escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Eng.ª Informática – Cibersegurança e Informática Forense

APPLICATION SECURITY TESTING

TOMÁS AVILA WELLING

Leiria, March of 2022

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Eng.ª Informática – Cibersegurança e Informática Forense

APPLICATION SECURITY TESTING

TOMÁS AVILA WELLING
Number: 2190381

Internship carried out under the guidance of Professor PhD Marco António de Oliveira Monteiro (marco.monteiro@ipleiria.pt) and under the supervision of Engineer Marco Cova (mc@voidsoftware.com).

Leiria, March of 2022

RESUMO

Este relatório descreve o trabalho que foi feito durante todo o estágio no software VOID para terminar o Mestrado em Cibersegurança e Informatica Forense. Durante este estágio foram implementados dois projetos principais: o primeiro teve a ver com o desenvolvimento de uma nova interface gráfica para o sistema Wazuh e a modificação da sua arquitetura através da substituição dos componentes que pertencem à Elastic stack; o segundo esteve relacionado com a instalação do sistema Wazuh dentro da sede da VOID para monitorizar alguns dos servidores que são utilizados para os projetos em desenvolvimento. Os principais objetivos destes projetos eram aprender sobre o sistema Wazuh, desenvolver uma interface gráfica para este sistema e configurar e utilizar o sistema Wazuh para monitorizar os hosts que pertencem à organização. No final deste estágio, foi desenvolvido uma interface gráfica e o sistema Wazuh foi implantado e está atualmente a funcionar dentro da organização VOID.

ABSTRACT

This report describes the work that was done throughout the internship at VOID software to finish the Master's degree in Cybersecurity and Digital Forensics. During this internship two main projects were implemented: the first one had to do with the development of a new Graphical User Interface for the Wazuh system and the modification of its architecture by replacing the components that belong to the Elastic stack; the second one was related to setting up the Wazuh system within the VOID headquarters to monitor some of the servers which are used for the projects in development. The main goals of these projects were to learn about the Wazuh system, develop a Graphical User Interface for this system and to setup and use the Wazuh system to monitor hosts that belong to the organisation. By the end of this internship a Graphical User Interface (GUI) was developed and the Wazuh system was deployed and is currently working within the VOID organisation.

INDEX

LIST OF FIGURES

LIST OF LISTINGS

LIST OF ACRONYMS

AD	Anomaly-based Detection.
AMI	Amazon Machine Image.
AV	Anti Virus.
AWS	Amazon Web Services.
CA	Certification Authority.
CDB	Constant Database.
CIRT	Computer Incidence Response Team.
CIS	Center of Internet Security.
CSS	Cascading Style Sheets.
CSV	Comma Separated Values.
DOM	Document Object Model.
EDR	Endpoint Detection and Response.
ELK Stack	Elastic Stack.
EP	Endpoint Protection.
EPP	Endpoint Protection Platform.
EU	European Union.
FDD	Feature Driven Development.
GDPR	General Data Protection Regulation.
GPG	Gnu Privacy Guard.
GUI	Graphical User Interface.

HIDS	Host-based IDS.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
IDE	Integrated Development Environment.
IDS	Intrusion Detection System.
IOC	Indicator of Compromise.
IoT	Internet of Things.
IR	Incidence Response.
IT	Information Technology.
JFX	JavaFX.
JS	JavaScript.
JWT	JSON Web Token.
KQL	Kibana Query Language.
MIDS	Mixed IDS.
MVVM	Model-View-ViewModel.
NAT	Network Address Translation.
NBA	Network Behavior Analysis.
NGAV	Nex Generation Anti Virus.
NGEP	Next Generation Endpoint Protection.
NIDS	Network-based IDS.
NIST	National Institute of Standards and Technology.
OSSEC	Open Source HIDS Security.
OVA	Open Virtual Appliance.

PC	Personal Computer.
PHI	Protected Health Information.
SANS	SysAdmin, Audit, Network, and Security.
SCA	Security Configuration Assessment.
SD	Signature-based Detection.
SIEM	Security Information and Event Management.
SPA	Stateful Protocol Analysis.
UI	User Interface.
UX	User Experience.
VM	Virtual Machine.
WIDS	Wireless-based IDS.
WPF	Windows Presentation Foundation.
XAML	Extensible Application Markup Language.
XDR	Cross-Layered Detection and Response.

1 INTRODUCTION

In concordance with the requirements to complete the Master's degree in Cybersecurity and Digital Forensics at the Polytechnic of Leiria, each student is required to complete one of three options: Master's thesis; Master's dissertation; or an internship. By these constraints, it was decided to fulfill this requirement by carrying out an internship within the VOID Software organisation. As well as fulfilling the requirements for the Master's degree, this internship will also provide growth within personal, technical and professional areas.

In this document one can find a detailed explanation of the different activities that were carried out during the course of the internship.

1.1 INTERNSHIP PROJECT

To initiate the internship process it was necessary to define the project that was developed alongside the organisation, in this case, VOID Software. After an initial meeting the project was defined, which consisted of implementing a standalone graphical environment for the Wazuh API and implementing the Wazuh security system within the installations of the VOID organisation. In the following subsection we will explore in greater detail what the requirements and constraints of this project were.

1.1.1 *Statement of the problem*

There are two main problems that were addressed during this internship, these problems are:

- Create a Graphical User Interface (GUI) for the Wazuh system (specifically the Wazuh server), which can aid a security team in managing security related events in a practical way, to further enhance prevention, detection and response capabilities against adversary security events, should these occur;

- Implement the Wazuh system within the headquarters of the organisation, so that critical systems can be monitored in real time and security incidents can be either prevented, detected or resolved in a timely fashion.

1.1.2 *Significance of the project*

Wazuh is a endpoint monitoring solution with several capabilities (Wazuh, 2021). Its architecture is composed of three main components: the Wazuh server, a GUI and full text search engine. Of these components two are based on products from the elastic stack family. Namely the GUI for this system is a plugin for Kibana, which is a GUI for searching, analyzing and visualizing information that is stored in Elasticsearch indexes. As its search engine, the Wazuh system uses Elasticsearch which is a database specialised in performing text searches. These search capabilities are possible since Elasticsearch is based on Lucene indexes which are specifically created to optimise the task of searching through high volumes of information.

As an organisation VOID software is creating a cybersecurity department. Within this department different tools will be used to perform various security-related operations such as security analysis, incidence response and security management.

During the definition of the cybersecurity department, the organisation also looks to develop, use and test their own tools within the team. As such an initial project was proposed by VOID, which consisted of setting up the Wazuh system within the headquarters of the organisation. Once this was done, the next step consisted of removing the components that are part of the Elastic Stack (ELK Stack) from the Wazuh system, more specifically to first create a GUI that replaces the current Kibana plugin and second to replace the current storage and search engine solution with the Apache Solr enterprise search platform.

Performing these modifications opens the possibility for VOID software to use the Wazuh system freely, without any of the restrictions associated with products that rely on the ELK Stack. Furthermore, the modifications to the already existing Wazuh system will provide the following benefits for the VOID organisation:

- Having a GUI that performs the same data search, analysis and visualization capabilities with the additional benefit of not depending on Kibana and Elasticsearch;

- The possibility of extending, adding or modifying the current capability set of the Kibana plugin, which, in turn will allow the organisation to adapt the GUI software to fit their specific needs;
- The assurance that the implemented GUI follows safe coding practices beyond the ones that are already implemented into the Java environment, of which the most vital is providing strong authentication and authorization mechanisms to protect the sensitive information that is collected by the Wazuh system;
- Since the products that belong to the ELK Stack can't be commercialised (without following specific guidelines), these modifications will give the possibility, if appropriate, to commercialise and implement the Wazuh flavoured system within other organisations, without any kind of restrictions.

1.1.3 *Objectives*

Before initiating the development of this project the organisation (VOID Software) and the developer (Tomás Avila) defined a set of requirements and objectives that were fulfilled, insofar as possible, by the end of project. These objectives and requirements were subject to change during the development stages, depending on the evolution of the project and the necessities of the client, in this case the organisation for whom the software is being developed.

The initial set of requirements and objectives for this project were the following:

- Implement a GUI for the Wazuh system that replaces the default one provided by Wazuh. The GUI should initially at least allow a user to perform the same actions as the Kibana plugin;
- Perform an initial migration of the storage solution behind the application, specifically replace Elasticsearch with Solr;
- Develop an application that is able to run on a wide range of common operating systems (e.g. Microsoft Windows, Linux and Mac OS);
- Provide several methods of authentication for the user (e.g. username and password, smart card);
- Allow the members of a security team to collaborate when performing different cybersecurity operations;
- Set up and run the Wazuh system within the VOID organisation;

- Create a configuration for the Wazuh system that is adapted to the organisation and collect general security information from this same organisation's hosts, servers and if necessary network devices;
- Develop the application following secure coding practices.

1.2 ORGANISATION WHERE THE INTERNSHIP WAS CARRIED OUT

VOID Software SA is a company the specialises in software development. Although software development is at the core of this company's functions, VOID also provides other services within the Information Technology (IT) industry such as those in the areas of cybersecurity, digital forensics, machine learning and data science.

VOID Software is a private company that was founded in Leiria, Portugal, in 2006. As mentioned, its main area of focus is software development, specifically with regards to web, mobile and desktop applications. The main installations of this company and the company's development team are based in Leiria, and the company is composed of around 30 people which include the roles of developers, testers and product owners among others.

This company has a modern working environment that seeks to keep their employees well looked after. Within their installations they have a main office where all the team works, which provides an easy flow of communication between team members. Furthermore these installations have areas where team members can take breaks as required such as the terrace, the games room and the kitchen. Because of the current pandemic situation, the company also allows its employees to work from home if they so desire, which in turn promotes remote work and the adoptions/reinforcement of smart tools and processes. To further increase the comfort of their employees, VOID also provides the necessary food and drink supplies during the day.

As a company VOID can provide the following services:

- Software Engineering;
- Blockchain;
- Machine Learning / Data Science;
- Augmented Reality / Virtual Reality;
- Mobile Application Development;

- Web Applications;
- Desktop Applications;
- Cybersecurity / Digital Forensics.

1.3 REPORT STRUCTURE

This report is structured into chapters, which describe the different steps that were carried out to complete the project done during the internship. In this section one can find an overview of the topics that are covered in each of the chapters.

In **Chapter 2** (*Graphical User Interface Concepts*) one can find the main concepts and technologies that were used while developing the application, which was a requirement of this internship. This chapter gives a summary of what JavaFX is and the architecture behind this development platform. Furthermore this chapter addresses the design pattern that was used for the application development. Finally important concepts for developing a secure and usable Graphical User Interface are explored.

In **Chapter 3** (*Security Concepts*) one can find the main security concepts that are related to this project. This chapter presents an overview of why cybersecurity is important nowadays. Furthermore the concept of endpoint security is addressed in this chapter, this topic is expanded on by understanding how technologies such as Endpoint Detection and Response, Security Information and Event Management and Intrusion Detection System work. Additionally, concepts related to the processes of regulatory compliance and incident response are overviewed. Finally the main concepts related to the Wazuh system are given.

