelerated approach to detection and response. It can collect and analyse data from multiple sources such as logs, packets, Netflow and endpoints while being compatible with any computing platform, physical, virtual or cloud-based.

- Sagan (Quadrant Information Security, 2022)

  Sagan by Quadrant Information Security is an open-source log analysis engine that utilises multi-threaded technology for higher performance. It also contains the capabilities to track user behaviour and create custom security rules and alerts. Combining seamlessly with other specialised tools can easily be coupled into a software stack to perform highly efficient log analysis.

- Snort (Cisco, 2022)

  Now developed by Cisco, Snort is an open-source network intrusion detection system (IDS). It works as a package sniffer and, by using its set of rules, it can define and detect malicious network activity and generate alerts in response.

- Sumo Logic (Sumo Logic, 2022)

  Sumo Logic focuses on providing data analysis tools that use cloud technology. Concentrating on cyber security, they offer a massive collection of features from SIEM capabilities, intrusion detection and prevention systems, cloud monitoring, and virtualisation tools monitoring and are compatible with most software being used today.

- Wazuh (Wazuh Inc., 2022)

  Wazuh is an open-source security monitoring solution. While having scaling capabilities up to enterprise levels, it provides many essential security features within a single software. These include intrusion detection, file integrity monitoring, vulnerability detection, incident response and cloud and containers security.

Chapter 4

Case Study: The DataPlaxe Platform

4.1 Overview

The DataPlaxe platform is a cloud service that aims to incorporate the functions of a Smart IoT platform with the storage, injection, and processing capabilities of a Data Lake. While simultaneously providing components for the execution of analytic or machine learning processes. Additionally, it gives a functional web interface for managing, exploring, and visualising data and processes.

The platform is unique as it allows users to upload both data and data-processing code into a controlled and scalable environment, providing users with enough processing power and resources to run their code with their data as they desire. As it aims to step into the realm of Big Data, the platform seeks to grab the attention of companies that wish to compile and draw conclusions from considerable amounts of data, such as finances or sales histories. These companies usually do not have the in-house resources to process this data, and the market still hasn't any great solutions to solve this issue.

Additionally, the DataPlaxe platform also contains a marketplace. In this marketplace, users can monetise their data, processing algorithms or even visualisation tools. A possible use case may occur because some companies wish to relate their collected data with data they don't have, but a different user might provide. For example, a company may want to know how weather conditions impact sales; instead of tracking the weather themselves, they can buy such a collection of data from a company that provides it through the DataPlaxe marketplace.

By allowing its users to use their processing code, the DataPlaxe platform offers total control and flexibility over their data. However, due to the unpredictability of the uploaded code, ensuring the system's integrity is a very complex and arduous task. Because this code will run on the platform's environment. It opens the system to many forms of attack, which must be appropriately contained and stopped to avoid abuse of the system's resources and stored data.

4.2 Architecture and Components

The DataPlaxe platform comprises three server clusters connected by three private networks. The clusters are the Data Ingestion and Access Layer cluster, the Platform Service cluster and the Administration and Service Monitoring cluster. At the same time, the three networks are the Client Network, the Data Network, and the Management Network. The DataPlaxe platform segregates its network traffic using these three networks, increasing efficiency and security.

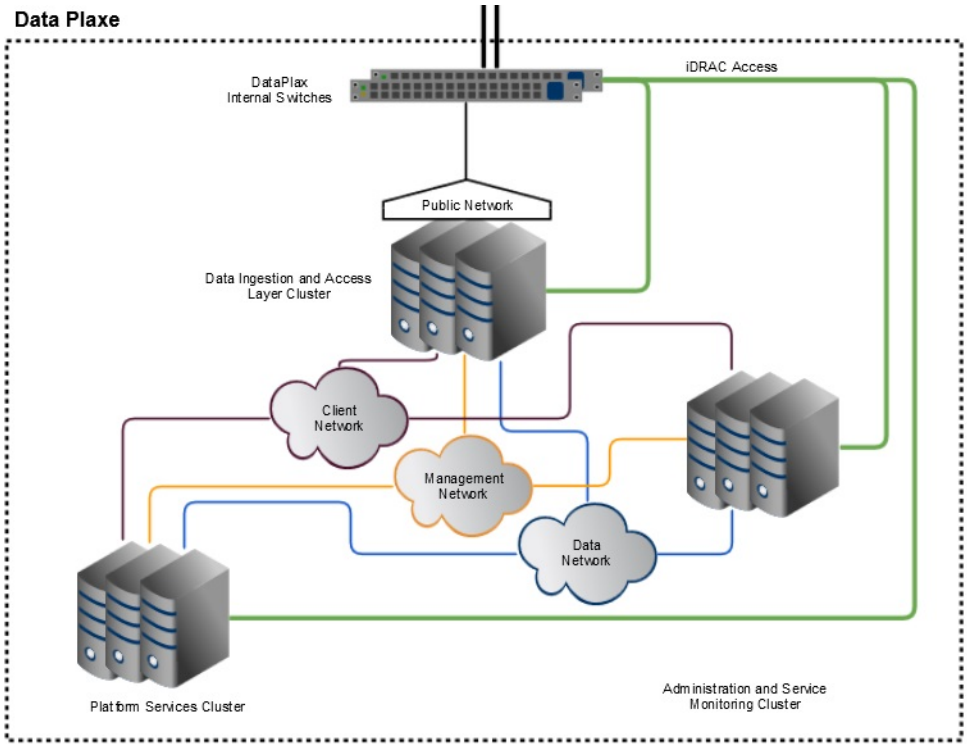

Figure 4.1: Overview of the DataPlaxe Architecture

The **Data Ingestion and Access Layer** cluster also referred to internally as the edge nodes, is the only one connected to the Public Network. Because of this fact, this cluster is responsible for exposing the DataPlaxe's internal services for data ingestion, querying, and reporting. It contains an OpenResty (OpenResty Inc, 2022) web server and a Minio S3 (MinIO, 2022) storage system. This cluster's services and its associated ports can be seen in the following table.

Table 4.1: Data Ingestion and Access Layer cluster's Ports and Services

Service	**Short Description**	**Port**
AuthD	Handles the authentication process of the DataPlaxe platform.	9755
Job Manager	Service that handles the scheduling of jobs withing the platform	9753
Plaxe Manager	Sevice responsible for handling domains and associated users	9751
Web Console	Service responsible for receiving requests and operation for the platform to process.	8585
Web Marketplace	Exposes items for acquisition and tracks subscriptions details	9595

The OpenResty (OpenResty Inc, 2022) web server redirects requests into the correct services. It works as a proxy for all user available functions and processes. All incoming traffic for the DataPlaxe platform will first meet the OpenResty web server and then be forwarded to the appropriate service. The Minio S3 (MinIO, 2022) storage system is used to hold information and execution logs about the jobs executed by the users. This information is present in this cluster to be easily accessed and consulted using the web console interface component of the DataPlaxe platform.

As said previously and made clear by its name, it's through the **Data Ingestion and Access Layer** cluster that all data enters the DataPlaxe platform. This injection process is driven by accessing the Web Console service, proxied by the OpenResty web server, which, upon receiving a request, forwards it to the Plaxe Manager service the appropriate parameters for data injection. Upon receiving those parameters, a user can insert data into the primary Minio S3 storage system present in the **Platform Service** cluster.

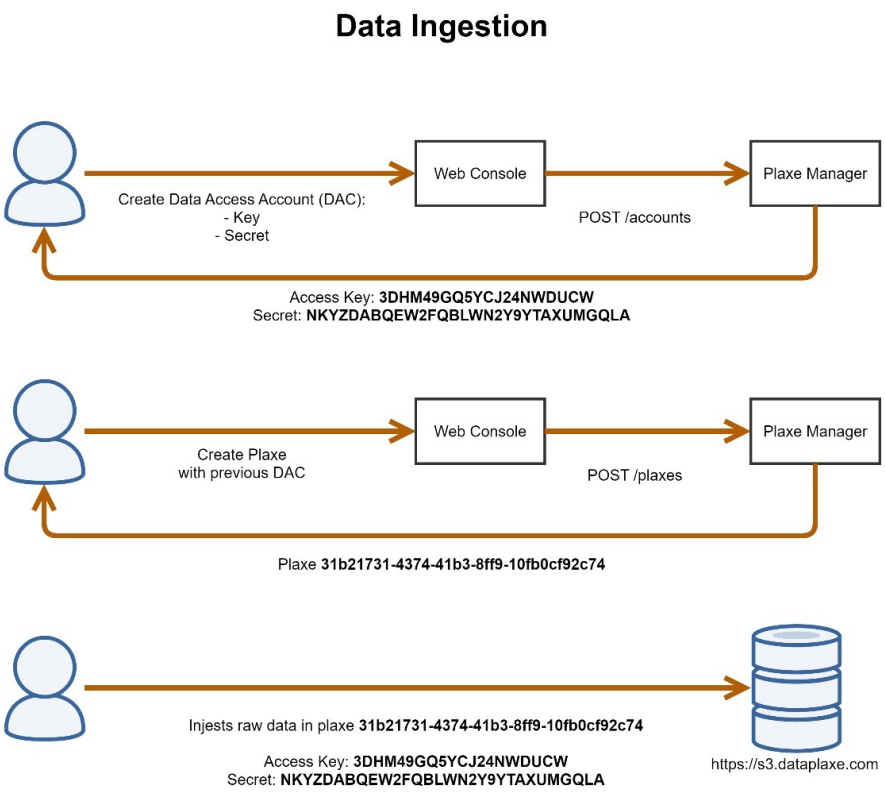

Figure 4.2: Data Injection Process of the DataPlaxe platform.

The **Platform Service** cluster, also referred to internally as the data nodes, is the cluster that holds all the platform primary services and processes for data storage and processing. It utilises Minio (MinIO, 2022) as its S3 storage solution, having a cluster that holds the DataPlaxe users' data. At the same time, it has access to the Minio S3 cluster present in the edge nodes, as this one is mounted in this cluster. Its data processing services utilise Docker (Docker Inc., 2022) containers and the combination of Hive (The Apache Software Foundation, 2022) with Presto (The Presto Foundation, 2022) as the SQL query engine for data contained in the S3 storage cluster. Additionally, it uses Metabase (Metabase, 2022) as the data analytics and visualisation tool of choice, allowing users to perform queries on their data and representing them in appealing graphics. This cluster's services and its associated ports can be seen in the following table.

Table 4.2: Platform Service cluster's Ports and Services

Service	**Short Description**	**Port**
Job Executer	Handles the execution of a requested operation	9752

As stated previously, it's the **Platform Service** cluster that handles data processing within the DataPlaxe. Still, the request for such a service must first come from the **Data Ingestion and**

Access Layer cluster. When a user requests a data processing operation or a job for short, it does so in Web Console service available in the edge nodes. This request is then forwarded to the Job Manager, that schedules the job and places it into a pool stored in a PostgreSQL database (The PostgreSQL Global Development Group, 2022). The Job Executor Service then picks up the job within the **Platform Service** cluster. After that, the job is executed in a contained environment using a Docker container. This process is illustrated in the following image.