In **Chapter 4** (*Development*) a detailed explanation of the methodology, architecture and stages of development for this project is given. The methodology used during this project is described by the process followed and the tools that were used. An overview of the architecture of the system that was used and the application that was developed is given. Finally the development of this project is described by its different stages: investigation and setup, where we look at the tasks executed at the start of the project; Wazuh setup, where we look at how the Wazuh system was implemented within the organisation; requests to the Wazuh API, where we look at how the communication with the Wazuh server was implemented; data storage migration, where we look at how the database of the Wazuh system was migrated; and finally, WazuhFx development where we look at the main tasks that were carried out to develop the new Graphical User Interface for the Wazuh system.

Finally in **Chapter 5** (*Conclusion*) a conclusion is given with regard to the tasks that were carried out during this project. This chapter proceeds to give a list of improvements and additions that could be carried out with the final results which were achieved by the end of the internship.

2 GRAPHICAL USER INTERFACE CONCEPTS

One of the main goals of this project was to develop a GUI for the Wazuh system. As such it was important to get an overview of the technologies that could be used to develop this type of front-end application, to then initiate the development process of the project. Thus an initial investigation was carried out with regards to which technologies could be used for this purpose.

Another important aspect of developing this GUI was the architecture that would be used for its implementation. This aspect is significant because it directly influences how the application was developed and how it can be maintained and improved in subsequent iterations of the project. Finally, since this project is related to the area of cybersecurity, it was important to look at what some of the best practices are for developing a secure GUI whilst respecting its usability.

In this chapter one can find information related to the concepts that were put into practice during the development of the GUI application. In **Section 2.1** (*JavaFX*) an overview of the JavaFX development platform is presented. In **Section 2.2** (*Model-View-ViewModel pattern*) the Model-View-ViewModel (MVVM) architecture and how it can be used alongside JavaFX is described. And finally in **Section 2.3** (*Graph user interface secure development*) important concerns that should be taken into consideration when developing a GUI are summarised bearing in mind the security and usability of the application.

2.1 JAVAFX

JavaFX is a library built upon the Java API, which provides a set of packages that enable developers to design, test and debug client GUI applications that can be run consistently across diverse platforms (Oracle, 2014).

Because of using the Java API, JavaFX can use any of the Java libraries for whatever purpose (e.g. access system resources). Furthermore, this feature provides the ability to develop the whole application using Java code. To add to the advantages of developing applications with JavaFX, this platform provides an easy way to separate the components of the GUI from the business logic that supports the application (Oracle, 2014). Finally JavaFX provides ways to customise the visual components of the application by using Cascading Style Sheets (CSS). The main characteristics of the JavaFX library that enable these features are:

- **Java APIs:** The JavaFX library is built with classes and interfaces written in Java;
- **FXML and Scene Builder:** JavaFX provides a declarative way of defining views through FXML, which is a XML based markup languages. FXML markup can either be created by coding it or by using the *Scene Builder* tool which is an interactive GUI which then generates the FXML markup;
- **WebView:** JavaFX can render web content by using this component;
- **Self-contained application deployment model:** Applications created with JavaFX have all the application resources and a copy of the Java and JavaFX runtime libraries in a single application package, this package can then be run as a native application on any operating system;
- **3D Graphic features:** JavaFX can present 3D graphics and models within an application with this feature;
- **Rich Text Support:** Rich text such as JSON and XML can be viewed within JavaFX applications with this feature.

2.1.1 *Architecture*

JavaFX uses different components to render and run applications. Together these components compose the JavaFX architecture which can be seen in **Figure 1**.

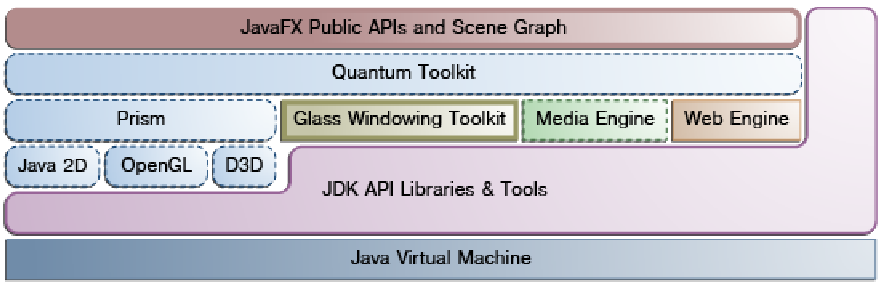


Figure 1: The JavaFX architecture (Oracle, 2014)

Although JavaFX has several components that make it work, in most cases, when developing an application using this library, the developer only interacts with the top level components of the architecture (JavaFX public APIs and Scene Graph). Because of this, in this section we will only focus on explaining the architecture of the scene graph and the components around it.

2.1.1.1 *Application architecture*

Every JavaFX application inherits from the *javafx.application.Application* class, which defines methods to initialise, start and tear down the application. Additionally JavaFX applications are divided into different components (Chin, 2019). These components can be seen in **Figure 2** and are described as:

- **Stage**: Which acts like a container for all the other JavaFX objects. When an application is started a primary stage is created;
- **Scene**: To be able to visualise the contents of the applications it is necessary to create a scene. The scene contains all the visible content of the application;
- **Scene Graph**: Is a hierarchical structure containing the visual object of the application, which are called nodes. The scene graph is considered the root node of the application;
- **Node**: Each visual element of the application is a Node, each node has one parent and may contain other nodes. The root node of the application will always be present as the first node in the scene graph. For a node to be rendered it must be added to the scene graph and can only be added once to each scene graph.

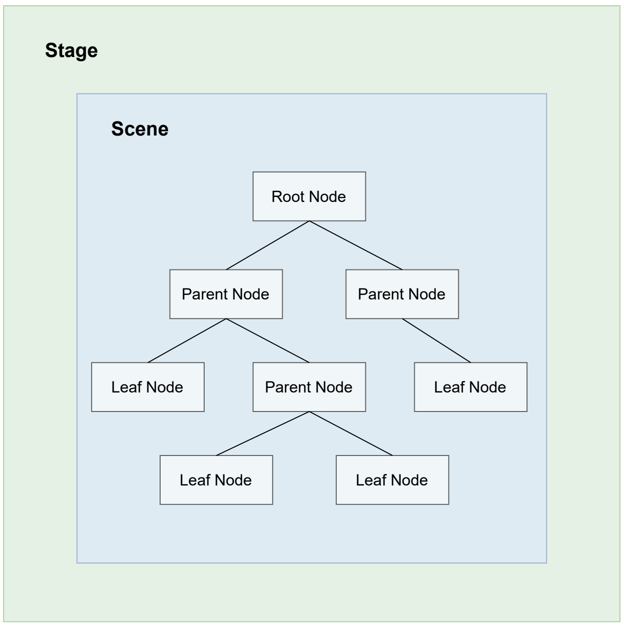


Figure 2: The JavaFX application architecture

2.2 MODEL-VIEW-VIEWMODEL PATTERN

The MVVM design pattern, as for many other patterns used for GUI development, seeks to separate the logic of the application from the visual components of the application.

This design pattern is an improvement of the *Presentation Model* created by Martin Fowler. Microsoft initially developed the MVVM pattern to aid in the development of applications using the Windows Presentation Foundation (WPF) (Microsoft, 2021).

The main objective of the MVVM pattern is to separate the UI view from the state and behaviour of an application. As noted in *The MVVM Pattern* (Microsoft, 2012), this pattern is especially useful alongside frameworks that provide two things: a declarative way to define views, for example, with markup language such as FXML and Extensible Application Markup Language (XAML) used in JavaFX and WPF respectively; and a way to bind variables to each other.

When defining views with this kind of approach (declarative syntax), it is common to implement the logic as code that is directly associated to the view. Because of how tightly coupled the logic and the view are in this model, the cost and complexity of making changes to a view and the logic behind it increases rapidly when the application grows (Microsoft, 2012).

As such using the MVVM pattern has the following advantages:

1. Separation of concerns - the UI, state and logic are all contained in separate locations;
2. It is a natural pattern for frameworks that provide a declarative syntax to define views;
3. It enables a designer-developer workflow - by allowing designers to be creative with the view design without generating extra complexity for the developer;
4. It increases the applications testability - by moving the logic to a class that can be instantiated without the need of the UI. Thus, it is easier to write a unit test for the functionalities implemented in such a class.

2.2.1 *JavaFX and MVVM*

As previously mentioned, there are two main requirements to implement the MVVM pattern, JavaFX has both of them. JavaFX provides a declarative syntax through FXML to define views which are then associated to controllers which implement the logic behind such view. Furthermore JavaFX provides variables called *Property Variables* which allow other variables to bind to them. When two variables are bound the values of such variables update depending on the value of the variable that they are bound to and the condition of the binding. These conditions can be either unidirectional, which means that a variable updates only when the variable it is bound to updates, or bidirectional, which means that either variable updates when either of the variable's values changes.

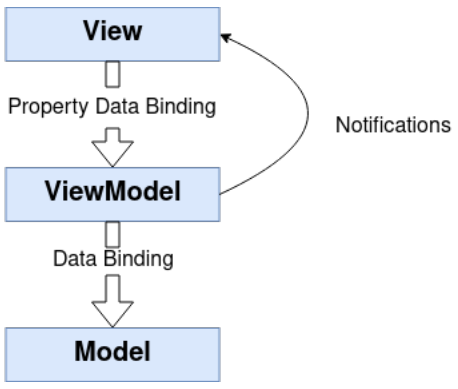


Figure 3: The communication flow of the different components used by the MVVM pattern.

When developing a JavaFX application while using the MVVM design pattern, the following components should be implemented for each view:

- **Model**: Can be implemented in different ways depending on the application that is being built. Generally contains the business logic and the data persistence for the application. The *Model* will not know about the *View* or the *ViewModel*;
- **View**: The *View* is defined by FXML and represents how the application will look visually. It is important that the *View* does not contain any logic. The *View* will hold references to all the visual components but will not know how they can be used or what actions they perform within the application. The *View* will contain a reference to the *ViewModel* and will not know about the *Model*;
- **ViewModel**: The main task of the *ViewModel* is to maintain the state of the *View*. It is important to note that this component is independent from the *View* and as such the *View* that is associated to it could be replaced. Finally the *ViewModel* provides a way for the *View* and the *Model* to communicate. For this purpose the *ViewModel* contains a reference to the *Model* and updates the *View* through data binding.

The communication of the different components while using the MVVM pattern can be seen in **Figure 3**.

2.3 GRAPHICAL USER INTERFACE SECURE DEVELOPMENT

When developing a User Interface (UI), two relevant concerns should be taken into account. The first has to do with security, specifically how to prevent unauthorised access to user's operations or information. To achieve security, a given application must implement security mechanisms and protocols (Möller et al., 2011). The second has to do with UIs usability, which is related to how a user performs actions within a system. On the one hand such actions influence the user's perception of the system (e.g. usable vs. unusable) and on the other hand it influences the user's perception of security (Möller et al., 2011). These two concerns are developed and explored by two engineering practices:

1. **Security engineering:** Practice that seeks to integrate security activities and policies within the software development process (Rudolph and Fuchs, 2012);

2. **Usability engineering:** Practice that seeks to guarantee efficiency, effectiveness and satisfaction for a user when performing a certain task and how to apply it to software development (Weir et al., 2009).