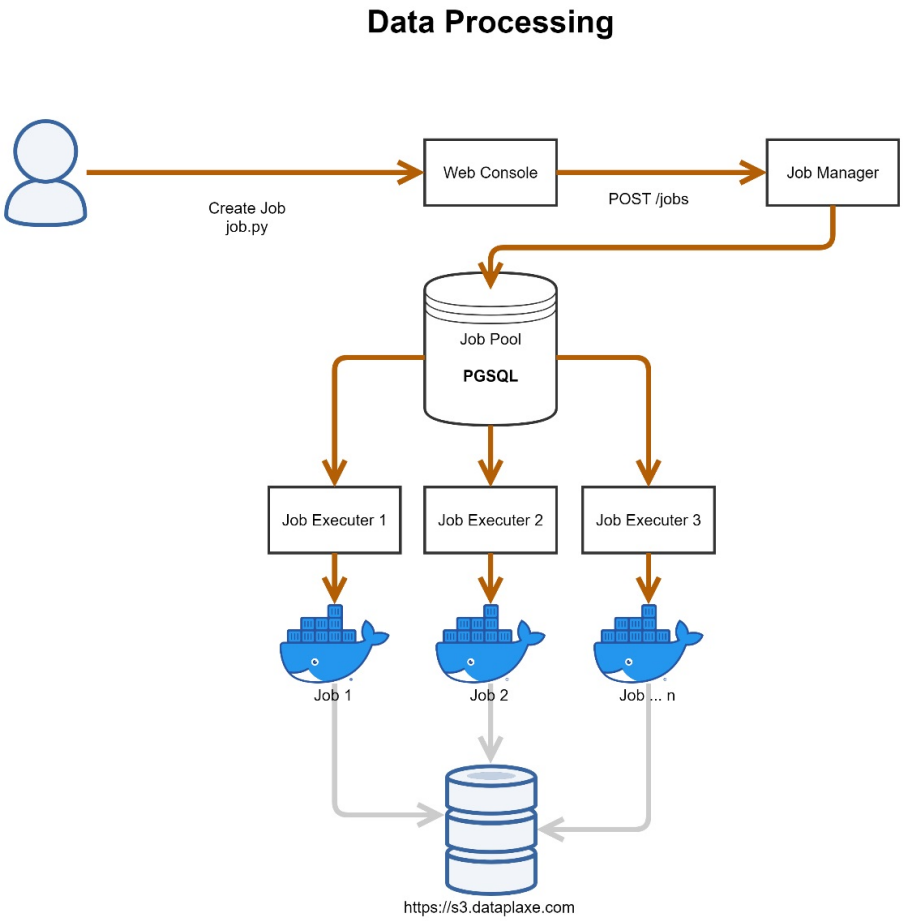

Figure 4.3: Data Processing Process of the DataPlaxe platform.

The **Administration and Service Monitoring** cluster also referred to internally as the admin nodes, is responsible for monitoring and managing the tools used in the DataPlaxe platform. One of its primary functions is to keep the base software of all clusters upgraded and up to date. Additionally, the core components of a monitoring and logging tool shall be deployed in this cluster.

The **Client Network** is a private 10Gbit internal network that connects the Internet-facing edge nodes to the backend platform services. Because it needs to accommodate bulk data ingestions, it needs a large bandwidth.

The **Data Network** is where the bulk of the traffic of the DataPlaxe platform will occur, although most will occur within the Platform Service cluster. It is also a 10Gbit network, but it uses dual connections with active load balancing for added bandwidth and redundancy in case of hardware failure.

The **Management Network** holds all traffic dedicated to managing and monitoring the different clusters of the DataPlaxe platform. Additionally, it provides SSH access to cluster nodes for administration.

Additionally, an Idrac Network is connected to the DataPlaxe platform, allowing network access via VPN. This network exists to provide access to the platform for updates and support.

4.3 Security Needs

After specifying the DataPlaxe platform architecture, this section will now focus on the security needs for the platform. One of its main lacking features from a security standpoint is the absence of a monitoring and logging tool. Missing this crucial feature leaves the platform vulnerable to cyberattacks due to its lack of intrusion detection and any form of incident analysis and response. It is also essential to consider what this monitoring and logging tool must provide to maximise its usefulness within the DataPlaxe environment. From this consideration, the following list of priority features is compiled:

- Platform wide Logging and Monitoring

  Any monitoring and logging tool implemented onto the DataPlaxe platform must monitor and collect logs from the entire application, including all service APIs, data ingestions, and different clusters.

- Endpoint Detection and Response (EDR)

  The monitoring and logging tool implemented must detect suspicious behaviour that may reach the platform's open ports and server endpoints on the Internet-facing nodes of the Data Ingestion and Access Layer cluster. Dataplaxe utilises an Openresty web server, and the chosen tool must be compatible.

- Intrusion Detection

  Intrusion detection is a given necessity in today's network-dependent environment. The monitoring and logging tool implemented must monitor and track users to detect intruders and their means of entry.

- Malware Detection

  The Dataplaxe platform comprises many interconnected machines. An effective form of malware detection is crucial to prevent a single infected machine from polluting the entire system.

- Incident Response

  In today's internet world, an automated and programmed response to common types of attacks is a big necessity, as dependency on human responses is simply unfeasible.

- Security Information and Event Management (SIEM) incorporation or integration

  Due to the Dataplaxe platform complexity and expansion intentions, a proper SIEM solution capable of compiling and processing vast amounts of logs and presenting only the relevant information in a readable way is needed.

- Vulnerability Detection and Component Monitoring

  The implemented monitoring and logging system must detect vulnerabilities within the system. As these vulnerabilities are mostly expected to come from the different software components of the platform becoming outdated over time, these two points are heavily interlinked. But vulnerability detection can extend beyond the software components, as it also includes unnecessary open port monitoring, lacking input sanitisation and insecure connection usage.

- Containers Security

  Containers are a crucial necessity of the DataPlaxe platform. Due to its reliance on Docker containers to perform data manipulation and processing, it is imperative that the monitoring and logging tool implemented be compatible and able to monitor and log Docker related events properly.

Chapter 5

Implementing a Monitoring and Logging System in the DataPlaxe Platform

5.1 Approach

This dissertation has first taken a look at the cyber security field as a whole, highlighting in the process common security attacks and vulnerabilities that plague informatic systems. Next, it explored the concept of monitoring and logging tools, as these are often referred to as either solutions or countermeasures to detect and prevent cyberattacks. After that, this dissertation took a look into the DataPlaxe platform, a multi-tenant service made by Altice Labs whose lack of a monitoring and logging tool puts it at risk from cyberattacks.

In this section, this dissertation will focus on finding a monitoring and logging tool that meets all the security requirements of the DataPlaxe platform. After this selection, a test installation of the tool will be implemented. Various tests will then be performed to confirm if such a tool meets the basic requirements of a monitoring and logging tool and the more specific requirements of the DataPlaxe platform. These tests then serve two purposes. They will showcase the capabilities of monitoring and logging tools. They will demonstrate if the specific tool selected is desirable to be implemented onto the DataPlaxe platform or any other product from Altice Labs.

5.2 Selection

As stated previously, there are many monitoring and logging tools in the market, some more specialised and some more generic. It is essential to add direction and guidelines to the selection process of the monitoring and logging to be implemented onto the DataPlaxe platform. For this, it is crucial to consider the security requirements of the DataPlaxe platform, as a tool that meets more of them is more valuable than one that meets less. Additionally, the developers of Altice Labs have a general preference for open-source tools, so this can also be used as a decision making factor.

After considering the security requirements of the DataPlaxe platform, a study was performed on eight currently available tools on the market to see if they meet said requirements. These tools are QRadar by IBM Security (IBM Security, 2022), NextGen SIEM by LogRhythm (LogRhythm, 2022), OSSIM by AlienVault (AT&T Business, 2022), NetWitness by RSA (RSA Security LLC, 2022), Sagan by Quadrant Information Security (Quadrant Information Security, 2022), Snort by Cisco (Cisco, 2022), the SIEM solution by Sumo Logic (Sumo Logic, 2022) and Wazuh by the company of the same name (Wazuh Inc., 2022). The following table was created from this study, highlighting the requirements met by each tool and their price and if they are or not open-source.

Table 5.1: Monitoring and Logging tools Requisites Comparison.

		Monitoring and Logging Tools							
		QRadar	NextGen SIEM	OSSIM	NetWitness	Sagan	Snort	Sumo Logic	Wazuh
Requisites	Platform Wide L&M[1]	✓	✓	NC[2]	✓	✓	✓	✓	✓
	EDR		✓	✓	✓		✓	✓	✓
	Intrusion Detection		✓	✓	✓	✓	✓	✓	✓
	Malware Detection			✓	✓			✓	✓
	Incident Response		✓		✓			✓	✓
	SIEM	✓	✓	✓	✓	✓		✓	✓
	Vulnerability Detection			✓	✓			✓	✓
	Component Monitoring							✓	✓
	Containers Security							✓	✓
	Open-Source			✓		✓	✓		✓
	Price	$\frac{\$800}{m}$	$\frac{\$28k}{y}$	Free	$\frac{\$857}{m}$	Free	Free	$\frac{\$13.75/h}{m}$	Free

QRadar by IBM Security (IBM Security, 2022) fails to meet most requirements of the DataPlaxe platform; this is because it is a highly specialised log processing and managing tool. Although it is compatible with some other very famous specialised tools, it lacks most features as stand-alone software, which is why it was not chosen.

OSSIM by AlienVault (AT&T Business, 2022) was discarded because it works as an operating system, and there was a risk for the DataPlaxe platform not being fully compatible with it. Additionally, because the test implementation will only encompass a section of the platform, there was

[1] Logging and Monitoring
[2] Not Compatible with the DataPlaxe Platform

a desire to avoid unexpected conflicts between the platform parts that could arise from running on different operating systems.

Sagan by Quadrant Information Security (Quadrant Information Security, 2022) was not chosen for a similar reason as QRadar, as it is a highly efficient multi-threaded engine for log analysis. And because it needs to be coupled with other software, it was not selected.

Snort by Cisco (Cisco, 2022) is also a specialised tool that works as a network packaging sniffing software to detect malicious traffic. It was discarded because it needed other complementing tools to meet all the DataPlaxe requirements. Additionally, it was unclear if it could be easily configured to handle the three internal networks used by the platform.

NextGen SIEM (LogRhythm, 2022), by LogRhythm, and NetWitness (RSA Security LLC, 2022), by RSA, were not chosen because other tools met more of the requirements of the DataPlaxe Platform.

Finally, Sumo Logic (Sumo Logic, 2022) and Wazuh (Wazuh Inc., 2022) both meet all the requirements of the DataPlaxe platform. But Wazuh was ultimately chosen as the monitoring and logging tool to be implemented onto the DataPlaxe platform because, unlike the SIEM solution by Sumo logic, it is open-source and, as a bonus, free.

5.3 Wazuh

Since Wazuh has been chosen to be implemented into the DataPlaxe platform, this section serves as a further presentation of Wazuh and its relevant inner workings. However, this does not replace the tool's documentation, which can be found in Wazuh Inc. (2022).

As stated earlier, Wazuh is a free, open-source monitoring and logging tool. It offers many valuable features that our case study needs, such as security analytics, intrusion detection, log data analysis, vulnerability detection, and containers security. Additionally, it is easily scalable for any sized system (Wazuh Inc., 2022).

Wazuh was three integral parts, the Wazuh Agents, the Wazuh Manager, and the Elastic Stack (Wazuh Inc., 2022).

The Wazuh Agent is to be installed on the machines to monitor. These agents monitor the system endpoints and send information to the Wazuh Manager. Additionally, they provide prevention, detection, and response capabilities to their host machines (Wazuh Inc., 2022).

The Wazuh Manager is to be installed on a server machine or a cluster of servers. It receives the information from the many Wazuh Agents and processes it. It looks for indicators of compromise (IOCs) utilising thread technology and a set of decoders and rules, both general and custom (Wazuh Inc., 2022).

The Elastic Stack is a set of 3 software, Filebeat, ElasticSearch, and Kibana, all of each open source and made by Elastic (Elastic NV, 2022). Filebeat transfers pertinent events and alerts from the Wazuh Manager to ElasticSearch. Kibana is a web interface used for visualising and analysing data, being the most appealing entry point of this system for human eyes. ElasticSearch can be

seen as the database used by Wazuh. It works as a distributed full-text search and analytics engine, and it is here that Wazuh stores and indexes its alerts (Wazuh Inc., 2022).

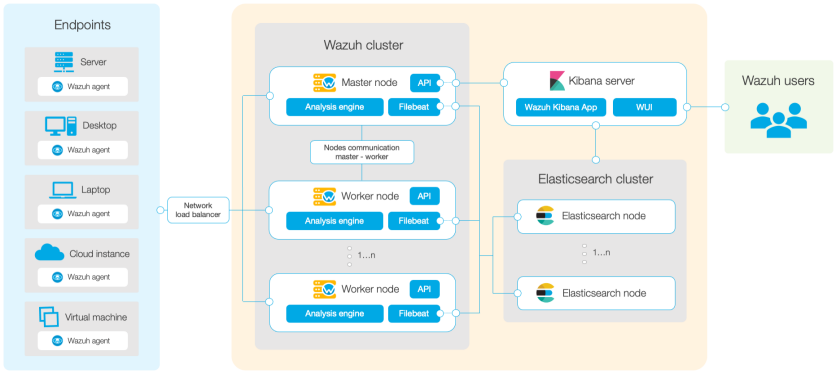

Figure 5.1: Overview of Wazuh's architecture (Wazuh Inc., 2022).

As stated above, the Wazuh Manager works by employing rules. These rules can be seen as pattern templates that trigger an alert upon matching with data sent by the Wazuh agents. These alerts are then associated with a severity level from 0 to 15, depending on the corresponding rule. Alerts with levels 0 to 3 are related to regular system operation and are usually discarded by the Wazuh Manager not to fill up storage with nonrelevant information. Alerts with levels 4 to 7 are connected with error messages sent by the system. Alerts with levels 8 to 12 are generated in response to unusual and suspicious events in the monitored machines, like multiple user-generated errors in quick succession or modification to necessary system files. Alerts with levels 13 to 15 are high severity alerts and should be investigated immediately as they most definitely indicate an ongoing attack. The full breakdown of these rules levels can be seen in Wazuh Inc (2022).

5.4 Tests Performed

5.4.1 Test Implementation

A simple test environment was created to perform tests to see if Wazuh meets the necessities of the DataPlaxe platform.

A Wazuh agent was installed in three Administration and Service Monitoring cluster machines: machines edge01, edge02 and edge03. These agents shall monitor these machines by collecting logs and sending them to the Wazuh Manager via a connection with AES encryption.

The Wazuh Manager was installed on a machine outside of the DataPlaxe environment. In this same machine, ElasticSearch and Kibana were also installed. ElasticSearch is a free and open-source search and analytics engine for all types of data used by Wazuh to process its logs. And Kibana is an ElasticSearch data visualisation and management tool, which also has a Wazuh plugin allowing it to visualise Wazuh logs and alerts. Additionally, the open-source software Filebeat sends Wazuh logs to ElasticSearch. This software stack is usually not installed on a single

machine, confusing the installation. But it was made this way to resemble better the structure that should be made if Wazuh is implemented in the DataPlaxe platform. The additional details of this installation can be seen in the following diagram.

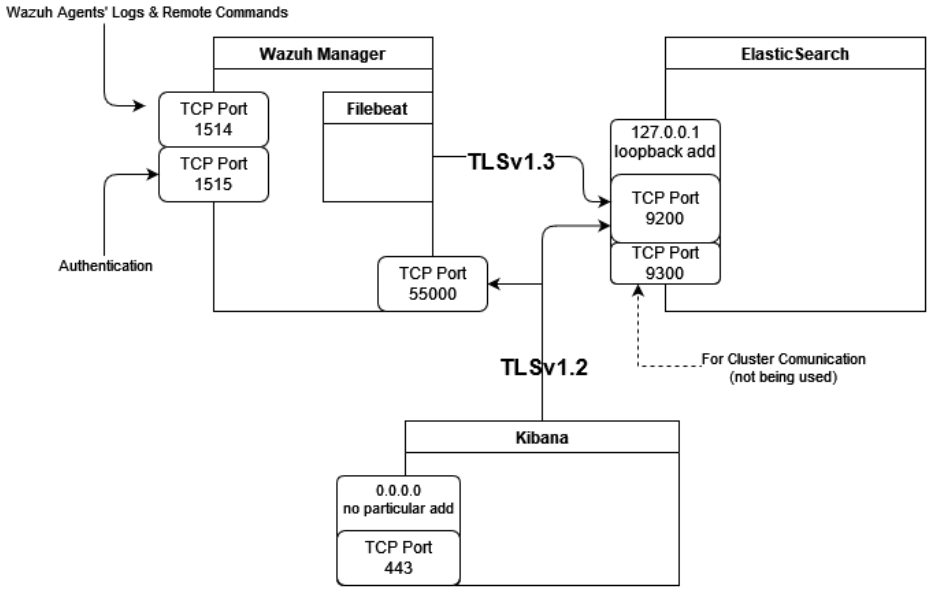

Figure 5.2: Installation diagram of the Wazuh Manager, Filebeat, Elasticsearch, and Kibana.

5.4.2 Tests Performed and Results

5.4.2.1 Detecting a Brute-force attack

- **Overview**

  This test performs a brute-force attack on a machine with a Wazuh Agent installed. It is expected that the test installation of Wazuh will be able to detect and correctly identify the multiple authentications attempts in quick succession by a user as a brute-force attack.

  This test aims to highlight the capabilities of a monitoring and logging tool to detect and correctly identify a cyber-attack, in this case, a brute-force attack. As well as demonstrate that Wazuh, while in the DataPlaxe platform, does have such capabilities.

  To perform the brute-force attack, we will use Hydra, an open-source parallelised login cracker. (Kali Linux, 2021)

- **Prerequisites**

  We will need an external Linux system with Hydra and SSH installed to serve as an attacker to perform this test.

- **Test Steps**

Run the following command to use Hydra to perform multiple logins attempts on a machine with a Wazuh Agent.

```
$ hydra -l <user_name> -p <user_password> <agent.endpoint> ssh
```

The implementation of the previous command used on the test is the following:

```
$ hydra -l victim -p wrong_password <machine edge1 IP address>
   ssh
```

In this case, a brute-force attack is performed on the machine identified as “edge1”. This attack tries to gain access to the “victim” ’s account. Because this is just a simple test, it will always attempt the password “wrong_password”. This field would be changed on every attempt in an actual situation, which would also be more numerous.

- **Test Results**

  After executing the previous command, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. There, we find that in response to the experiment, Wazuh has generated multiple copies of two alerts with rule IDs numbers 5710 and 5712. The abbreviated details of each can be seen below.

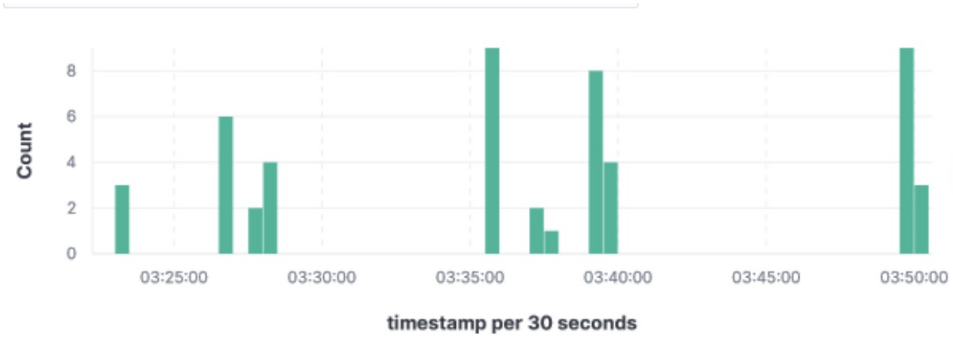

Figure 5.3: Graphic of the alerts generated by Wazuh in response to the brute-force attack

Table 5.2: The summary of the Wazuh alerts generated in response to the brute-force attack

Rule ID	Description	Level	Count
5710	sshd: Attempt to login using a non-existent user	5	47
5712	sshd: brute force trying to get access to the system.	10	4

Table 5.3: The truncated details of both Wazuh alerts generated in response to the brute-force attack

Field Name	**Alert Id: 5710**	**Alert Id: 5712**
rule.description	sshd: Attempt to login using a non-existent user	sshd: brute force trying to get access to the system.
rule.firedtimes	47	4
rule.id	5710	5712
rule.level	5	10
rule.mitre.tactic	Credential Access	Credential Access
rule.mitre.technique	*Brute Force*	*Brute Force*

Additionally to the field above, alerts with rule ID 5712 have an additional entry called previous_output. This field contains the multiple unsuccessful attempts that have been aggregated to trigger a warning with rule ID 5712. An example of such a field can be seen below.

```
"previous_output":

Dec 31 03:49:54 dplx-tst-edge1 sshd[17759]: Invalid user victim
    from 10.112.208.10 port 63674
Dec 31 03:49:51 dplx-tst-edge1 sshd[17757]: Failed password for
    invalid user victim from 10.112.208.10 port 63672 ssh2
Dec 31 03:49:49 dplx-tst-edge1 sshd[17757]: Invalid user victim
    from 10.112.208.10 port 63672
Dec 31 03:49:49 dplx-tst-edge1 sshd[17755]: Invalid user victim
    from 10.112.208.10 port 63670
Dec 31 03:49:45 dplx-tst-edge1 sshd[17747]: Failed password for
    invalid user victim from 10.112.208.10 port 63668 ssh2
Dec 31 03:49:43 dplx-tst-edge1 sshd[17747]: Invalid user victim
    from 10.112.208.10 port 63668
Dec 31 03:49:43 dplx-tst-edge1 sshd[17745]: Invalid user victim
    from 10.112.208.10 port 63666
```

As we can see, multiple alerts 5710 trigger an alert of type 5712, which means that Wazuh can identify numerous individual alerts of a failed user login as a brute-force attack.

5.4.2.2 Detecting an SQL Injection attack

- **Overview**

  This test performs a SQL injection attack on a machine with a Wazuh Agent installed. It is expected that the test installation of Wazuh will be able to detect and correctly identify the SQL patterns, such as "select" or "union", present in the attack.

  This test, much like the last, aims to highlight the capabilities of a monitoring and logging tool to detect and correctly identify a cyber-attack, in this case, an SQL Injection attack. It is also essential to demonstrate the ability of Wazuh to detect such attacks, as an SQL Injection attack on the DataPlaxe platform could prove disastrous as it is a data storing platform.

- **Prerequisites**

  We will use an Apache server running on the monitored machine to perform this test.

  Additionally, the Wazuh needs to be configured to capture events from the Apache server. This setup can be done by adding the following block to the monitored machine's `/var/ossec/etc/ossec.conf` file.

  ```
  <localfile>
  <log_format>apache</log_format>
  <location>/var/log/httpd/access_log</location>
  </localfile>
  ```

  After changing the configuration, it is necessary to restart the Wazuh Agent.

  ```
  # systemctl restart wazuh-agent
  ```

- **Test Steps**

  We shall send the following request to the Apache server from an external machine:

  ```
  $ curl -XGET "http://<apache_web_server_address>/?id=SELECT+*+
     FROM+users";
  ```

- **Test Results**

  After executing the command above, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. We can see the alert created in response to the previous server request, which has been correctly identified as a SQL injection attack, meaning that Wazuh can detect SQL injection attacks.

Table 5.4: The truncated details of the Wazuh alert generated in response to the SQL Injection attack

Field Name	**Alert Details**
data.url	`/?id=SELECT+*+FROM+users`
rule.description	*SQL injection attempt.*
rule.groups	web, accesslog, attack, sql_injection
rule.id	31103
rule.level	7
rule.mitre.tactic	Initial Access
rule.mitre.technique	Exploit Public-Facing Application

5.4.2.3 Detecting a Shellshock attack

- **Overview**

  A shellshock attack is a type of code injection attack. This attack aims to run shell commands on the target machine by sending maliciously crafted web requests. When a server receives these requests, the device is tricked into running foreign shell commands if proper filtering and escaping don't occur, causing disastrous effects.

  This test also aims to highlight the capabilities of a monitoring and logging tool to detect and correctly identify a cyber-attack. Although this is already demonstrated by the tests above, this particular cyber-attack requires greater attention as the DataPlaxe platform could be vulnerable to such types of attacks due to its architecture and data processing methodology. As the platform users provide the data processing code, malicious attackers could use this unique vector to attempt these cyberattacks. Should they be successful, they could compromise the Platform Service cluster machines, where data is stored.

  This test performs a shellshock attack on a machine with a Wazuh Agent installed. It is expected that the test installation of Wazuh will be able to detect and correctly identify the shell commands present in the attack.