To apply these engineering practices into the software development process it is necessary, in the initial stages, to determine the security requirements of the software followed by taking security design decisions. Furthermore the security and system requirements should be related to usability with a focus on the users.

Cranor and Buchler (Cranor and Buchler, 2014) state on the one hand, that it is not possible to develop secure software based only on usability requirements. On the other hand, they state that it is also not possible to develop user-friendly software when the design is based on security. These problems arise from there not being enough investigation about security in GUI when compared to User Experience (UX) and UI design. This limits the ability to predict how the user will react to a security incident, which in turn limits the security design decisions. Since most investigations that focus on usability neglect the security aspects, it is not possible to fulfill the unique requirements of secure applications.

Because of this, it is important to find a balance between security and usability, which involves finding a good trade-off between the two. For security to be implemented and work well within a GUI application it should be implemented into the design of such application.

From a user's perspective a system seems to be more secure when there are more security mechanisms implemented by such system (Gunson et al., 2011) (ex. password authentication and two factor authentication). Although the perception of security is augmented, such behaviour also reduces the usability of the system. This proves that security (at least from a user's perspective) is greatly influenced by how the system is designed. As a consequence of the influence of a system's design, whenever possible, it is recommended to go beyond a human-centered design and implement user decision making (Cranor and Buchler, 2014) (aka. guiding the user), to prevent users from performing actions that may compromise the security of a system.

3 SECURITY CONCEPTS

One of the main goals of this project was to implement and configure the Wazuh system within the VOID headquarters. This system has three main security functionalities. The first is a Host-based IDS (HIDS), the second is regulation compliance and security management and finally the system can be used as a comprehensive Security Information and Event Management (SIEM) solution.

Wazuh was then used to monitor a set of endpoints within and outside of the organisation with the objective to prevent, detect and respond to security related events.

In this chapter, the main security functionalities implemented by the Wazuh platform are explored, providing a general idea of which security concepts are related to this platform, to later be able to explore more in depth the inner workings of the Wazuh system. As such this chapter provides an overview of the security practices, concepts, technologies and methodologies surrounding the Wazuh system.

In **Section 3.1** an introduction to why cybersecurity is important is presented, highlighting the need to have security solutions within an organisation. In **Section 3.2** the concept of Endpoint Security is explored, additionally solutions for guaranteeing the security of endpoints are listed and described succinctly. In **Section 3.3** an in depth description of Endpoint Detection and Response (EDR) systems is presented. In **Section 3.4** an in depth description of SIEM systems is presented. In **Section 3.5** and in depth description of how Intrusion Detection System systems work and can be used to secure endpoints is presented. In **Section 3.6** an overview of what regulation compliance is and how its verified is presented. In section **Section 3.7** an in depth description of the Incidence Response process and the steps of this process is presented. Finally in **Section 3.8** an overview of the Wazuh tool and its components is presented.

3.1 SIGNIFICANCE OF CYBERSECURITY

Nowadays organisations rely on vast amount of technologies and devices to support their operations and provide their services. Typically these devices and technologies all correspond to different vendor and in most cases the software that runs on them have different versions. This in turn increases the attack surface which the organisation is exposed to, ultimately this opens the door for vulnerabilities to be exploited by malicious entities.

Furthermore organisations handle large amounts of data that belongs, among others, to their clients, partners and investors. The increase in data handling by such organisations has also converted information into one of the most important assets. As consequence of this there is a growing interest from cybercriminals to steal, modify or destroy such data to cause harm to an organisation. Although data is one of the most important assets for an organisation, it is not the only asset that an organisation must protect. David Kim (2016) outlines that the assets that an organisation should protect are the following:

- Customer Data;
- IT assets and network Infrastructure;
- Finances and Financial Data;
- Service availability and productivity;
- Reputation.

Attacks and data breaches on an organisation can be carried out by internal or external perpetrators. According to the Verizon, 2021 report, 80% of data breaches are performed by entities that are external to the organisations, the investigation also shows that this figure has been increasing over the years, while the internal breaches have been decreasing. Additionally the Verizon, 2021 report show that data breaches, in most cases, are only discovered days, week or even moths after they occur. Failing to detect a data breach as soon as possible or in the worst case, not even detecting it, can cause a great amount of harm to an organisation there for it is in the organisations best interest to discover such breaches as soon as possible. With this in mind it is necessary to invest in improving cybersecurity detection capabilities as is outlined by Podzins and Romanovs (Podzins and Romanovs, 2019).

3.2 ENDPOINT SECURITY

The term *Endpoint Security* refers to the protection afforded to devices that are connected to a network, such as workstations, servers or mobile devices from being exploited by malicious users (Comodo, 2021). Since technology, software and security are areas in constant progress, this term has evolved with time to accompany such progression. With the evolution of endpoint security definition, the technology and processes that support it have also evolved (McFee, 2020). Solutions implemented for endpoint security have expanded and range from simple programs installed on hosts that verify file signatures to complex systems that can detect attacks based on the behaviour of several systems and network devices that are being monitored.

Gartner (Gartner, 2021) defines a Endpoint Protection Platform (EPP) as "a solution deployed on endpoint devices to prevent file-based malware attacks, detect malicious activity, and provide the investigation and remediation capabilities needed to respond to dynamic security incidents and alerts". As such EPPs are installed on host as a countermeasures to unwanted malicious activity, some examples of these platforms are:

- **Anti virus:** Which uses a database with malicious file signatures to detect threats;
- **Next generation anti virus:** That builds upon the threat detection capabilities of the Anti Virus (AV) by adding context base detection (monitoring processes, network traffic and other important system information) and in some cases even provides means to respond automatically to threats;
- **Endpoint protection:** Which expands the concept of Nex Generation Anti Virus (NGAV) to monitor all hosts connected to an organisations network, providing a more holistic view over the security status of the organisation;
- **Next generation endpoint protection:** That builds upon the concept on Endpoint Protection (EP) by adding behaviour based detection, machine learning and artificial intelligence strategies. These strategies help detect a wider range of threats including those classified as zero-day [1] attacks;
- **Endpoint detection and response:** Since it is not always possible to prevent security threats. Endpoint Detection and Response (EDR) use mechanisms that go beyond detecting threats and provide means to react to these events once they have occurred.

1 An attack on a system that explores a vulnerability that has not been registered before.

3.3 ENDPOINT DETECTION AND RESPONSE

EDRs are a type of security solutions that monitors different system-level events that occur on an endpoint, in real time, with the objective of being able to respond to security threats either, automatically or with the help of a security analyst. All the information that is gathered from the monitored hosts is stored, enriched and consolidated to gain an overview of the state of an endpoint during the whole life cycle of an attack.

The term EDR was first defined by Chuvakin (Chuvakin, 2013), with the objective to expand the definition and capabilities of endpoint security solutions. Chuvakin outlined that the key functionalities that these types of systems should possess are:

1. **Event detection:** Which consists of being able to classify an action that is executed on a host as normal or anomalous. To achieve this the activities of a host must continuously be collected and analysed;
2. **Event isolation:** Which consist of isolating a host that is compromised, so that the threat can't spread to other endpoints on the same network;
3. **Event investigation:** Which consist of providing means to perform forensic investigations after a security incident has occurred on an host. This is achieved by continuously collecting and storing data generated by a host;
4. **Response:** Which consists of providing the ability to react and recover from an incident on a host, allowing it to go back to a normal state of operations.

The mode of operation of an EDR, can generally be described by 5 different steps. The first is endpoint monitoring, which consists of continuously collecting information about a system and its events, this is generally done through the systems logs. With this information, in the second step EDRs creates a profile of each system it monitors, this profiling action is referred to as behavioural analysis. The third action is the ability to quarantine an endpoint that has been affected by an attack, from the rest of the hosts on the network. After quarantining an endpoint a EDR provides detailed information of how the infection happened, allowing analysts to identify the same behaviour on other hosts. Finally EDRs are capable of enriching and providing extra information about an incident so it can be analysed and understood more effectively.

Although EDRs have great capabilities to improve security on the endpoints of an organisation, there are some limitations that these systems have, namely:

- Agents must be deployed on all the monitored systems which becomes harder the more hosts there is to monitor;
- Agents have a lot of privileges on the host system where they are installed on, which could enable a malicious user to take advantage of such privileges;
- It is hard to use agents to monitor Internet of Things (IoT) devices;
- In most cases EDRs are not capable of correlating network events with events that occur on the hosts.

3.4 SECURITY INFORMATION AND EVENT MANAGEMENT SYSTEMS

Miller (Miller, 2011) defines a SIEM as "A system which is a complex collection of technologies designed to provide vision and clarity on the corporate IT system as a whole, benefitting security analysts and IT administrators as well". The main goal of these systems is to monitor, identify and sometimes respond to security incidents. A SIEM achieves these goals by analysing individual security events and correlating them, providing an individual a global overview of such security events.

As outlined in Miller (2011), the core functionalities of a SIEM are:

- **Log management:** As the name denotes, log management is the process of managing the logs[2] of one or several systems. The process starts by configuring each node which logs will be collected from, to then forward relevant logs to a centralised location. The SIEM system is then in charge of storing, organising and archiving all the information that has been received. Finally the SIEM makes all the information that it has collected available through services;
- **IT regulation compliance:** By creating a set of requirements that should be fulfilled by all monitored system, these requirements are checked at defined intervals of time by using rules. This is done with the objective of identifying compliance (or not) of a given standard that is enforced by the organisation;
- **Event correlation:** By correlating the events that were collected, the SIEM system can provide a higher level of intelligence, by making the alerts that it generates more accurate with regards to a security incident;
- **Active response:** A SIEM can be configured to respond automatically and actively to specific types of events. This is done by interacting with one or several systems that could be compromised by a certain threat;

2 System information and application events

- **Endpoint security:** By looking at the logs a of an endpoint, the SIEM can validate if the security mechanisms of such endpoint are up to date and running.

Although a SIEM can generally be characterised by these functionalities, there are many different implementations of these systems, which depend on the vendor, the organisation that implements the system and other factors. Because of this the scope of functionalities and monitoring capabilities of a SIEM can vary.

3.5 INTRUSION DETECTION SYSTEMS

An Intrusion Detection System (IDS) is usually installed on a specific sector of a network or directly on a host. The objective of this system is to continuously monitor network activity, in order to detect events that are not considered normal behaviour such as unauthorised network packets, vulnerability exploits, a large quantity of packets or other incidents that can compromise the security of a network. If such incidents are detected the IDS will create an alert, and in some cases depending on its configuration react to the the incoming threat. An IDS can be implemented as a software or hardware solution, and is designed to be deployed on different environments.

Liao et al. (2013) outlines different kinds of IDS based on different factors. These factors are: the methodology they use; the classification approach; and the types of IDS. In the following subsections we will give a small overview of each of these IDS classifications.