- **Prerequisites**

  We will use an Apache server running on the monitored machine to perform this test. Additionally, the Wazuh agent needs to be configured to capture events from the Apache server. This setup can be done by adding the following block to the monitored machine's `/var/ossec/etc/ossec.conf` file.

```
<localfile>
<log_format>apache</log_format>
<location>/var/log/httpd/access_log</location>
</localfile>
```

  After changing the configuration, it is necessary to restart the Wazuh Agent.

```
# systemctl restart wazuh-agent
```

- **Test Steps**

  We shall send the following request to the Apache server from an external machine:

  ```
  $ curl -H "User-Agent: () { :; }; /bin/cat /etc/passwd" <
     apache_web_server_address>
  ```

- **Test Results**

  After executing the command above, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. We can see the alert created in response to the previous server request, which has been correctly identified as a Shellshock attack, meaning that Wazuh can detect Shellshock attack attempts.

Table 5.5: The truncated details of the Wazuh alert generated in response to the Shellshock attack

Field Name	**Alert Details**
data.url	/
full_log	"GET / HTTP/1.1" 403 4897 "-" "() { :; }; /bin/cat /etc/passwd"
rule.description	*Shellshock attack attempt*
rule.groups	web, accesslog, attack
rule.id	31166
rule.level	6
rule.mitre.tactic	Privilege Escalation, Initial Access
rule.mitre.technique	Exploitation for Privilege Escalation, Exploit Public-Facing Application

5.4.2.4 Detecting File Changes

- **Overview**

  This test aims to check if Wazuh can detect changes to files within the system. Additionally, it seeks to see what information is logged when these changes are detected and if it is possible to track the origin or author of the change.

  An integral part of recognising and tracing the route of cyber-attack is the logging of system file changes. These logs should include information on when these changes happen and who performed these changes. This capability of these tools is crucial for performing damage recovery and preventing repeated attacks from happening. As such, this test aims to

demonstrate this ability and observe if Wazuh provides satisfying amounts of details when file systems are changed.

- **Prerequisites**

  Wazuh file integrity monitoring is configured by default on installation. The default settings are present in `/var/ossec/etc/ossec.conf`, and from this file it is possible to see that Wazuh is already monitoring the following directories: `/etc`, `/usr/bin`, `/usr/sbin`, `/bin`, `/sbin` and `/boot`.

  To record the author of the events in the monitored directories, Wazuh needs an auditing subsystem. For Linux machines, this subsystem is the Linux Audit subsystem, which can be installed with the following command.

  ```
  # yum install audit
  ```

- **Test Steps**

  1. The following entry will be added in the Wazuh configuration file `/var/ossec/etc/ossec.conf`:

     ```
     <directories check_all="yes" report_changes="yes" whodata="
        yes" tags="test">/home/<user_name>/test</directories>
     ```

     This entry configures Wazuh also to monitor the directory `/home/<user_name>/test`. The modifier "check_all" allows recording file sizes, permissions, owner, last modification date, and more. At the same time, the modifier "whodata" will use the Linux Audit subsystem to record the author of the events in the directory. The modifier "report_changes" will ensure that the logs will also contain the changed content of the file.

  2. Restart the Wazuh-agent.

     ```
     # systemctl restart wazuh-agent
     ```

  3. After adding the entry, a test file will be added, changed, and deleted in the `/home/<user_name>/test` directory.

- **Test Results**

  The following tables show the Wazuh alerts generated in response to the changes in the folder `/home/<user_name>/test`. They correctly display the file's creation, the two changes made to it and its deletion.

Table 5.6: Alert generated by Wazuh in response to the creation of the test file.

Field Name	**1st Alert Details**
rule.id	554
rule.description	*File added to the system.*
rule.level	5
syscheck.audit.login_user.name	ptin_admin
syscheck.audit.effective_user.name	ptin_admin
syscheck.audit.process.name	`/usr/bin/touch`
syscheck.event	*added*
syscheck.path	`/home/ptin_admin/test/test_file.txt`

Table 5.7: Alert generated by Wazuh in response to the first modification of the test file.

Field Name	**2nd Alert Details**
rule.id	550
rule.description	*Integrity checksum changed.*
rule.level	7
syscheck.audit.login_user.name	ptin_admin
syscheck.audit.effective_user.name	ptin_admin
syscheck.audit.process.name	`/usr/bin/nano`
syscheck.event	*modified*
syscheck.path	`/home/ptin_admin/test/test_file.txt`

Table 5.8: Alert generated by Wazuh in response to the second modification of the test file.

Field Name	**3rd Alert Details**
rule.id	550
rule.description	*Integrity checksum changed.*
rule.level	7
syscheck.audit.login_user.name	ptin_admin
syscheck.audit.effective_user.name	root
syscheck.audit.process.name	`/usr/bin/nano`
syscheck.event	*modified*
syscheck.path	`/home/ptin_admin/test/test_file.txt`

Table 5.9: Alert generated by Wazuh in response to the deletion of the test file.

Field Name	**4th Alert Details**
rule.id	553
rule.description	*File deleted.*
rule.level	7
syscheck.audit.login_user.name	ptin_admin
syscheck.audit.effective_user.name	root
syscheck.audit.process.name	`/usr/bin/rm`
syscheck.event	*deleted*
syscheck.path	`/home/ptin_admin/test/test_file.txt`

If we take a closer look at the alerts generated, it is possible to see that they contain fields that allow the identification of the author of the changes and what process was used to make the changes. In the case of this test, it's possible to see that the file was created by the user "ptin_admin" with the command "`touch`" and then altered with the process "`nano`". After that, the user "ptin_admin", as effective user "root", changed the file again with "`nano`" and then deleted the file with the command "`rm`".

Table 5.10: A comparative look at the alerts generated in the detecting file changes test.

rule.description	[2]**login_user.name**	[2]**effective_user.name**	[2]**process.name**
File added to the system.	ptin_admin	ptin_admin	`/usr/bin/touch`
Integrity checksum changed.	ptin_admin	ptin_admin	`/usr/bin/nano`
Integrity checksum changed.	ptin_admin	root	`/usr/bin/nano`
File deleted.	ptin_admin	root	`/usr/bin/rm`

Additionally, both modification-related alerts have fields containing the details of the change made to the file.

[2]syscheck.audit.

Table 5.11: Additional field present in the modification related alerts.

Field Name	**2nd Alert Details**	**3rd Alert Details**
rule.id	550	550
rule.description	*Integrity checksum changed.*	*Integrity checksum changed.*
syscheck.event	*modified*	*modified*
syscheck.changed_attributes	size,mtime,md5,sha1,sha256	size,mtime,md5,sha1,sha256
syscheck.size_before	0	38
syscheck.size_after	38	73
syscheck.diff	0a1,2 > First line of text as a normal user. >	1c1 < First line of text as a normal user. --- > Replacing the first line of text as the same user but with root access.

As it is possible to see, Wazuh can catalogue and show both the author of the change and the changes made to the file. Therefore, Wazuh can provide crucial information on the aftermath of an attack for its identification and tracing.

5.4.2.5 Detecting Suspicious Binaries

- **Overview**

  This test aims to discern if the Wazuh installation can detect malware attacks by identifying suspicious binaries such as trojans or viruses. An original system binary is replaced by a "harmless" trojan version for this test.

  Malware related attacks are one of the most common forms of cyberattacks. By utilising a combination of file monitoring and pattern recognition, monitoring and logging tools can detect such malicious files within a system before they cause damage to the infected system. This test aims to demonstrate this ability of monitoring and logging tools and see if Wazuh has such capabilities.

- **Prerequisites**

  Wazuh already comes with a list of common trojan locations and signatures in the `/var/ossec/etc/shared/rootkit_trojans.txt` file, available in A.1. By combining the contents of this file with the capabilities demonstrated in the Detecting File Changes test, Wazuh should be capable of detecting changes to the system provoked by malware and trojans.

Additionally, Wazuh should be configured to check for malware and trojans every 12 hours by default on installation; this can be verified in the monitored machine's configuration file `/var/ossec/etc/ossec.conf`.

- **Test Steps**

  1. First, we will create a copy of an original system binary with the following command. For this test, we will use the binary `/usr/bin/w`. This standard command in Unix systems provides a quick summary of every user logged into the computer and its activity. But for this test, it is unimportant which binary is used.

     ```
     # cp -p /usr/bin/w /usr/bin/w.copy
     ```

  2. Next, we will replace the original system binary with the following script.

     ```
     #!/bin/bash
     echo "`date` this is evil" > /tmp/trojan_created_file
     echo 'test for /usr/bin/w trojaned file' >> /tmp/
        trojan_created_file

     #Now running original binary
     /usr/bin/w.copy
     ```

- **Test Results**

  After executing the previous steps, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. We can see the alert created in response to the trojan file, proving that Wazuh has successfully detected the suspicious binary.

Table 5.12: The truncated details of the Wazuh alert generated in response to the suspicious binary

Field Name	**Alert Details**
data.file	`/usr/bin/w`
data.title	*Trojaned version of file detected.*
full_log	`Trojaned version of file '/usr/bin/w' detected. Signature used: 'uname -a|proc\.h|bash' (Generic).`
rule.description	Host-based anomaly detection event (rootcheck).
rule.groups	ossec, rootcheck
rule.id	510
rule.level	7

5.4.2.6 Detecting Vulnerable Applications

- **Overview**

  This test aims to see if Wazuh can detect and identify vulnerable applications within the system.

  An essential function of monitoring and logging tools is to detect vulnerabilities within a system. These vulnerabilities can have many different sources, but most commonly are from applications or tools not being up to date and lacking an upgrade. This test aims to demonstrate this ability of monitoring and logging tools and see if Wazuh has such capabilities.

- **Prerequisites**

  Wazuh already has a vulnerability detector module, and it just needs to be correctly configured for the test installation within the DataPlaxe machines.

- **Test Steps**

  1. Enable the vulnerability detector module in the `/var/ossec/etc/ossec.conf` file at the Wazuh Manager. As the machines run on CentOS for this test installation, the configuration will be as follows.

```
<ossec_config>
<vulnerability-detector>
    <enabled>yes</enabled>
    <interval>5m</interval>
    <ignore_time>6h</ignore_time>
    <run_on_start>yes</run_on_start>

        [...]

    <!-- RedHat OS vulnerabilities -->
    <provider name="redhat">
    <enabled>yes</enabled>
    <os>5</os>
    <os>6</os>
    <os>7</os>
    <os>8</os>
    <os allow="Centos Linux-8">8</os>
    <update_interval>1h</update_interval>
    </provider>

        [...]
```

```
    <!-- Aggregate vulnerabilities -->
    <provider name="nvd">
    <enabled>yes</enabled>
    <update_from_year>2010</update_from_year>
    <update_interval>1h</update_interval>
    </provider>

</vulnerability-detector>
</ossec_config>
```

2. Restart the Wazuh Manager

```
# systemctl restart wazuh-manager
```

3. Configure the "syscollector" module in the `/var/ossec/etc/ossec.conf` file at the Wazuh Agent. As this test installation uses Linux machines, the configuration file will be as follows.

```
<wodle name="syscollector">
 <disabled>no</disabled>
 <interval>1h</interval>
 <scan_on_start>yes</scan_on_start>
 <hardware>yes</hardware>
 <os>yes</os>
 <network>yes</network>
 <packages>yes</packages>
 <hotfixes>yes</hotfixes>
 <ports all="no">yes</ports>
 <processes>yes</processes>
</wodle>
```

4. Restart the Wazuh Agent

```
# systemctl restart wazuh-manager
```

- **Test Results**

  After activating Wazuh's vulnerability detector module, we can access the Vulnerabilities section of the Wazuh Kibana plugin. We find that each machine of our test implementation, as they are copies of each other, have around 940 vulnerabilities. A high number was expected as these are merely test machines and haven't been updated in a while.

Vulnerabilities (940) Export formatted

Filter or search vulnerabilities

CVE	Name ↑	Version	Architecture
CVE-2012-6711	bash	4.2.46-33.el7	x86_64
CVE-2019-18276	bash	4.2.46-33.el7	x86_64
CVE-2019-9924	bash	4.2.46-33.el7	x86_64
CVE-2018-5745	bind-export-libs	32:9.11.4-9.P2.el7	x86_64
CVE-2019-6465	bind-export-libs	32:9.11.4-9.P2.el7	x86_64
CVE-2019-6477	bind-export-libs	32:9.11.4-9.P2.el7	x86_64

Figure 5.4: The abbreviated list of detected vulnerabilities within the test machine.

Additionally, each vulnerability is assigned a severity level from critical to low. This classification makes the prioritisation of fixes easier.

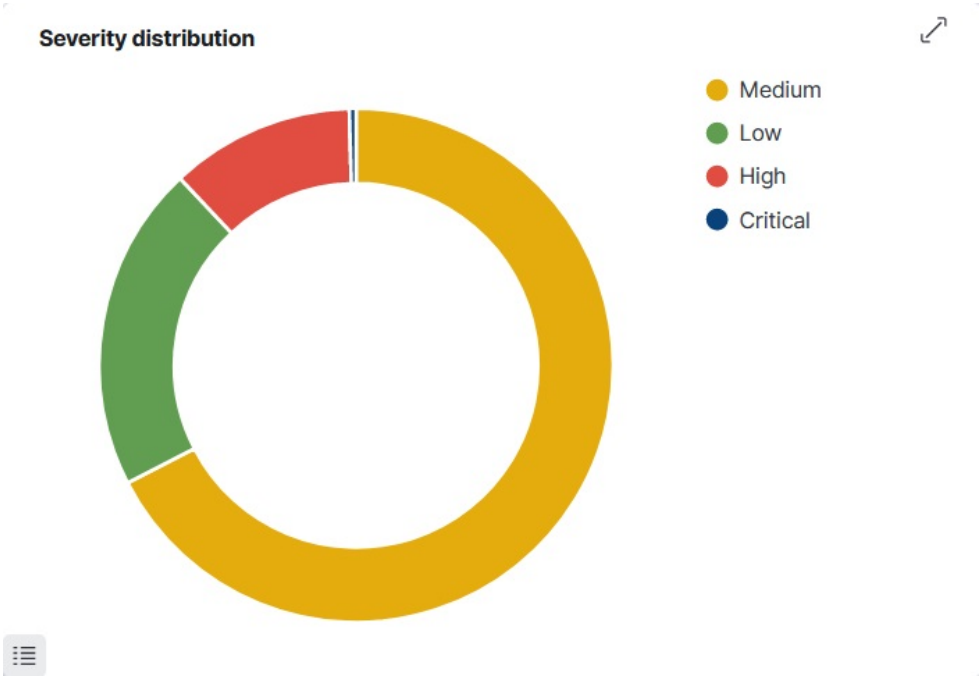

Figure 5.5: The Severity distribution of the vulnerabilities detected.

Table 5.13: The abbreviated list of detected vulnerabilities sorted with descending severity.

data.vulnerability.package.name	**data.vulnerability.cve**	**data.vulnerability.severity**
nss-tools	CVE-2021-43527	Critical
expat	CVE-2015-2716	Critical
nss-sysinit	CVE-2021-43527	Critical
nss	CVE-2021-43527	Critical
python-perf	CVE-2021-37576	High
python-perf	CVE-2020-10757	High
minio	CVE-2020-11012	High
dhclient	CVE-2021-25217	High

Each vulnerability also contains a field explaining it, which gives helpful context to developers when fixing it.

Table 5.14: The abbreviated details of an example Vulnerability

Field Name	**Vulnerability Details**
data.vulnerability.title	CVE-2021-43527 affects nss-tools
data.vulnerability.package.name	nss-tools
data.vulnerability.cve	CVE-2021-43527
data.vulnerability.severity	Critical
data.vulnerability.rationale	NSS (Network Security Services) versions prior to 3.73 or 3.68.1 ESR are vulnerable to a heap overflow when handling DER-encoded DSA or RSA-PSS signatures. Applications using NSS for handling signatures encoded within CMS, S/MIME, PKCS #7, or PKCS #12 are likely to be impacted. Applications using NSS for certificate validation or other TLS, X.509, OCSP or CRL functionality may be impacted, depending on how they configure NSS. *Note: This vulnerability does NOT impact Mozilla Firefox.* However, email clients and PDF viewers that use NSS for signature verification, such as Thunderbird, LibreOffice, Evolution and Evince are believed to be impacted. This vulnerability affects NSS < 3.73 and NSS < 3.68.1.

It is possible to see that Wazuh can detect vulnerabilities within a system.

5.4.2.7 Detecting Unauthorised Processes

- **Overview**

  This test aims to see if Wazuh can detect the execution of black-listed processes.

  Monitoring running processes on a system is a valuable capability of monitoring and logging tools. This capability alerts when crucial processes or functions stop unexpectedly and identifies unknown or unwanted ones suddenly starting within the system. This test aims to demonstrate this ability of monitoring and logging tools and see if Wazuh has such capabilities.

- **Prerequisites**

  For this test to occur, it is necessary to configure the Wazuh agent to obtain a list of currently running processes periodically. That configuration can be done by adding the following code block to the `<localfile>` section of the `/var/ossec/etc/ossec.conf` file of the monitored machine.

```
<ossec_config>
    <localfile>
        <log_format>full_command</log_format>
        <alias>process list</alias>
        <command>ps -e -o pid,uname,command</command>
        <frequency>30</frequency>
    </localfile>
</ossec_config>
```

  This code block configures the Wazuh agent to run `ps -e -o pid,uname,command` to obtain the running process list every 30 seconds.

  As a reminder, it is necessary to restart the Wazuh agent after adding the code block.

```
# systemctl restart wazuh-agent
```

  The Wazuh manager must also be configured to receive the running process list correctly. This configuration can be done by adding the following rule in the `var/ossec/etc/rules/local_rules.xml` file of the Wazuh Manager.

```
<group name="ossec,">
    <rule id="100050" level="0">
        <if_sid>530</if_sid>
        <match>^ossec: output: 'process list'</match>
        <description>List of running processes.</description>
        <group>process_monitor,</group>
    </rule>
</group>
```

This rule level is marked as zero, meaning that the collection of running processes will not trigger an alert. Additionally, the `<match>` section pairs with the configuration block added to the Wazuh Agent.

Additionally, it is necessary to choose a process to black-list. In the case of this test, we will black-list Netcat, a standard computer networking utility that allows for port scanning and port listening. Netcat can be easily installed with the command.

```
# yum install nmap-ncat
```

- **Test Steps**

  1. First, it is necessary to create a rule that triggers with the execution of Netcat. This rule shall be a child to the process list rule, and, as such, it shall be added in the `var/ossec/etc/rules/local_rules.xml` file of the Wazuh Manager. An example of an implementation of this rule can be seen below.

     ```
     <rule id="100051" level="7" ignore="900">
         <if_sid>100050</if_sid>
         <match>nc -l</match>
         <description>Netcat listening for incoming connections
        .</description>
         <group>process_monitor,</group>
     </rule>
     ```

  2. Restart the Wazuh Manager

     ```
     # systemctl restart wazuh-manager
     ```

  3. Next, to trigger the created rule, it is necessary to run Netcat with the following command.

     ```
     $ nc -l 8000
     ```

- **Test Results**

  After executing the previous command, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. We can see the alert created in response to the execution of the black-listed Netcat command.

Table 5.15: The truncated details of the Wazuh alert generated in response to the black-listed Netcat Command

Field Name	Alert Details
location	process list
rule.description	Netcat listening for incoming connections.
rule.groups	ossec, process_monitor
rule.id	100051
rule.level	7

These results demonstrate that Wazuh can monitor the running processes within the system and be configured to detect unwanted changes to the list of running processes.

5.4.2.8 Exposing Rootkits and Hidden Processes

- **Overview**

  This test aims to see if Wazuh can detect rootkits that can hide from the kernel module list and hide their processes from the "`ps`" command.

  This test can be seen as a continuation of the previous one, as it demonstrated that Wazuh was capable of monitoring running processes. Still, it did so by analysing the list provided by the system. This test aims to see if Wazuh can detect processes if those have been hidden by a rootkit, a standard cover tactic used by malicious attackers.

  The rootkit Diamorphine will be installed on the machine to perform this test, as it matches the description above.

- **Prerequisites**

  The Wazuh agent is configured by default to check for rootkits and hidden processes every 12 hours. It does this by utilising system calls such as `setsid()`, `getpid()` and `kill()` that indirectly expose any processes running in the system. This function can be sped up for the sake of this test in the "`rootcheck`" and "`syscheck`" sections of the `var/ossec/etc/ossec.conf` file.

  As the Wazuh is already configured, it is just a matter of downloading and installing the Diamorphine rootkit. The steps for the installation are the following.

```
# yum -y update
# shutdown -r now
# yum -y install kernel-devel libgcc gcc git
# git clone https://github.com/wazuh/Diamorphine.git
```

- **Test Steps**

  The rootkit kernel module first needs to be loaded, which can be done with the following command.

  ```
  # insmod diamorphine.ko
  ```

  As a note, if an error occurs after executing the command, the best solution is to restart the machine and start again. Multiple restarts may be needed to perform this test, but the installation should be correct if the following sequence can be executed without error.

  1. `# lsmod | grep diamorphine`

     This command should have no output as Diamorphine is hidden until it receives the following kill command.

  2. ```
     # kill -63 509
     # lsmod | grep diamorphine
     ```

     This last command should show Diamorphine running without issue, which became visible after the kill command.

  3. ```
     # kill -63 509
     # lsmod | grep diamorphine
     ```

     This last kill command is to make Diamorphine hidden again.

  After the rootkit has been correctly loaded, it can hide processes from the "`ps`" command. The following command hides the `rsyslogd` process, a typical system utility that supports local and remote logging.

  ```
  # kill -31 $(pidof rsyslogd)
  ```

  This hiding function of Diamorphine can be checked with the following command, which should have no output.

  ```
  # ps auxw | grep rsyslog | grep -v grep
  ```

- **Test Results**

  After executing the previous steps, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. We can see the alert created in response to the Diamorphine rootkit hiding a process from the "`ps`" command, meaning that Wazuh has successfully detected the hidden processes running in the machine.

Table 5.16: The truncated details of the Wazuh alert generated in response to the usage of the Diamorphine rootkit.

Field Name	Alert Details
data.title	Process '1045' hidden from /proc.
full_log	Process '1045' hidden from /proc. Possible kernel level rootkit.
rule.description	Possible kernel level rootkit
rule.id	521
rule.level	11
rule.mitre.tactic	Defense Evasion
rule.mitre.technique	Rootkit

5.4.2.9 Monitoring Docker

- **Overview**

  Docker is a commonly used platform to deploy, ship, and run applications quickly and effectively. Developers use it because it separates the applications from the infrastructure used to develop or run them. This separation is possible because Docker allows the creation of a semi-isolated environment called a container. This container has all the application's necessities and can be easily distributed and maintained. (Docker Inc., 2022)

  Because the DataPlaxe platform utilised Docker for many of its components, it is essential to check if Wazuh can correctly log Docker related events.

- **Prerequisites**

  It is necessary to configure a Docker listener in the monitored machine to log Docker events. This configuration can be done in `var/ossec/etc/ossec.conf` with the following code block.

```
<ossec_config>
    <wodle name="docker-listener">
    <interval>10m</interval>
    <attempts>5</attempts>
    <run_on_start>yes</run_on_start>
    <disabled>no</disabled>
    </wodle>
</ossec_config>
```

  If it was necessary to configure the Docker listener, it is also required to restart the Wazuh-agent.

```
# systemctl restart wazuh-agent
```

Additionally, but somewhat obviously, it is necessary to have Docker installed on the machine. Although for the particular installation of Docker of the DataPlaxe platform, it is also required to add some python related dependencies. These libraries can be added with the following commands.

```
# pip3 install docker
# yum install python-docker
```

- **Test Steps**

  To generate the Docker related Wazuh logs, we shall

  1. Pull a Docker image, in this case, Nginx.

     ```
     $ docker stop `docker ps -a -q` && docker rm `docker ps -a -
        q`
     $ docker pull nginx
     ```

  2. Start the Docker container.

     ```
     $ docker run -d -P --name nginx_container nginx
     ```

  3. Run a Docker command.

     ```
     $ docker exec -ti nginx_container cat /etc/passwd
     $ docker exec -ti nginx_container /bin/bash
     $ exit
     ```

  4. And then Delete the container.

     ```
     $ docker stop nginx_container
     $ docker rm nginx_container
     ```

- **Test Results**

  We can then visualise the Wazuh logs in the Wazuh Kibana plugin, and in every record, the field titled "data.docker.Action" states the action it was performed.