3.5.1 *Detection methodologies*

Detection methodology refers to how the IDS detects incidents on a system or network when they occur. Liao et al. (2013) states that there are three major kinds of detection methodologies.

- **Signature-base detection**: Signature-based Detection (SD) refers to the technique that uses signatures, which are patterns or strings that are associated to a known threat. Detection is achieved by comparing the signature with all captured events;

- **Anomaly-based detection**: Anomaly-based Detection (AD) works by creating profiles of normal or expected behaviours, profiles are created by monitoring different system, network and user activities. Such profiles can be static which don't change over time or dynamic which are updated over time based on the system that is being monitored. AD detects incidents by comparing profiles with observed events, if there is a deviation from the normal profile an alert is created;
- **Stateful protocol analysis**: Stateful Protocol Analysis (SPA)s detect incidents by tracing protocol states, this method of detection requires vendors to develop generic profiles for different protocols;
- **Hybrid**: Uses a combination of the other methods to detect threats.

3.5.2 *Detection approaches*

Detection approach refers to the the conditions which are used by the IDS to decide if an event can be classified as an incident or not. In their review, Liao et al. (2013) states that there are five approaches to classifying incidents.

- **Statistics-based:** Uses a predefined threshold, such as mean and standard deviation, and probabilities to identify intrusions;
- **Pattern-based:** Focuses on known attacks through string matching;
- **Rule-based:** If–Then or If–Then–Else rules are applied to construct the model and profile of known intrusions;
- **State-based:** Uses finite state machines based on the network behaviour to identify attacks;
- **Heuristic-based:** Is based on biological concepts such as genetic algorithms and artificial intelligence to detect attacks.

3.5.3 *Technology types*

There are several types of IDS technologies. Technologies types are classified based on where the IDS is deployed to detect attacks, based on this parameter there are four classes.

- **Host-based IDS**: A HIDS monitors the system it is installed on and is designed to protect against internal and external threats. HIDS monitoring capabilities are limited to the host on which it is installed on decreasing the scope on which it can take a decision as to what traffic is classified as malicious;
- **Network-based IDS**: Network-based IDS (NIDS) are placed at specific network points to capture network traffic and then analyses the activities of applications and protocols to recognise suspicious events;
- **Wireless-based IDS**: A Wireless-based IDS (WIDS) is similar to NIDS, but it captures wireless network traffic;
- **Network behaviour analysis**: A Network Behavior Analysis (NBA) recognises attacks by identifying unexpected traffic flows;
- **Mixed IDS**: A Mixed IDS (MIDS) combines any of the methods mentioned before to increase the probability of detecting an attack.

3.6 REGULATION COMPLIANCE

Regulations help companies improve their information security by providing guidelines and best practices. These regulations serve as a framework, that is based on the sector of industry the company is in and the type of data that the company deals with, meaning that different regulations may or may not affect specific organisations. Generally compliance regulations address security and privacy at the same time, putting responsibility on organisations to protect themselves from security incidents.

Regulation compliance is verified through assessments and audits. Assessments are done by companies to verify if they are compliant, with no consequences if the regulations are not met. On the other had audits are generally conducted by the entities that enforce regulations, and if an organisation is not compliant it can result in consequences such as financial fines or loss of reputation.

Complying with regulation helps organisations gain advantages within their sector by, improving security with regards to their systems and data, minimise losses in case of undergoing an attack, increase control over what their employees can or cannot do with their credentials and maintain trust among their clients and providers.

3.6.1 *Security regulation compliance laws*

There are several regulation compliance laws, which affect different sectors of the industry as mentioned before. Some of the most important ones are:

- **Payment card industry data security standard**: This standard targets businesses that process, store or transmit credit or debit card-holder data. Any application, network, server or any other device connected to card-holder data environment are also included in this standard. This regulation separates controls into 12 different areas, each of those areas has its own specific requirements;
- **Healthcare insurance portability and accountability act**: This regulation requires administrative, technical and physical safeguards to guarantee the confidentiality, integrity and security of Protected Health Information (PHI) which refers to sensitive medical patient information such as medical records and financial information;
- **General data protection regulation**: This is the core digital privacy legislation in Europe. It defines a set of rules that gives European Union (EU) citizens control over their personal information. The General Data Protection Regulation (GDPR) defines laws an obligations that are based around privacy, personal data and consent.

3.7 INCIDENCE RESPONSE

Incidence Response (IR) is the capacity an organisation has to react to a incident. IR is normally based on a framework which helps to streamline and automate the process as much as possible, such framework is referred to as a IR plan. Hence the main objective of such plan is to reduce as much as possible the time between when a incident happens, the time it takes detect and deal with it. A secondary, yet important, objective of the IR plan is to gain knowledge about each incident that occurs to later enhance the security capabilities of an organisation.

To further understand what a IR plan is and what the main objectives of IR are, we will go over the SysAdmin, Audit, Network, and Security (SANS) institute high level definition of a incident response plan as defined the in *Incident Handler's Handbook* (Karl, 2012). We will look at this definition since it separates the IR process into 6 phases, as opposed to the 4 phases defined by the National Institute of Standards and Technology (NIST), which in turn helps to explore more in detail the IR process.

As can be seen in **Figure 4**, the IR process is composed of 6 stages.

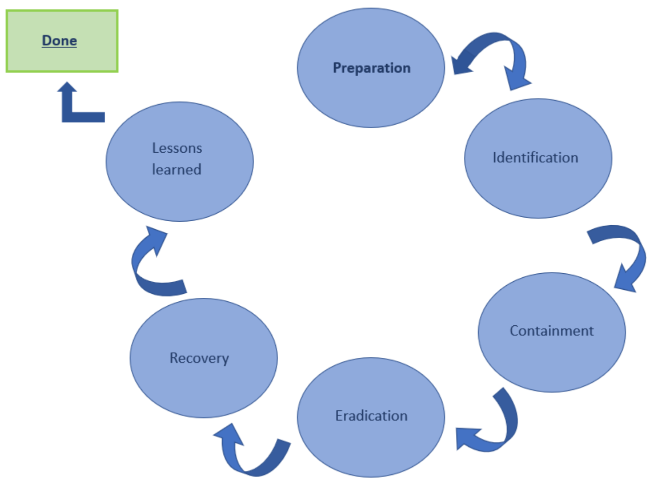


Figure 4: Stages of the IR plan as per the SANS definition.

- **Preparation**: During the preparation stage all the ground rules and processes for responding to an incident are defined. Since this stage lays out the grounds of how to act for other stages, it can be considered one of the most crucial ones in the IR process (Karl, 2012);
- **Identification**: When an incident is reported a more in depth analysis is required to have a better understanding of the event. One of two outcomes can be observed from such analysis. In the first outcome, the incident is classified as a false positive in which case the security team rejects the alert and doesn't take further action. In the second, the incident is acknowledge by the security team and further actions are taken by the responsible members;
- **Containment**: In this stage of the IR plan the security team will aim to isolate all the affected devices, either by disconnecting such devices from the network, where they could infect other device, or by alerting and informing whomever it might interest about the incident so they can identify it if necessary. The main objective of this stage is to prevent other users or devices from being affected by the incident that has been detected, thus restricting the consequences of such event;
- **Eradication**: In this stage of the IR plan the security team seeks to eliminate the threat that caused the incident from the affected devices. Depending on

the extent of the incident the team will have to work on different range of cases;

- **Recovery**: In this stage of the IR plan the security team performs the required operations to restore the organisations operations to normal. The steps followed during this stage can vary depending on how serious the incident was, in some cases alerting the required entities is enough to return to normal operations. On the other hand if the attack caused a large amount of damage, critical systems will have to restored from backups, which would require a large amount of time and effort;
- **Lessons Learned**: After every incident there are key takeaways that can help improve the security of an organisation and improve the processes that are activated in case of an adversary event. In this stage of IR plan, the final stage, a complete overview of whole IR process can be observed

3.8 WAZUH OVERVIEW

Wazuh is an open source security platform that provides a range of security functionalities. Many of these characteristics were inherited from the Open Source HIDS Security (OSSEC) project, additionally Wazuh extends these functionalities to create a well rounded security tool. Wazuh as a tool can be categorised under the term EDR as such it provides the following functionalities:

- **Host based intrusion detection**: By installing a agent on each host which should be monitored, Wazuh observes such hosts either by looking at its behaviour or using file signatures. Additionally Wazuh agents can monitory different activities on a host and check its configurations;
- **Regulatory compliance**: By mapping security alerts to regulation requirements Wazuh performs the necessary controls to assure that a system complies with the specifications defined by different IT security standards;
- **SIEM functionalities**: All the alerts generated by the Wazuh server are stored, so that at any point in time, they can be accessed and analysed for greater visibility of the overall security of the monitored systems.

Wazuh as a tool is mainly used to perform incidence response activities, specifically it can aid in the process of threat prevention, detection and recovery. Although these are the main areas on which this tool focuses, it also provides ways to gain insight on an attack after it occurred and check if systems are compliant with regards to IT regulations and organisation policies.

One of Wazuh's main features is that it can monitor different environments through its agents, ranging from workstations to containerised components. In cases when agents cannot be installed on a device, as is the case of network devices, syslog[3] can be used to forward relevant logs to the manager.

To achieve its functionalities the Wazuh system relies on three main components: the Wazuh agent, the Wazuh manager or server (these terms can be used interchangeably) and the ELK Stack; as can bee seen in **figure 5**. Each of these components have specific objectives and functionalities that contribute to the Wazuh system. Moreover these components communicate, in a secure manner, with each other by either sending information or commands to build this system as a whole.

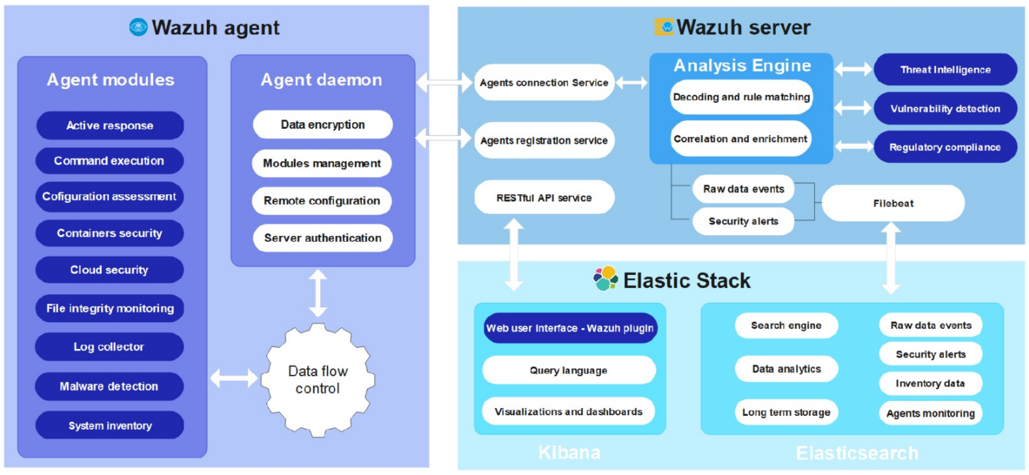


Figure 5: The Wazuh system architecture.