Table 5.17: The Wazuh alerts generated in response to the Docker commands executed.

rule.description	rule.level	data.docker.Action
Container nginx_container created	3	create
Network bridge connected	3	connect
Container nginx_container started	3	start
Command launched in container nginx_container. Action: "exec_start: cat /etc/passwd"	3	exec_start: cat /etc/passwd
Started shell session in container nginx_container	5	exec_start: /bin/bash
Container nginx_container received the action: kill	7	kill
Container nginx_container received the action: die	7	die
Network bridge disconnected	4	disconnect
Container nginx_container stopped	3	stop
Container nginx_container destroyed	5	destroy

These alerts demonstrate that Wazuh can monitor Docker-related events, which was one of the leading security requirements of the DataPlaxe platform.

5.4.2.10 Auditing Commands Run by a User

- **Overview**

  This test aims to see if Wazuh can log and report commands executed by a pre-determined user.

  System users, here differentiated from platform users, can come in many forms, from admin or root users to even software-related ones. Due to their permissions, some of these system users can perform system defining actions. It may be valuable to monitor these users and confirm that their activity does not compromise the platform either because of misconfiguration or genuine compromise.

- **Prerequisites**

  To record the author of the executed commands, Wazuh needs an auditing subsystem. The Linux Audit subsystem was already used in the Detecting File Changes test. But as a reminder, this system can be installed with the following command.

  ```
  # yum install audit
  ```

  The Wazuh agent should be configured by default to read the `audit.log` file. But this can be checked in the `/var/ossec/etc/ossec.conf` file, where the following code block should be present.

  ```
  <localfile>
  ```

```
  <log_format>audit</log_format>
  <location>/var/log/audit/audit.log</location>
</localfile>
```

Additionally, it is necessary to add auditing rules to the Wazuh agent. These auditing rules contain the user id of the monitored user. They configure the Wazuh agent to report every command logged by the Linux auditing subsystem that that user executed. These rules need to be added to the `/etc/audit/rules.d/wazuh.rules` file and have the following format:

```
-a exit,always -F euid=<monitored_user_id> -F arch=b32 -S
   execve -k audit-wazuh-c
-a exit,always -F euid=<monitored_user_id> -F arch=b64 -S
   execve -k audit-wazuh-c
```

The command to load the auditing rules is the following:

```
# auditctl -R /etc/audit/rules.d/wazuh.rules
```

As a quick note, it is not recommended to use the root user for this test as the root user, being a unique user, performs many actions in the background, and it can be challenging to discern what is happening for testing purposes.

- **Test Steps**

  With the auditing rules in place, it is needed to log into the machine with the Wazuh Agent as the monitored user and perform a command. For this test, the monitored user will perform a simple `ping` to reach `www.google.com` with the following command:

  ```
  $ ping www.google.com
  ```

- **Test Results**

  After executing the previous command, we access the Wazuh Alerts in the Security Events module of the Wazuh Kibana plugin. We can see the alert created in response to the `ping` command. Every detail about the executed command is present: from the arguments used, in the “execve” fields, to the user’s id, which in the case of this test was 1003.

Table 5.18: The truncated details of the Wazuh alert generated in response to the command executed by the monitored user.

Field Name	Alert Details
data.audit.auid	1003
data.audit.command	ping
data.audit.cwd	/home/ptin_admin
data.audit.euid	1003
data.audit.exe	/usr/bin/ping
data.audit.execve.a0	ping
data.audit.execve.a1	www.google.com
data.audit.exit	0
rule.description	Audit: Command: /usr/bin/ping

These reports created by Wazuh have enough information to perform a complete picture of the activity of a system user, which could prove irreplaceable information on the aftermath of a cyber-attack.

5.4.2.11 Block a Malicious Actor

- **Overview**

  The purpose of this test is to see if Wazuh is capable of handling and blocking a known attacker. Once a bad actor has been identified, a monitoring and logging system is expected to have a certain level of an automated response. This test aims to evaluate the response capabilities of Wazuh and see if they are acceptable.

- **Prerequisites**

  For this test, the "know attacker" will try to access an Apache server running on the monitored machine. As such, this server needs to be installed and prepared.

  Additionally, the Wazuh agent needs to be configured to capture events from the Apache server. This setup can be done by adding the following block to the monitored machine's `/var/ossec/etc/ossec.conf` file.

  ```
  <localfile>
  <log_format>apache</log_format>
  <location>/var/log/httpd/access_log</location>
  </localfile>
  ```

  After changing the configuration, it is necessary to restart the Wazuh Agent.

  ```
  # systemctl restart wazuh-agent
  ```

Next, Wazuh needs a list of known bad actors to reference. The Alienvault IP reputation database (FireHOL, 2022) will be used for this test as it is compatible with Wazuh. After this list has been downloaded to the Wazuh manager, the "know attacker" IP address will be added to the list. This "know attacker" is an outside machine from the test environment. After adding the new IP address, the list is compiled into a readable format for Wazuh. The commands to perform these actions are the following:

```
# wget https://raw.githubusercontent.com/firehol/blocklist-
   ipsets/master/alienvault_reputation.ipset -O /var/ossec/etc/
   lists/alienvault_reputation.ipset
# echo "<know_attacker_ip_address>" >> /var/ossec/etc/lists/
   alienvault_reputation.ipset
# wget https://wazuh.com/resources/iplist-to-cdblist.py -O /tmp
   /iplist-to-cdblist.py
# python /tmp/iplist-to-cdblist.py /var/ossec/etc/lists/
   alienvault_reputation.ipset /var/ossec/etc/lists/blacklist-
   alienvault
# rm -rf /var/ossec/etc/lists/alienvault_reputation.ipset
# rm -rf /var/ossec/etc/lists/iplist-to-cdblist.py
# chown ossec:ossec /var/ossec/etc/lists/blacklist-alienvault
# chmod 660 /var/ossec/etc/lists/blacklist-alienvault
```

After compiling the list into a readable format for the Wazuh Manager, a custom rule is created to use the created list in the `/var/ossec/etc/rules/local_rules.xml` file. For this test, the rule has the following body:

```
<group name="attack,">
<rule id="100100" level="10">
    <if_group>web|attack|attacks</if_group>
    <list field="srcip" lookup="address_match_key">etc/lists/
   blacklist-alienvault</list>
    <description>IP address found in AlienVault reputation
   database.</description>
</rule>
</group>
```

The previous rule makes it so Wazuh detects our "know attacker", but to respond to his access, the following code block needs to be added to the `var/ossec/etc/ossec.conf` file:

```
<ossec_config>
        <ruleset>
                ...
```

```
        <list>etc/lists/blacklist-alienvault</list>
        ...
        </ruleset>
        ...
        <active-response>
                <command>firewall-drop</command>
                <location>local</location>
                <rules_id>100100</rules_id>
                <timeout>60</timeout>
        </active-response>
    ...
</ossec_config>
```

The previous code block configures Wazuh to reference the previously created list and to, upon having a match on the created rule, with id 100100, drop the connection for 60 seconds.

After changing the configuration, it is necessary to restart the Wazuh Manager.

```
# systemctl restart wazuh-manager
```

- **Test Steps**

  With the Wazuh Manager configured to timeout for 60 seconds every connection attempt from the IP addresses contained in the created list, to perform this test, the “know attacker” machine will attempt to connect to the Apache server on the monitored machine through a simple web browser.

- **Test Results**

  After attempting to connect to the monitored machine, in the Security Events module of the Kibana plugin, it is possible to see the alerts created by Wazuh. In those, it is possible to see the correct identification of the attacker and the automated response that was configured.

Table 5.19: The truncated details of the Wazuh alert generated in response to the "know attacker" access.

Field Name	**Alert Details**
data.id	404
data.protocol	GET
data.srcip	*know_attacker_ip_address*
rule.description	IP address found in AlienVault reputation database.
rule.firedtimes	32
rule.groups	attack
rule.id	100100
rule.level	10

Table 5.20: The truncated details of the Wazuh alert generated from the automated response.

Field Name	**Alert Details**
data.parameters.alert.data.id	404
data.parameters.alert.data.protocol	GET
data.parameters.alert.data.srcip	*know_attacker_ip_address*
data.parameters.alert.rule.description	IP address found in AlienVault reputation database.
data.parameters.alert.rule.firedtimes	33
data.parameters.alert.rule.groups	attack
data.parameters.alert.rule.id	100100
data.parameters.alert.rule.level	10
data.parameters.program	active-response/bin/firewall-drop
rule.description	Host Blocked by firewall-drop Active Response
rule.firedtimes	33
rule.groups	ossec, active_response
rule.id	651
rule.level	3

5.4.2.12 Dealing with a Log Flood

- **Overview**

  The purpose of this test is to see if Wazuh can handle a heavy number of logs. A log flood may happen for multiple reasons, from misconfigured applications to malicious actors' activity such as DDOS attacks, and it's important to discern if Wazuh does not overtax the network on such an event.

  This test will lower the event transmission speed of a Wazuh agent, and then it will generate a more significant number of logs than that. After that, it will observe the actions performed by Wazuh while experiencing a log flood.

- **Prerequisites**

  To perform this test, the Wazuh agent needs to be configured to produce fewer logs than what is advised in a real situation. This change can be done in the `/var/ossec/etc/ossec.conf` file in the `<client_buffer>` section by changing the `<events_per_second>` value. For this test, this value will be set to 50 events per second (EPS), making the above section look like this:

```
<client_buffer>
    <!-- Agent buffer options -->
    <disabled>no</disabled>
    <queue_size>5000</queue_size>
    <events_per_second>50</events_per_second>
```

```
</client_buffer>
```

After changing the configuration, it is necessary to restart the Wazuh Agent.

```
# systemctl restart wazuh-agent
```

Additionally, to see the full extent of the Wazuh behaviour during this test, it is necessary to lower the Wazuh manager alert severity threshold. This change is needed because some alerts from this test only have a severity level of 2, and Wazuh manager is configured by default to only report alerts with severity levels above or equal to 3. This configuration can be done in file `/var/ossec/etc/ossec.conf` of the Wazuh manager, by changing the `<log_alert_level>` field from 3 to 1, making the `<alerts>` section look like the following.

```
<alerts>
    <log_alert_level>1</log_alert_level>
    <email_alert_level>12</email_alert_level>
</alerts>
```

After changing the configuration, it is necessary to restart the Wazuh manager.

```
# systemctl restart wazuh-manager
```

Furthermore, this test utilises Netcat to create the logs for the log flooding, so this tool needs to be installed in the machine with the Wazuh agent.

```
# yum install nmap-ncat
```

- **Test Steps**

  This test needs to create logs above the 50 EPS configured, and for that, the following script will be used. This script utilises Netcat to write logs directly into the Wazuh agent internal socket, registering 10000 log entries faster than 50 EPS.

```
#!/bin/bash
for i in {1..10000}
do
  echo -n "1:floodtest:Feb  3 03:08:47 linux-agent centos:
   fatal firehose $i" | ncat -Uu /var/ossec/queue/sockets/queue
  echo -n "."
done
```

  Having created the above script, named `makeflood`, it is just a matter of making the script executable and running it to perform this test.

```
# chmod 700 <path_to_script>/makeflood
# makeflood
```

- **Test Results**

  After executing the script, it is possible to see Wazuh's behaviour within the Security Events module of Kibana.

  The following graph aggregates the alerts created by the test script into one second periods. It is possible to confirm that the Wazuh agent does not transmit more than 50 alerts per second. Although, due to this limitation, it has discarded and lost some alerts, as it only captured 5264 out of 10000. But this is an acceptable compromise, as the alternative is to run the risk of overtaxing the network with redundant data and stopping activity flows of the DataPlaxe system.

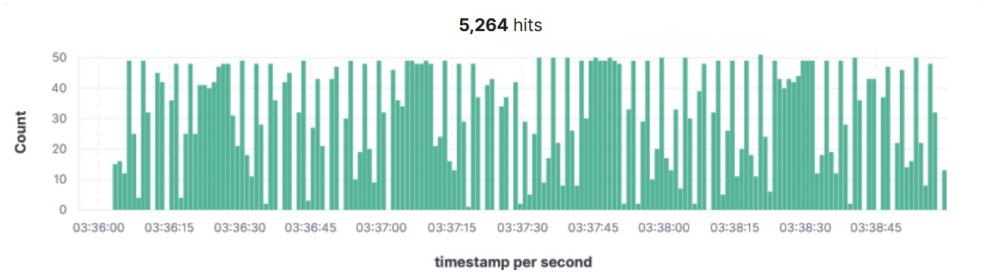

Figure 5.6: The graph showing the distribution of the logs generated by the log flood script.

Table 5.21: The truncated details of the Wazuh alert generated from the log flood script.

Field Name	**Alert Details**
full_log	Feb 3 03:08:47 linux-agent centos: fatal firehose 9995
location	floodtest
rule.firedtimes	10,662
rule.id	1002
rule.level	2

The Wazuh agent also sends alerts notifying that it is experiencing a log flood. The details of these can be seen in the following table.

Table 5.22: The Wazuh alerts generated from the filling of the Agent's buffer due to the log flooding script.

rule.description	rule.level	full_log	data.level
Agent event queue is 90% full.	7	Agent buffer: '90%'.	90%
Agent event queue is full. Events may be lost.	9	Agent buffer: 'full'.	full
Agent event queue is flooded. Check the agent configuration.	12	Agent buffer: 'flooded'.	flooded
Agent event queue is back to normal load.	3	Agent buffer: 'normal'.	normal

As it is possible to see from the obtained results, Wazuh is indeed capable of handling a heavy amount of logs in a way that does not overtax the internal network of the DataPlaxe platform.

5.5 Implementation

Having taken a look at Wazuh and its capabilities, this section will describe the ideal implementation of this tool in the context of the DataPlaxe platform.

Considering the purpose of monitoring all of the platform's components, Wazuh Agents must be installed on every single machine of the Data Ingestion and Access Layer cluster and the Platform Service cluster.

Next, the traffic generated by the agents should be funnelled through the Management Network onto the Wazuh Manager installed in the Administration and Service Monitoring cluster machines.

The Administration and Service Monitoring cluster machines should be divided into two subgroups. The first group will have multiple instances of Wazuh Managers, which adds redundancy in case of technical failure, and Filebeat. These managers should be configured as a sub-cluster to perform load balancing and send their logs via Filebeat to the second group. The second group will have ElasticSearch instances in a sub-cluster setup to perform load balancing. The Kibana server can be installed onto a single device as long as it is configured to communicate with the managers and the ElasticSearch sub-clusters correctly.

The Wazuh Agents should be configured to collect logs from the OpenResty server in the Data Ingestion and Access Layer cluster. As OpenResty is based on Nginx (Nginx Inc, 2022), this configuration can be done in the `/var/ossec/etc/ossec.conf` file of the Wazuh agents present in the edge nodes by adding the following code block.

```
<ossec_config>
    <localfile>
        <log_format>apache</log_format>
        <location>/var/log/nginx/access.log</location>
    </localfile>
```

```
    <localfile>
        <log_format>apache</log_format>
        <location>/var/log/nginx/error.log</location>
    </localfile>
</ossec_config>
```

And in the Platform Service cluster, the agents need to be configured to capture Docker related events. This configuration can be done in the `var/ossec/etc/ossec.conf` file with the following code block.

```
<ossec_config>
    <wodle name="docker-listener">
    <interval>10m</interval>
    <attempts>5</attempts>
    <run_on_start>yes</run_on_start>
    <disabled>no</disabled>
    </wodle>
</ossec_config>
```

Chapter 6

Conclusions and Future Work

6.1 Conclusion

This dissertation aimed to find a way of detecting, preventing and securing systems against cyber-attacks. The concept of monitoring and logging tools was discovered and explored by researching books and the words of authority figures in the cyber security space. By utilising the DataPlaxe platform, a multi-tenant service by Altice Labs, as a case study and a testing ground for one of these tools, namely Wazuh, it seeks to validate if monitoring and logging tools are indeed capable of detecting cyber-security attacks. Secondarily it also aimed to provide the DataPlaxe platform with added security and resistance to cyberattacks by again utilising the concept of monitoring and logging tools.

Most of the current literature available on monitoring and logging tools is, like most cyber security literature, mainly focused on guidelines and best practices and often from an organisation perspective and level. This focus makes it difficult to grasp these concepts from a developer and implementation level. As such, this dissertation exposed the monitoring and logging concepts from a more ground-level approach in the hopes of filling this perspective vacuum regarding this subject.

But even though the usage of monitoring and logging tools is often lacking in modern systems, these tools do populate the market should one look for them. And even though this dissertation only deeply investigated one of them, the results from the tests performed were entirely satisfactory. These tools can detect cyberattacks, as made clear by the brute-force, SQL injection, shellshock and malware tests. Additionally, these tools can collect valuable information from the happenings within the system they are implemented, as demonstrated by the detecting file changes test, the auditing commands from a user test and the Docker monitoring test. And, if correctly configured, they could respond automatically, faster than any human, at known and common forms of attacks as demonstrated by the blocking a malicious actor test. They serve as a great line of defence against cyberattacks that should be used whenever possible.

Regarding the secondary goal of this dissertation of improving the security of the Dataplaxe platform, the selected monitoring and logging tool, Wazuh, performed admirably, as stated previ-

ously, as general-purpose and all-encompassing monitoring and logging tool. And it was even able to meet additional requirements specific to the DataPlaxe platform as it demonstrates the capabilities to monitor Docker activity and avoid overtaxing the platform's internal networks in the event of a log flood. It has become clear that the security of the DataPlaxe platform would significantly improve with the implementation of a monitoring and logging tool, and Wazuh has proven to be capable of meeting the expectation of such a tool within the platform.

6.2 Further Work

This dissertation answered how to detect cyberattacks with monitoring and logging tools. And although that is a helpful answer and a topic that needs more general awareness. There could be other ways to detect cyberattacks and many other forms to prevent them and mitigate their damages should they occur, which should also be studied and highlighted.

Additionally, further exploration of the monitoring and logging tools available in the market is needed. Many of them are very specialised in what they do. This dissertation has glanced chiefly over them, preferring instead to look at a more general all-purpose tool.

For the DataPlaxe platform, there is the implementation of Wazuh on the platform as described in the sub-chapter Implementation. And its configuration and adjusting when the platform is made available to the public.

Additionally, the concept of integrating Wazuh with a network intrusion detecting system (IDS) was untouched by this dissertation. Wazuh can be combined with Suricata (OISF, 2022) an open-source network IDS, and should Suricata be compatible with the DataPlaxe platform, it could allow for better monitoring of the platform's internal networks and their network traffic.

References

AppDynamics. Logging vs Monitoring: Best Practices for Integration | AppDynamics, 2021. URL https://www.appdynamics.com/product/how-it-works/application-analytics/log-analytics/monitoring-vs-logging-best-practices.

AT&T Business. OSSIM: The Open Source SIEM | AlienVault, 2022. URL https://cybersecurity.att.com/products/ossim.

Awake Security. Network Intrusion Definition and Examples, 2021. URL https://awakesecurity.com/glossary/network-intrusion/.

Kristian Beckers. *Pattern and security requirements: Engineering-based establishment of security standards*. Springer International Publishing, jan 2015. ISBN 9783319166643. doi: 10.1007/978-3-319-16664-3. URL www.springer.com.

Rebecca M Blank and Patrick D Gallagher. Guide for Conducting Risk Assessments, sep 2012. URL https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final.

Wen Lin Cheng, Ting Che Chuang, Chien Wen Yang, Yueh Hsien Lin, Min Liu, and Chuan Yin. An integrated security monitoring system for digital service network devices. In *19th Asia-Pacific Network Operations and Management Symposium: Managing a World of Things, APNOMS 2017*, pages 118–122, Seoul, South Korea, nov 2017. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/APNOMS.2017.8094189.

Cisco. Snort - Network Intrusion Detection & Prevention System, 2022. URL https://www.snort.org/.

João Marques Correia. *Execução segura com contentores*. M.s. thesis, UA, Aveiro, Portugal, jul 2020. URL https://ria.ua.pt/handle/10773/29674.

Jason Creasey. *Cyber Security Monitoring and Logging Guide DTP notes A Good Tip Cyber Security Monitoring and Logging Guide*. CREST, 2015. URL http://www.crest-approved.org.

Docker Inc. Docker overview | Docker Documentation, 2022. URL https://docs.docker.com/get-started/overview/.

Nuno Filipe Lopes da Costa Duarte. *Segurança contra intrusão em redes informáticas*. M.s. thesis, Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto, Porto, Portugal, 2008. URL https://recipp.ipp.pt/handle/10400.22/1883.

EC-Council. What Is a Denial-Of-Service (DoS) Attack, 2021. URL https://www.eccouncil.org/what-is-a-denial-of-service-dos-attack/.

Elastic NV. Free and Open Search: The Creators of Elasticsearch, ELK & Kibana | Elastic, 2022. URL https://www.elastic.co/.

FireHOL. alienvault_reputation by Alien Vault, reputation IPs list, at FireHOL IP Lists, 2022. URL https://iplists.firehol.org/?ipset=alienvault_reputation.

Stephen Groat, Joseph Tront, and Randy Marchany. Advancing the defense in depth model. In *Proceedings - 2012 7th International Conference on System of Systems Engineering, SoSE 2012*, pages 285–290, Genova, Italy, jul 2012. IEEE. ISBN 9781467329750. doi: 10.1109/SYSOSE.2012.6384127.

IBM Security. IBM Security QRadar SIEM | IBM, 2022. URL https://www.ibm.com/qradar/security-qradar-siem.

Intersoft Consulting. General Data Protection Regulation (GDPR) – Official Legal Text, 2021. URL https://gdpr-info.eu/.

Kali Linux. Hydra | Kali Linux Tools, 2021. URL https://www.kali.org/tools/hydra/.

Kaspersky. Kaspersky IT Encyclopedia, 2021. URL https://encyclopedia.kaspersky.com/.

Liu Cricket and Albitz Paul. *DNS and BIND, 5th Edition | Cricket Liu, Paul Albitz | download*. O'Reilly, fifth edition, 2006. ISBN 9780596100575. URL https://pt1lib.org/book/3677472/5fa27c.

LogRhythm. SIEM Solution | Security Information & Event Management | LogRhythm, 2022. URL https://logrhythm.com/solutions/security/siem/.

Metabase. Metabase, 2022. URL https://www.metabase.com/.

Inc MinIO. MinIO | High Performance, Kubernetes Native Object Storage, 2022. URL https://min.io/.

Ferdy Mulyadi, Leela Aditya Annam, Ridnarong Promya, and Chalermpol Charnsripinyo. Implementing Dockerized Elastic Stack for Security Information and Event Management. In *InCIT 2020 - 5th International Conference on Information Technology*, pages 243–248, Chonburi, Thailand, oct 2020. Institute of Electrical and Electronics Engineers Inc. ISBN 9781728166940. doi: 10.1109/INCIT50588.2020.9310950.

Nginx Inc. Advanced Load Balancer, Web Server, & Reverse Proxy - NGINX, 2022. URL https://www.nginx.com/.

Michael Nieles, Kelley L. Dempsey, and Victoria Y. Pillitteri. *An Introduction to Information Security*. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, Gaithersburg, MD, jun 2017. doi: 10.6028/NIST.SP.800-12R1. URL https://www.nist.gov/publications/introduction-information-security.

OISF. Home - Suricata, 2022. URL https://suricata.io/.

Rui Miguel Almeida Oliveira. *Analysis of Intrusion Detection Log Data on a Scalable Environment*. M.s. thesis, FEUP, UP, Porto, Portugal, aug 2020. URL https://repositorio-aberto.up.pt/handle/10216/128588.

OpenResty Inc. OpenResty® - Official Site, 2022. URL https://openresty.org/en/.

OWASP Foundation. OWASP Top Ten Web Application Security Risks | OWASP, 2021a. URL https://owasp.org/www-project-top-ten/.

OWASP Foundation. OWASP Top 10:2021, 2021b. URL https://owasp.org/Top10/.

OWASP Foundation. Logging - OWASP Cheat Sheet Series, 2021c. URL https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html.