3.8.1 *Wazuh agents*

These agents are installed and executed on every endpoint which is desired to be monitored by Wazuh and run as a lightweight background process. Agents are divided into separate modules that perform specific tasks, such as collecting and forwarding the logs of a system. Individual modules can be activated/deactivated through a configuration file that can either belong to a single agent or a group of agents.

3 Protocol designed to forward system logs or event messages from a device to a specific server

Agents can be grouped depending on what kind of system they monitor, Wazuh has some predefined groups which are assigned to agent when they register to the manager for the first time. If necessary agents can be assigned groups depending on the organisation's requirements.

For an agent to be able to communicate with the Wazuh manager they must register to it first, during this process the agents receive a cryptographic key which is used to secure all communication between one another. After the registration process is complete, the manager and agents are ready to exchange information.

Agents can perform the following actions depending on the configuration of their modules:

- **Collect logs**: System and application logs are collected by the agents, then these logs are forwarded to the Wazuh manager where they will be analysed in depth;
- **Execute commands**: Agents can execute authorised commands sent by the manager on the hosts that they are installed on. These commands can perform actions such as collecting system information, blocking network traffic or deleting a file on the system;
- **Check file integrity**: Agents maintain a database with the signatures of predefined files or paths on the system, if a file is created, modified or deleted, the agent generates send this information to the manager;
- **Asses security configurations**: Depending on the profile of the system, agents check the configuration files of such system and compare it to the Center of Internet Security (CIS) standard configuration for that specific system;
- **Perform an active response**: Predefined actions to certain events can be performed by the agents, in this way threats can be mitigated before they escalate and become a bigger problem;
- **Detect malware**: Agents use different techniques to identify malware on the system it is installed on;
- **Cloud and container security monitoring**: Agents can be installed on cloud and container environments, detecting changes when they occur.

3.8.2 *Wazuh manager*

The manager is the central component of the Wazuh solution, it is responsible for gathering the data sent by the agents, processing it and creating meaningful alerts and statistics it, this information can then be analysed at any point of time. In addition to handling all the information of the system, the manager is also responsible for keeping agents up to date, updating configurations and executing actions on monitored hosts.

The Wazuh manager can be installed on a Linux machine either as single node server or a multi-node cluster. The type of installation of the manager will depend on the amount of devices that will be monitored and the quantity of information that will be generated by each of those hosts.

The core functions that are carried out by the manager are performed within the analysis engine. This engine uses a set of decoders to interpret and extract information from the data it is sent by the agents, subsequently the extracted data is compared against several rules that are associated to the decoder. If a rule is triggered an alert is generated. The default installation of the Wazuh server already has a large amount of predefined decoders for a wide variety of log formats and rules to detect different security incidents and events. Additionally rules and decoders can be created for specific situation and log formats respectively to meet an organisations requirements.

To further enhance the quality of the information generated by the Wazuh server, events are correlated with other events and enriched with additional information, thus providing a greater overview of each event. Finally each log and security alert is forwarded to the Elasticsearch cluster to be stored.

In addition to performing such analysis on the data that is received, the Wazuh manager checks for vulnerabilities, verifies the state of the endpoint, reviews policies and regulations to check if a host complies with them, verifies if any adversary tactics are being used against the host based on the mitre ATT&CK framework, uses a threat intelligence databases to check if a system is compromised and can integrate with other systems to trigger events or share information.

Finally the Wazuh manager provides a REST API which can be used to interact with it.

3.8.3 *Elastic stack*

The final components of the Wazuh system are part of the ELK Stack, which are tools specifically designed for collecting, storing and visualising information. Precisely this system uses the following tools which are part of the ELK Stack:

- **Elasticsearch**: This text search engine is used to store the logs, alerts and statistics that the Wazuh system generates;
- **Kibana**: Is a web interface built on top of Elasticsearch, which can be used among other things to visualise, analyse and mine the information stored in the Elasticsearch indexes. Wazuh has a dedicated plugin for Kibana which has predefined graphs for visualising alert information and different panels to interact with the Wazuh API;
- **Filebeat**: Is used to forward the logs and alerts that the Wazuh manager generates to Elasticsearch, furthermore it guarantees that the information has arrived to its intended destination.

4 DEVELOPMENT

The development phase of this project complements the investigation that was laid out in **Chapters 2 and 3** by putting into practice the theory that was acquired to develop a project related to the fields of cybersecurity and application development.

The development process was divided into stages to help keep track of the tasks that had already been done and to separate tasks which despite being related to each other within the project, had their own goals with respect to one another.

This chapter discusses all the tasks that were carried out over the duration of this project to fulfill the objectives that were set when the project began. Additionally, this chapter serves as a guide for anyone who would like to acquire a greater understanding of the project, whether to continue with the development of this project, use it for their own purposes or any other use of this work. Finally, this chapter addresses how the project was developed.

In **Section 4.1** an overview of the methodology for developing this project is given. In **Section 4.2** the architectures related to the GUI application and the Wazuh system (before and after modifications) are explored. In **Section 4.3** the initial steps of this project are overviewed and a guide to set up the development environment for this project is given. In **Section 4.4** the process to set up the Wazuh system for development purposes and within the VOID organisation is described. In **Section 4.5** the process for implementing the required entities in Java to communicate with the Wazuh API is explained. In **Section 4.6** the process to migrate the Elastic search database to the Solr database is explained. And finally in **Section 4.7** the process to develop the GUI application is overviewed.

4.1 METHODOLOGY

In this section we will look over how the project was implemented. This section covers the tools and technologies that were used and the process that was applied in the implementation of the project.

4.1.1 *Tools*

Every software development process requires a combination of tools to support it. For this project different tools for tracking progress, developing software and team communication were defined, with the objective of making the process as simple and effective as possible. The tools that were used for developing this project were:

- **Intellij IDE**: This tool was used to perform all the development and automated testing of the developed application. It was also used, wherever necessary, to perform debugging of the project's code;
- **Scene Builder**: This tool was used to design the views within the application;
- **Maven**: This project management tool was used to manage the different software and library dependencies that were used to develop the proposed solution;
- **Trello**: This tool was used to track the progress and requirements of this project. This tool was also used as a knowledge base where the most important information about the project's development can be found;
- **Rocket chat**: This chat client was used to communicate with the client of the project and other developers when help was needed;
- **Git repository**: This tool was used as a central repository for the code of the project and for version control of the features that were implemented progressively. The repository was also used as a reference for accepted features;
- **Cloud storage**: This tool was used to store important information that had to do with the project, such as files, collected logs, images or any other form of media needed to develop this project.

4.1.2 *Procedure*

For the development of the project a form of the Feature Driven Development (FDD) model was used. This methodology is, in essence, an agile methodology since it focuses on delivering features continuously over the course of the project. Features are defined and accepted by the client: as such, when working with this methodology, it is possible that during the development process the requirements of the project might change. Although this model formed the foundation of how development was carried out, some modifications were made. These modifications were necessary due to the structure of the development team and the roles the client played.

The development team was constituted by a single developer who owned each class. These classes were defined in the planning stages of the project. Owning a class implies responsibility for designing, coding, testing and documenting features. Additionally, the developer also took on the role of domain expert whose main responsibility was to make sure that the features developed worked as the client expected them to.

There was a project manager supporting the development team who oversaw the whole project. The project manager also aided the development team in taking decisions with regards to the architectural design of the system. Finally, the project manager played the role of the client, meaning that they received each iteration of the system and either accepted it or asked for modifications.

The steps followed while developing software using the FDD methodology are:

1. **Develop and overall model**: Define the domain model and the business problem that will be solved with the project;
2. **Build a features list**: Define all the features required to complete the project. Make sure that each feature takes no more than a maximum amount of time (generally the sprint duration);
3. **Plan by feature**: Alongside the client, the priority of each feature is defined;
4. **Design by feature**: Select a set of features that should be developed during the sprint;
5. **Build by feature**: In this step, all the parts to complete a feature are developed. When finished, the feature is reviewed. If the feature is approved and passes all the tests, the feature is added to the main build for the client to review.

4.2 ARCHITECTURE

During the development of this project there were two goals that required an architecture to be defined. The first was for GUI application that was developed; the architectural decisions for this component of the project were entirely up to the developer and the project owner. The second is was relative to the Wazuh system and the modifications done to it. As opposed to the requirement to develop the

GUI, the architecture of this system was already defined by the team developing it. Although this is the case, after performing the modifications required by this project, the final architecture is slightly different.

In the following sub-sections, we will explore in greater detail the architecture of the GUI and the architecture of the Wazuh system, before and after performing the modifications.

4.2.1 *WazuhFX architecture*

The architecture of the GUI is presented in **Figure 6**.

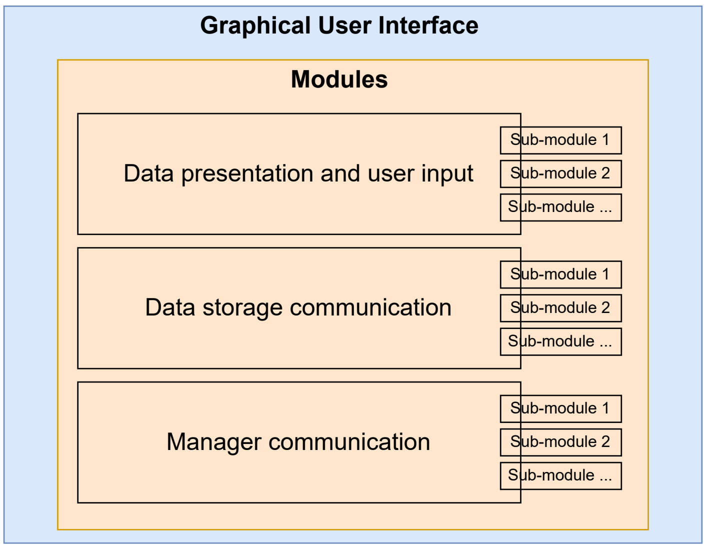


Figure 6: High level architecture of the GUI application.

The GUI is separated into three main modules:

- **Manager communication module**: Which is in charge of sending and receiving information from the Wazuh manager;
- **Data storage communication module**: Responsible for performing queries to the Solr instance. As such this module is in charge of fetching information about the alerts generated by the Wazuh manager;
- **Data presentation and user input**: Which is responsible of presenting and receiving information from the user by using the MVVM pattern. This module uses the other two to implement its functionalities.

4.2.2 *Final Wazuh system architecture*

As mentioned previously, the Wazuh system's architecture has already been defined by the Wazuh team (Wazuh, 2022a), this initial architecture can be seen in **Figure 7**. Although this is the case, one of the goals of this project was to modify this architecture by removing the ELK Stack components and replacing them with the new GUI and the Solr search engine. After performing these modifications, the final architecture can be seen in **Figure 8**.

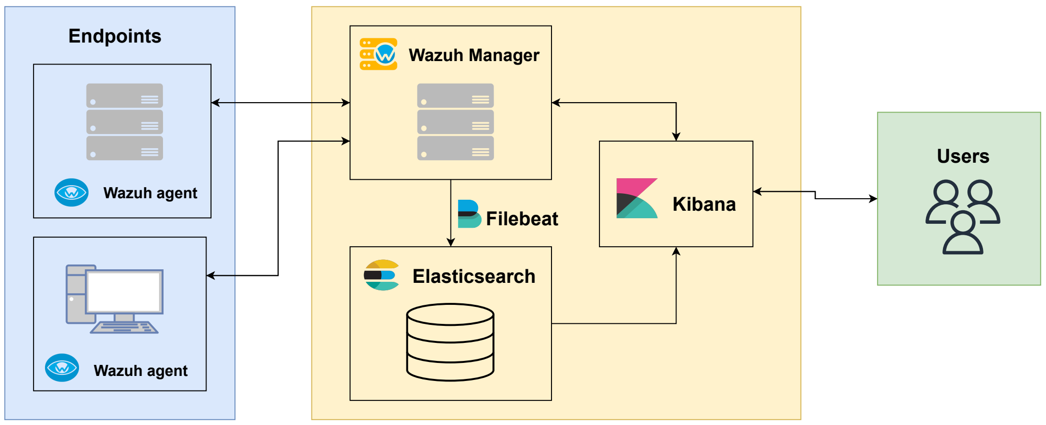


Figure 7: The original Wazuh architecture.

The architecture of this system is based on four main components:

- **The agents**: Which are in charge of monitoring and collecting information from a host. This information is then forwarded to the manager;
- **The Wazuh manager**: Which is in charge of processing all the information collected by the hosts and enriching it. Additionally, the manager is in charge of managing the agents that are connected to it and itself. The manager communicates with the GUI to either present information or receive commands;
- **The Solr instance**: Which is in charge of storing the information generated by the Wazuh manager. The Solr instance receives the alerts generated by Wazuh through the post tool which is included with the Solr installation and then makes these alerts accessible through queries to its REST API;
- **The JavaFX GUI**: Which is in charge of presenting the information that it fetches from the Wazuh manager and Solr. The GUI is also in charge of sending information to the Wazuh manager allowing the user to perform administrative tasks.

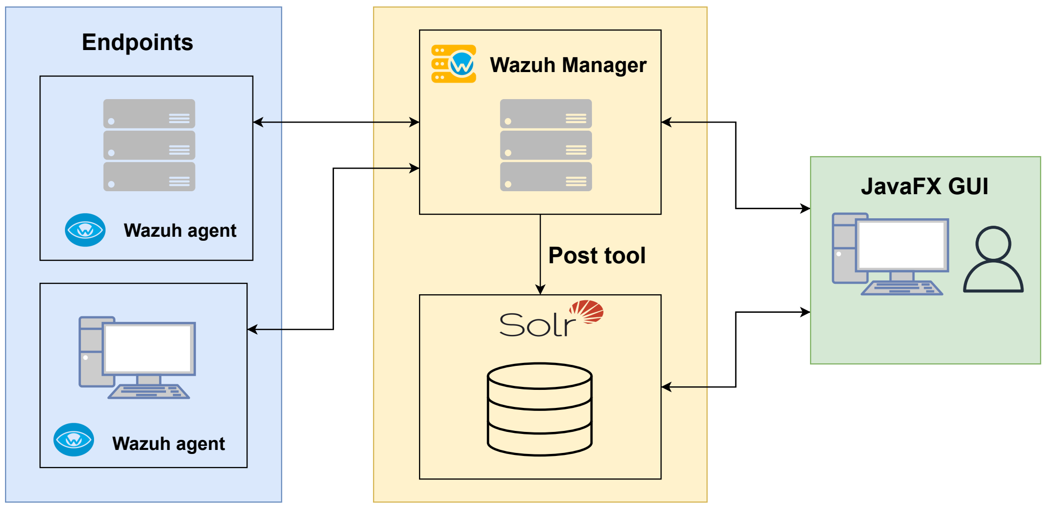


Figure 8: The new Wazuh architecture after performing the modifications of this project.

4.3 INVESTIGATION & SETUP

In this stage of the project an initial investigation was carried out, the purpose of which was to understand what technologies would be used to develop the required solution. Additionally, this phase of the project helped to define the requirements for the project's development.

Initially the GUI replacement for the Kibana dashboard, was going to be developed with ReactJS. We decided to use React since it is the main front-end development library used within the VOID organisation, hence if help were needed during any stage of development, other people at VOID would be able to provide guidance. Moreover, React applications can be run on any machine that supports a browser (and most machines have this feature), which in turn fulfils one of the project's requirements: "The application should be able to run on any operating system".

For that reason, during the initial part of the investigation phase a course to learn React development was followed [1]. This course follows a project-based learning methodology, by providing a set of small projects that introduce important concepts about React and web development in general.

Some of the most important topics covered by this course are:

- JSX;
- Components (Class and Fucntion);
- Document Object Model (DOM);

1 https://www.udemy.com/course/react-redux/

- Props;
- Life-cycle Methods;
- The hooks system (Including named hooks);
- Redux;

After a month or so of learning about React and how to write web applications using this library, a meeting was held with the client (VOID Software). During this meeting the requirements of the project were further elaborated on: this new set of requirements could still be implemented by using the React library, although some of them would be incomplete or even not dependent on the developer, namely the requirements that were related to security. For this reason, JavaFX was suggested and taken into consideration as an option for developing the project.

At this point of the project a comparison of both technologies was done to have a better understanding of which one of these technologies would better serve the purpose of developing the GUI for the Wazuh system. After performing the comparison, a decision was made to continue the development of the project using the JavaFX library.

This decision was made because JavaFX is not dependant on the browser and the security features associated to it. Additionally JavaFX provides the ability to interact with the hardware of the PC, making the application perform better if optimised correctly and allows to interact with dedicated hardware such as smart card readers, which will be eventually be required to login to the application.

The JavaFX library provides different features and APIs to develop GUI applications by using Java. At this point in the project, it was necessary to move the learning focus from react to JavaFX. To initiate the development, a set of tutorial and articles were followed to learn the main concepts of the JavaFX development platform.

Among other things, the main concepts that were learnt were:

- Java APIs;
- FXML and Scene Builder;
- JavaFX UI controls and CSS;
- The JavaFX Scene Graph;
- JavaFX layouts;
- 2D and 3D graphs.

To further understand how JavaFX works a set of small applications was developed, exploring the main concepts of the JavaFX library. To complete this stage of the learning phase, 4 weeks were required.

4.3.1 *Development environment setup*

As mentioned, the Integrated Development Environment (IDE) that was used during the development of this project was IntelliJ. This IDE was chosen since the developer was already familiar with it, making it simpler to navigate and use. To manage the dependencies and the builds of the project, the Maven framework was used, this helped streamline how dependencies are fetched and also provided a simple way to understand what the structure of the project was.

4.3.2 *Add maven to the project*

IntelliJ also provides a simple way to add Maven to a JavaFX project. The steps to add Maven are:

1. Right click the root of the project and click on “Add Framework Support”;
2. Select Maven as the framework;
3. Give a name to the project using the reverse-DNS syntax;
4. Click to load the changes within the project.

After adding maven to the project, the initial dependencies for the JavaFX project had to be added. A list of these dependencies, as presented in the dependencies file, can be seen in **Listing 1**.

Listing 1: Initial Maven Configuration

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         ↪  http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <groupId>com.voidsoftware.wazuhFX</groupId>
    <artifactId>internship</artifactId>
    <version>1.0-SNAPSHOT</version>

    ...