Tao Qin, Chao He, Hezhi Jiang, and Ruoya Chen. Behavior rhythm: An effective model for massive logs characterizing and security monitoring in cloud. In *2018 IEEE Conference on Communications and Network Security, CNS 2018*, Beijing, China, aug 2018. Institute of Electrical and Electronics Engineers Inc. ISBN 9781538645864. doi: 10.1109/CNS.2018.8433138.

Quadrant Information Security. Open Source - Quadrant Information Security, 2022. URL https://quadrantsec.com/sagan_log_analysis_engine/.

Rapid7. Common Types of Cybersecurity Attacks, 2021. URL https://www.rapid7.com/fundamentals/types-of-attacks/.

Fanny Rivera-Ortiz and Liliana Pasquale. Automated Modelling of Security Incidents to represent Logging Requirements in Software Systems. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, New York, NY, USA, 2020. ACM. ISBN 9781450388337. doi: 10.1145/3407023. URL https://doi.org/10.1145/3407023.3407081.

RSA Security LLC. NetWitness Platform – See Everything, Fear Nothing, 2022. URL https://www.netwitness.com/.

Aneela Safdar, Hanif Durad, and Masoom Alam. Design and implementation of real-time visualization tool for network security monitoring. In *Proceedings of 2018 15th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2018*, volume 2018-Janua, pages 477–483, Islamabad, Pakistan, mar 2018. Institute of Electrical and Electronics Engineers Inc. ISBN 9781538635643. doi: 10.1109/IBCAST.2018.8312267.

Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, aug 1975. ISSN 15582256. doi: 10.1109/PROC.1975.9939.

Joanna C.S. Santos, Katy Tarrit, and Mehdi Mirakhorli. A Catalog of Security Architecture Weaknesses. In *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, pages 220–223. Institute of Electrical and Electronics Engineers Inc., apr 2017. ISBN 9781509047932. doi: 10.1109/ICSAW.2017.25. URL https://www.researchgate.net/publication/317929320_A_Catalog_of_Security_Architecture_Weaknesses.

Muhammad I.H. Sukmana, Kennedy A. Torkura, Feng Cheng, Christoph Meinel, and Hendrik Graupner. Unified logging system for monitoring multiple cloud storage providers in cloud

storage broker. In *International Conference on Information Networking*, volume 2018-Janua, pages 44–49, Chiang Mai, Thailand, apr 2018. IEEE Computer Society. ISBN 9781538622896. doi: 10.1109/ICOIN.2018.8343081.

Sumo Logic. Cloud Log Management, Monitoring, SIEM Tools | Sumo Logic, 2022. URL https://www.sumologic.com/.

Jan Svacina, Jackson Raffety, Connor Woodahl, Brooklynn Stone, Tomas Cerny, Miroslav Bures, Dongwan Shin, Karel Frajtak, and Pavel Tisnovsky. On Vulnerability and Security Log analysis: A Systematic Literature Review on Recent Trends. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, pages 175–180, New York, NY, USA, 2020. ACM. ISBN 9781450380256. doi: 10.1145/3400286. URL https://doi.org/10.1145/3400286.3418261.

Vitsunee Teeraratchakarn and Yachai Limpiyakorn. Exploring Network Vulnerabilities for Corporate Security Operations. In *Lecture Notes in Electrical Engineering*, volume 621, pages 341–351. Springer, Singapore, 2020. ISBN 9789811514647. doi: 10.1007/978-981-15-1465-4_35. URL https://link.springer.com/chapter/10.1007/978-981-15-1465-4_35.

The Apache Software Foundation. Apache Hive TM, 2022. URL https://hive.apache.org/.

The PostgreSQL Global Development Group. PostgreSQL: The world's most advanced open source database, 2022. URL https://www.postgresql.org/.

The Presto Foundation. Presto | Distributed SQL Query Engine for Big Data, 2022. URL https://prestodb.io/.

Eric. Vyncke and Christopher. Paggen. *LAN switch security : what hackers know about your switches*. Cisco Press, first edition, 2007. ISBN 978-1-58705-256-9.

Lina Wang, Ying Ren -, Steven Lam, Warren Dodd, Kelly Skinner, Al , Yong Sun, and Haiwen Wang. Intelligent Computer Security Monitoring Information Network Analysis. *IOP Conference Series: Materials Science and Engineering*, 612(4):042042, oct 2019. ISSN 1757-899X. doi: 10.1088/1757-899X/612/4/042042. URL https://iopscience.iop.org/article/10.1088/1757-899X/612/4/042042https://iopscience.iop.org/article/10.1088/1757-899X/612/4/042042/meta.

Yun Wang and Qianhuizhi Zheng. A Logging Overhead Optimization Method Based on Anomaly Detection Model. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12634 LNCS, pages 349–359. Springer, Cham, dec 2020. ISBN 9783030706258. doi: 10.1007/978-3-030-70626-5_37. URL https://link.springer.com/chapter/10.1007/978-3-030-70626-5_37.

Wazuh Inc. Rules classification - Ruleset · Wazuh documentation, 2022. URL https://documentation.wazuh.com/current/user-manual/ruleset/rules-classification.html.

Wazuh Inc. Wazuh · The Open Source Security Platform, 2022. URL https://wazuh.com/.

Appendix A

Wazuh Configuration Files

This section contains the Wazuh configuration files relevant to the tests performed.

A.1 File Containing the Trojan Signatures

```
# Copyright (C) 2015-2020, Wazuh Inc.
#
# This program is a free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation
#
# rootkit_trojans.txt, (C) Daniel B. Cid
#
# Imported from the rootcheck project.
# Some entries taken from the chkrootkit project.
#
# Blank lines and lines starting with '#' are ignored.
#
# Each line must be in the following format:
# file_name !string_to_search!Description

# Common binaries and public trojan entries
ls          !bash|^/bin/sh|dev/[^clu]|\.tmp/lsfile|duarawkz|/prof|/
   security|file\.h!
env         !bash|^/bin/sh|file\.h|proc\.h|/dev/|^/bin/.*sh!
echo        !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cl]|^/bin/.*sh!
chown       !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cl]|^/bin/.*sh!
chmod       !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cl]|^/bin/.*sh!
```

```
chgrp       !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cl]|^/bin/.*sh!
cat         !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cl]|^/bin/.*sh!
bash        !proc\.h|/dev/[0-9]|/dev/[hijkz]!
sh          !proc\.h|/dev/[0-9]|/dev/[hijkz]!
uname       !bash|^/bin/sh|file\.h|proc\.h|^/bin/.*sh!
date        !bash|^/bin/sh|file\.h|proc\.h|/dev/[^cln]|^/bin/.*sh!
du          !w0rm|/prof|file\.h!
df          !bash|^/bin/sh|file\.h|proc\.h|/dev/[^clurdv]|^/bin/.*sh
   !
login       !elite|SucKIT|xlogin|vejeta|porcao|lets_log|sukasuk!
passwd      !bash|file\.h|proc\.h|/dev/ttyo|/dev/[A-Z]|/dev/[b-s,
   uvxz]!
mingetty    !bash|Dimensioni|pacchetto!
chfn        !bash|file\.h|proc\.h|/dev/ttyo|/dev/[A-Z]|/dev/[a-s,
   uvxz]!
chsh        !bash|file\.h|proc\.h|/dev/ttyo|/dev/[A-Z]|/dev/[a-s,
   uvxz]!
mail        !bash|file\.h|proc\.h|/dev/[^nu]!
su          !/dev/[d-s,abuvxz]|/dev/[A-D]|/dev/[F-Z]|/dev/[0-9]|
   satori|vejeta|conf\.inv!
sudo        !satori|vejeta|conf\.inv!
crond       !/dev/[^nt]|bash!
gpm         !bash|mingetty!
ifconfig    !bash|^/bin/sh|/dev/tux|session.null|/dev/[^cludisopt]!
diff        !bash|^/bin/sh|file\.h|proc\.h|/dev/[^n]|^/bin/.*sh!
md5sum      !bash|^/bin/sh|file\.h|proc\.h|/dev/|^/bin/.*sh!
hdparm      !bash|/dev/ida!
ldd         !/dev/[^n]|proc\.h|libshow.so|libproc.a!

# Trojan entries for troubleshooting binaries
grep        !bash|givemer!
egrep       !bash|^/bin/sh|file\.h|proc\.h|/dev/|^/bin/.*sh!
find        !bash|/dev/[^tnlcs]|/prof|/home/virus|file\.h!
lsof        !/prof|/dev/[^apcmnfk]|proc\.h|bash|^/bin/sh|/dev/ttyo|/
   dev/ttyp!
netstat     !bash|^/bin/sh|/dev/[^aik]|/prof|grep|addr\.h!
top         !/dev/[^npi3st%]|proc\.h|/prof/!
ps          !/dev/ttyo|\.1proc|proc\.h|bash|^/bin/sh!
tcpdump     !bash|^/bin/sh|file\.h|proc\.h|/dev/[^bu]|^/bin/.*sh!
pidof       !bash|^/bin/sh|file\.h|proc\.h|/dev/[^f]|^/bin/.*sh!
```

```
fuser       !bash|^/bin/sh|file\.h|proc\.h|/dev/[a-dtz]|^/bin/.*sh!
w           !uname -a|proc\.h|bash!

# Trojan entries for common daemons
sendmail    !bash|fuck!
named       !bash|blah|/dev/[0-9]|^/bin/sh!
inetd       !bash|^/bin/sh|file\.h|proc\.h|/dev/[^un%]|^/bin/.*sh!
apachectl   !bash|^/bin/sh|file\.h|proc\.h|/dev/[^n]|^/bin/.*sh!
sshd        !check_global_passwd|panasonic|satori|vejeta|\.ark|/hash
   \.zk|bash|/dev[a-s]|/dev[A-Z]/!
syslogd     !bash|/usr/lib/pt07|/dev/[^cln]]|syslogs\.h|proc\.h!
xinetd      !bash|file\.h|proc\.h!
in.telnetd  !cterm100|vt350|VT100|ansi-term|bash|^/bin/sh|/dev[A-R
   ]|/dev/[a-z]/!
in.fingerd  !bash|^/bin/sh|cterm100|/dev/!
identd      !bash|^/bin/sh|file\.h|proc\.h|/dev/[^n]|^/bin/.*sh!
init        !bash|/dev/h
tcpd        !bash|proc\.h|p1r0c4|hack|/dev/[^n]!
rlogin      !p1r0c4|r00t|bash|/dev/[^nt]!

# Kill trojan
killall     !/dev/[^t%]|proc\.h|bash|tmp!
kill        !/dev/[ab,d-k,m-z]|/dev/[F-Z]|/dev/[A-D]|/dev/[0-9]|proc
   \.h|bash|tmp!

# Rootkit entries
/etc/rc.d/rc.sysinit    !enyelkmHIDE! enye-sec Rootkit

# ZK rootkit (http://honeyblog.org/junkyard/reports/redhat-
   compromise2.pdf)
/etc/sysconfig/console/load.zk   !/bin/sh! ZK rootkit
/etc/sysconfig/console/load.zk   !usr/bin/run! ZK rootkit

# Modified /etc/hosts entries
# Idea taken from:
# http://blog.tenablesecurity.com/2006/12/detecting_compr.html
# http://www.sophos.com/security/analyses/trojbagledll.html
# http://www.f-secure.com/v-descs/fantibag_b.shtml
/etc/hosts  !^[^#]*avp\.ch!Anti-virus site on the hosts file
/etc/hosts  !^[^#]*avp\.ru!Anti-virus site on the hosts file
```

```
/etc/hosts  !^[^#]*awaps\.net! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*ca\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*mcafee\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*microsoft\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*f-secure\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*sophos\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*symantec\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*my-etrust\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*nai\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*networkassociates\.com! Anti-virus site on the
   hosts file
/etc/hosts  !^[^#]*viruslist\.ru! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*kaspersky! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*symanteclive update\.com! Anti-virus site on the
   hosts file
/etc/hosts  !^[^#]*grisoft\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*clamav\.net! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*bitdefender\.com! Anti-virus site on the hosts
   file
/etc/hosts  !^[^#]*antivirus\.com! Anti-virus site on the hosts file
/etc/hosts  !^[^#]*sans\.org! Security site on the hosts file
```