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>10</source>
                    <target>10</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-controls</artifactId>
            <version>${javafx.version}</version>
        </dependency>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-fxml</artifactId>
            <version>${javafx.version}</version>
        </dependency>
    </dependencies>
</project>
```

4.4 WAZUH SETUP

When it comes to installing the Wazuh server, the ELK Stack and deploying agents on the machines which should be monitored, there are several options. The options listed by the Wazuh documentation (Wazuh, 2022c) take into consideration that Wazuh will be used to monitor different environments. Additionally, it also considers that the resources available and the number of machines to be monitored will vary depending on the organisation where it is implemented.

As such it is important to review these criteria before setting up Wazuh, to make sure the solution fits the organisation's needs. After performing this revision, it is possible to continue with the installation process of the different components that are part of Wazuh.

4.4.1 *Network architecture of the organisation*

As mentioned in **Section 1.2**, VOID has its headquarters in Leiria, Portugal. This is where the workstations used by the developers are located, as well as the servers that host VOIDs internal services (file sharing, chat server, among others). Moreover these servers also host the test environments for different projects that are being developed by VOID. In addition to the aforementioned resources, VOID has servers for pre-production and production environments hosted by another entity in Germany, which are accessible from anywhere.

The network within the organisation is sectioned into separate VLANs for different types of devices and for the most part, can only be accessed from within the organisation's installations because of the Network Address Translation (NAT) rules defined on the firewall. A general structure of the network can be seen in **Figure 9**.

Based on this network design and the resources that are present within it, the decisions in the subsequent sections were taken.

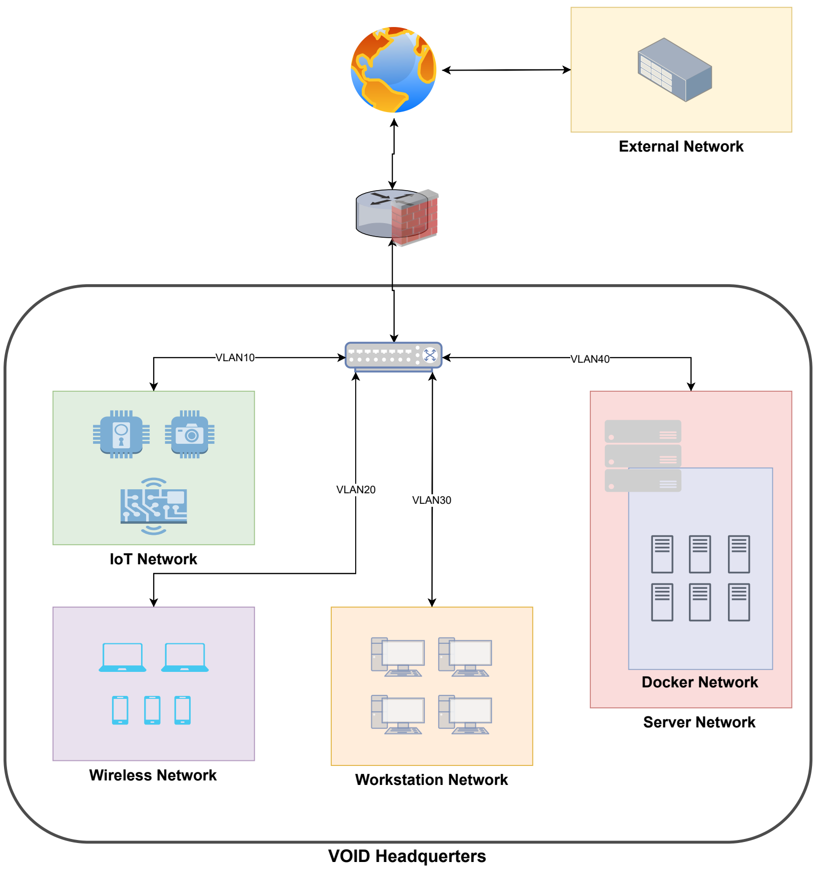
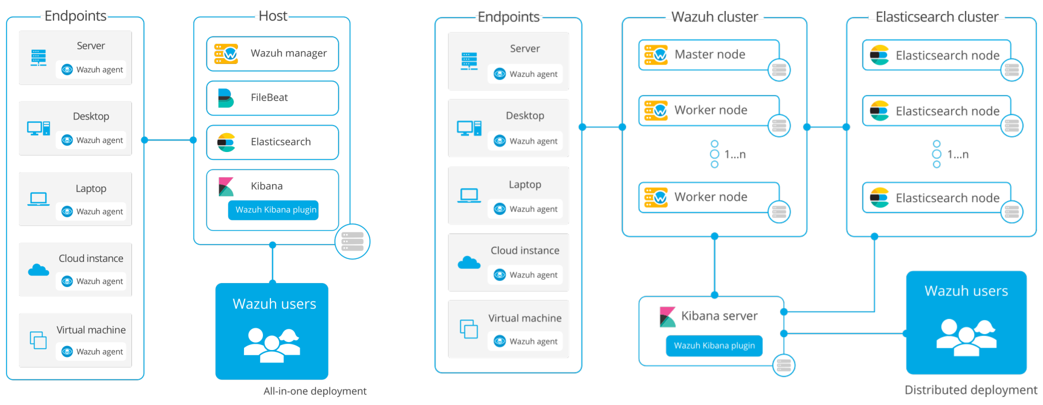


Figure 9: The general network architecture within the VOID organisation.

4.4.2 *Available methods for installation*

As per the Wazuh documentation there are three main ways to install the Wazuh components (not including Wazuh agents). These methods are: a step-by-step installation (Wazuh, 2022d); a Virtual Machine (VM) installation (Wazuh, 2022e); or a containerised installation for docker (Wazuh, 2022b). It is also possible to use a pre-built Amazon Machine Image (AMI) to use in Amazon Web Services (AWS) cloud. However, this option was not taken into consideration since the organisation was looking to host the Wazuh server within their installations. Furthermore the Wazuh components can be deployed either in an all-in-one or distributed mode, as can be seen in the following figures: **Figure 10a** and **Figure 10b**.



(a) Wazuh all-in-one deployment (b) Wazuh distributed deployment

Figure 10: Options for deploying Wazuh (Wazuh, 2022c).

When doing an all-in-one deployment, both the Wazuh manager and the ELK Stack components are installed on the same machine, meaning that all the resources of the machine on which they are installed on are shared. Conversely when performing a distributed deployment each of the components (Wazuh manager and ELK Stack) are installed on separate machines, thus each component has access to all the resources of the machine it is installed on. If a distributed deployment is done, there is also the option to create a multi-node cluster, which adds redundancy allowing the system to be up and running *all* the time and, if necessary, also grants the system the capability to scale.

4.4.3 *Development environment installation*

Two installations were carried out for development and testing purposes. Each installation was done on a separate machine and with specific objectives. In the following sub-sections, we will look at the installation process carried out and the goals that were accomplished, among other things.

4.4.3.1 *VOID PC installation*

The first development installation was carried out on a workstation at the VOID headquarters. The specifications of the hardware and operating system of this computer were the following:

- 32 GB of RAM;
- 512 GB of SSD (Additional storage was available if required);

- i5-2500k CPU;
- Linux Mint (x86_64) OS installed.

Since this was the first installation of Wazuh and it was also being used for development purposes, a step-by-step installation was carried out. The main advantage of performing this kind of installation is that it provided a general overview of the components of the Wazuh system, expanding the knowledge gained from reading the documentation. As such the steps found in **Appendix B** were followed.

After the Wazuh manager and the ELK Stack were installed successfully, it was possible to proceed with the next step of the Wazuh system set up; installing the agents on the hosts that were to be monitored.

The workstation where Wazuh was installed is located within the *Workstation Network* as can be seen in **Figure 9**. As such only other machines connected to this network can communicate with this Wazuh server. Consequently, the hosts that were monitored by this installation (where the agents were installed), were the workstations of other employees within the VOID organisation.

To be able to carry out the monitoring process, two steps must be followed:

1. Assign a static IP to the Wazuh server workstation;
2. Install the agents on the other workstations located in the headquarters' office **Appendix C**.

After some days of undertaking this process, it was not possible to work from the office anymore due to the pandemic restrictions, thus, this setup was abandoned for the rest of the project.

Although this was the case, performing this initial setup provided us with some valuable information about the security of a default installation of the Wazuh system, which then aided with the production installation of this system. The following security issues were identified:

- Default passwords are used to access the Kibana dashboards and the REST API;
- The communication between the Wazuh server, specifically the REST API and the clients that connect to it was not encrypted.

These security issues are considered minimal, since the Wazuh installation is located in the *Servers Network* (**Figure 12**) within the organisation, making it only accessible from within the VOID headquarters.

4.4.3.2 *Personal PC installation*

Since the access to the Personal Computer (PC) in the VOID headquarters was limited, it was necessary to do a second installation of the Wazuh server on another PC. In this case, the development PC for this project was used. This new installation was mainly used to carry out the process of developing the new GUI for the Wazuh system, leaving aside the monitoring capabilities of Wazuh.

The specification of the hardware and operating system for this computer were the following:

- 16 GB of RAM;
- 1 TB of HDD storage and 256 GB of SSD storage;
- i7-7700HQ CPU;
- Ubuntu 20.04.0 (x86_64) OS.

Considering that the step-by-step installation had already been carried out and the appropriate takeaways from this process had been gained, the decision to use the pre-built VM created by the Wazuh team was made.

This VM is provided in Open Virtual Appliance (OVA) format. The VM can then be imported into a virtualization software that supports this file format. For this project the *VMware Workstation 16 Pro* software was used.

The VM is configured to work out of the box, which makes it simple to use. Furthermore it has all the Wazuh components already installed and configured on it, specifically the VM has the following characteristics:

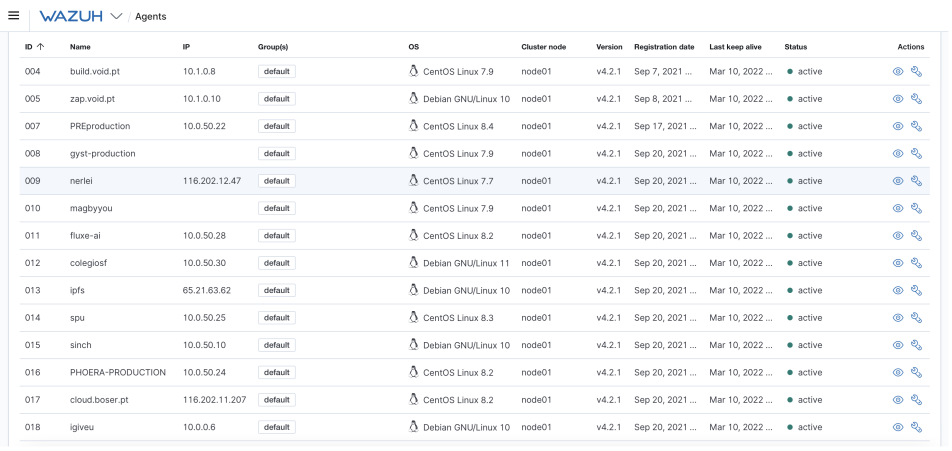
- CentOS 7;
- Wazuh manager;
- Open Distro for Elasticsearch;
- Filebeat-OSS;
- Kibana;
- Wazuh Kibana plugin.

This VM should not be used for a production environment unless secured properly, which is not a problem since it will only be used for development related purposes.

4.4.4 *Production environment*

It was decided that the production environment for the Wazuh server should be hosted within the VOID headquarters, alongside the other services which are run internally by the organisation. This means that the Wazuh system will be located in the *Servers Network* (**Figure 9**).

Before initiating the installation process, a decision was made to place the Wazuh manager in charge of monitoring other servers located within the VOID headquarters as these servers are located on the same network as the Wazuh manager. Additionally, this installation would be in charge of monitoring the pre-production and production servers which are located on the *External Network* (**Figure 12**). A list of the servers monitored by the initial Wazuh installation can be seen in **Figure 11**.



ID	Name	IP	Group(s)	OS	Cluster node	Version	Registration date	Last keep alive	Status	Actions
004	build.void.pt	10.1.0.8	default	CentOS Linux 7.9	node01	v4.2.1	Sep 7, 2021 ...	Mar 10, 2022 ...	active	
005	zap.void.pt	10.1.0.10	default	Debian GNU/Linux 10	node01	v4.2.1	Sep 8, 2021 ...	Mar 10, 2022 ...	active	
007	PREproduction	10.0.50.22	default	CentOS Linux 8.4	node01	v4.2.1	Sep 17, 2021 ...	Mar 10, 2022 ...	active	
008	gyst-production		default	CentOS Linux 7.9	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
009	nerlei	116.202.12.47	default	CentOS Linux 7.7	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
010	magbyyou		default	CentOS Linux 7.9	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
011	fluxe-ai	10.0.50.28	default	CentOS Linux 8.2	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
012	colegiosf	10.0.50.30	default	Debian GNU/Linux 11	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
013	ipfs	65.21.63.62	default	Debian GNU/Linux 10	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
014	spu	10.0.50.25	default	CentOS Linux 8.3	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
015	sinch	10.0.50.10	default	Debian GNU/Linux 10	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
016	PHOERA-PRODUCTION	10.0.50.24	default	CentOS Linux 8.2	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
017	cloud.boser.pt	116.202.11.207	default	CentOS Linux 8.2	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	
018	igiveu	10.0.0.6	default	Debian GNU/Linux 10	node01	v4.2.1	Sep 20, 2021 ...	Mar 10, 2022 ...	active	

Figure 11: Servers monitored by the Wazuh manager.

Bearing these factors in mind, it was decided to carry out an all-in-one installation, meaning that both the Wazuh manager and ELK Stack would be installed on the same machine. The Wazuh documentation states that an installation like this can monitor up to 100 hosts (Wazuh, 2021). In the case of more hosts needing to be monitored, it may be necessary to run a distributed installation of the system, where the Wazuh manager and ELK Stack are installed on separate machines, each having their own resources.

The steps carried out in the first installation (Appendix B) were followed to install the Wazuh server within the organisation's headquarters. After the manager was set up correctly, the agents were installed on all the hosts that were required to be monitored, one by one (Appendix C). Once these steps were complete, the final network architecture of the organisation was slightly different to the original, and the new architecture can be seen in **Figure 12**.

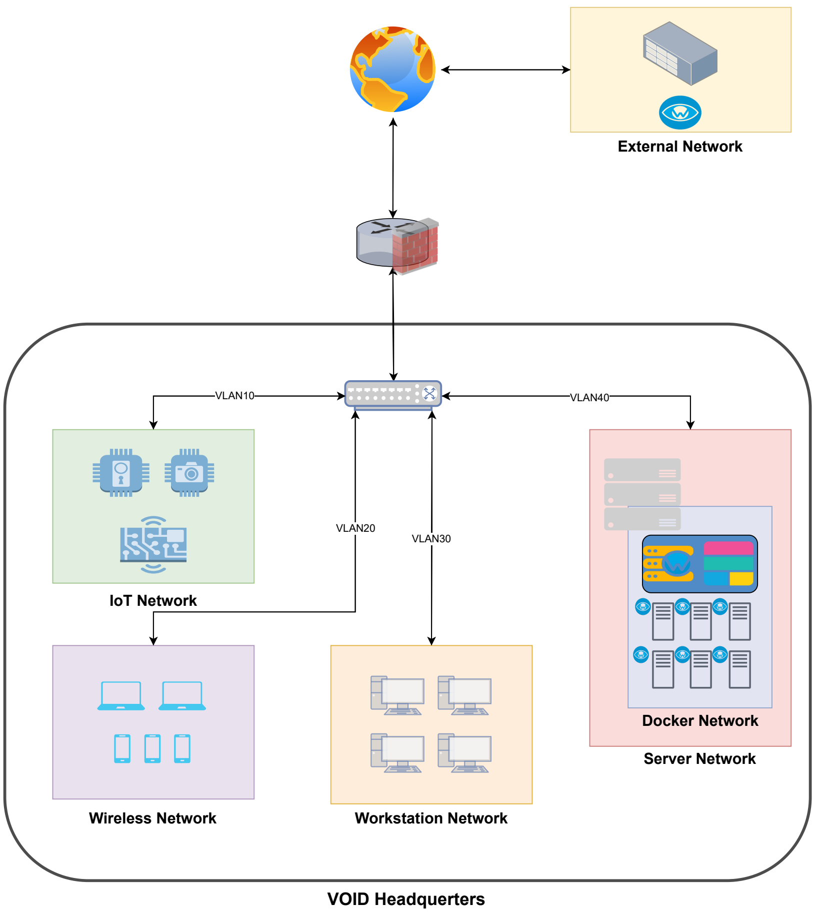


Figure 12: The network architecture within VOID after installing all the Wazuh components.

Finally, some of the security issues that were identified in **Section 4.4.3.1** needed to be solved, namely the following issues:

- Enable Hypertext Transfer Protocol Secure (HTTPS) for the communication with the REST API;
- Change the default passwords for the REST API.

4.4.4.1 *Enabling HTTPS for the Wazuh manager REST API*

To enable HTTPS for the REST API of the Wazuh manager it was necessary to do two things:

1. **Provide the necessary certificates**: When deciding what certificates to use, there was the possibility to either pick the ones created by Wazuh when the system was installed or generate our own certificates. Since the system will be placed on a network that is not internet-facing, it was decided to keep the certificates generated by Wazuh. These certificates are self-signed by a root Certification Authority (CA) created during the installation of the Wazuh system;
2. **Edit the REST API configuration**: After deciding what certificates to use, the REST configuration, which is located at */var/ossec/api/configuration/api.yaml* on the Wazuh server, was modified. Specifically the lines in **Listing 2** were changed. If any modifications were to be made to the REST API HTTPS configuration, these lines would have to be modified again.

Listing 2: Configuration example for activating HTTPS for the Wazuh API.

```yaml
https:
  enabled: yes
  key: "api/configuration/ssl/server.key"
  cert: "api/configuration/ssl/server.crt"
  use_ca: False
  ca: "api/configuration/ssl/ca.crt"
  ssl_protocol: "TLSv1.2"
  ssl_ciphers: ""
```

Finally, to apply these changes, it was necessary to restart the Wazuh manager service.

4.4.4.2 *Changing the default passwords*

The next security issue to be addressed was changing the passwords associated to users who can authenticate the Wazuh API. The default users have wazuh and wazuh-wui as their usernames, and the passwords for these users are the same as the usernames.

To be able to communicate with the Wazuh manager's API, two authentication steps are required:

1. **BasicAuth**: The API requires any user to provide a username and password to authenticate. If the credentials match those stored on the server, the server replies with a JSON Web Token (JWT);

2. **JWT Authentication**: All subsequent requests to the Wazuh API must include a header with the JWT that was issued after performing the basic authentication.

After receiving the JWT it was possible to modify the passwords for the default users, one by one, by making a PUT request to the */security/users/{user_id}* endpoint with the new password included in the body of the request.

The steps [2], the process to change the password can be seen in **Listing 3**.

Listing 3: Steps followed to change the passwords of the default users for the Wazuh API.

```
tom@grom:~$ WAZUH=https://192.168.254.128:55000
tom@grom:~$ TOKEN=$(curl -k -X GET -u wazuh-wui:wazuh-wui
↪  $WAZUH"/security/user/authenticate?raw=true")
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   271  100   271    0     0   1266      0 --:--:-- --:--:-- --:--:--  1266
tom@grom:~$ echo $TOKEN
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ3YXp1aCIsImF1ZCI6IldhenVoIEFQSSB
↪  SRVNUIiwibmJmIjoxNjQ2OTk1MjE1LCJleHAiOjE2NDY5OTYxMTUsInN1YiI6IndhenVoLXd1aSI
↪  sInJ1bl9hcyI6ZmFsc2UsInJiYWNfcm9sZXMiOlsxXSwicmJhY19tb2RlIjoid2hpdGUifQ.bAlD
↪  jIuPdro5ZUxaAlFIPw_xAkZZgKvL7BXG6XADrl0
tom@grom:~$ curl -k -X GET $WAZUH"/security/users?pretty=true" -H "Accept:
↪  application/json" -H "Authorization: Bearer ${TOKEN}"
{
   "data": {
      "affected_items": [
         {
            "id": 1,
            "username": "wazuh",
            "allow_run_as": true,
            "roles": [
               1
            ]
         },
         {
            "id": 2,
            "username": "wazuh-wui",
            "allow_run_as": true,
            "roles": [
               1
            ]
         }
      ],
      "total_affected_items": 2,
      "total_failed_items": 0,
      "failed_items": []
   },
```

2 These steps were reproduced in a test environment for demonstration purposes. Since the commands were not copied when the initial installation was done

```
   "message": "All specified users were returned",
   "error": 0
}tom@grom:~$curl -k -X PUT $WAZUH"/security/users/1?pretty=true" -H
↪  "Content-Type: application/json" -H "Authorization: Bearer ${TOKEN}" -d '{
↪  "password": "N3wSecureP$$wd" }'
{
   "data": {
      "affected_items": [
         {
            "id": 1,
            "username": "wazuh",
            "allow_run_as": true,
            "roles": [
               1
            ]
         }
      ],
      "total_affected_items": 1,
      "total_failed_items": 0,
      "failed_items": []
   },
   "message": "User was successfully updated",
   "error": 0
  }
```

4.5 REQUEST TO THE WAZUH API

The development of the requests to the Wazuh manager was the first stage of the actual development of the GUI application. During this stage requests to all the API endpoints provided by the Wazuh server were created. The implementation of such requests enabled the application to: get information from the Wazuh manager; and perform actions on the resources managed by the Wazuh server, which in later stages will enable the user to interact with the Wazuh manager when using the GUI. In addition to providing an interface to communicate with the Wazuh server, this stage also helped several aspects of the Wazuh system to be understood:

- A general idea of the different capabilities that the Wazuh manager offers;
- An overview of how the manager handles authentication for the REST API it provides;
- The different entities that can be manipulated by the manager;
- The configurations that can be added/removed to modify the manager and agents functionalities as needed;

- The different groups, configurations, users, agents and agent configurations that the server manages;
- The data that is exchanged between agents, manager and the UI.

These insights then helped during the development process, specifically when developing the UI more effectively, by providing a better understanding of the capabilities it would have to possess.

Since the API is the main communication point for most of the operations executed by the Wazuh manager, it was decided to implement this part first so that it would be possible to focus on the GUI development in later stages.

4.5.1 *Entities implemented to communicate with the Wazuh manager*

To fulfil the objective of communicating with the Wazuh manager four types of entities were implemented. These four types were: requests, parameters, request body schemas and enumerates. Each of these types of entities are used to build the requests that are used to communicate with the Wazuh manager, which then replies in JSON format (unless otherwise specified). Finally, when a response is received from the manager, it is processed by the method that makes the request and a JSON object is returned to the method caller.

In the following sub-sections, we will have a closer look at the four types of objects used to communicate with the Wazuh server.

4.5.1.1 *Requests*

The requests package contains a set of classes which are mapped to each kind of operation that can be performed on the manager. All the classes in this package are abstract and only contain static methods. Such methods are mapped to the actions that can be performed by the Wazuh server on the entities it manages, making it easier to identify and invoke required actions. An example of one of these classes can be seen in **Listing 4**.

Listing 4: Example of a class used to perform active response actions, specifically running a command on one or more agents

```
package com.voidsoftware.wazuhFX.util.requests;

import com.voidsoftware.wazuhFX.models.parameters.query.QueryParameter;
import com.voidsoftware.wazuhFX.models.requestBodySchemas.ActiveResponseCommand;
```

```java
import com.voidsoftware.wazuhFX.util.enums.ContentType;
import com.voidsoftware.wazuhFX.util.enums.WazuhAPIEndpoint;
import kong.unirest.json.JSONObject;

public abstract class ActiveResponse extends WazuhRequest{
    /*Params: P: ; Q:agent_list, pretty, wait_for_complete  ; RBS:
    ↪  ActiveResponseCommand.java */
    public static JSONObject runCommand(QueryParameter queryParameters,
    ↪  ActiveResponseCommand command) {
        return verifyResponseAndReturnBody(makePUTRequest(WazuhAPIEndpoint.ACTIV
        ↪  E_RESPONSE.getPath(),
        ↪  null,
                queryParameters.getParameters(), command, ContentType.APP_JSON));
    }
}
```

Each class in the requests package extends the base request class *WazuhRequest*. This class is responsible for configuring and initializing the Hypertext Transfer Protocol (HTTP) client, which is implemented by using the Unirest library (Kong, 2022).

Furthermore the *WazuhRequest* class provides implementations for performing GET, POST, PUT and DELETE requests asynchronously. These methods are used by the classes that inherit from this class to interact with the manager's REST API.

Finally the *WazuhRequest* class implements methods to verify and process the HTTP responses from the manager and returns a JSON object [3] containing the relevant response information, i.e. whether the action performed has been successful or failed on the server's side.

This implementation provides a way to change the underlying HTTP client if necessary.

4.5.1.2 *Parameters*

The parameters package contains all the necessary classes to instantiate parameters used by the requests to the Wazuh manager. Such parameters can either be query parameters or route parameters. Parameters can be passed to the request made by the HTTP client by using a Map data structure, where the key of the Map is the parameter's name (String) and the value is the parameter (Object). Due to this reason, all parameters extend the *WazuhParameter* base class, which defines the

3 Implemented by the Unirest library.

parameters HashMap and provides a method to get such HashMap while removing the unnecessary entries (Null entries).

Different strategies are used to create these parameters, depending on the type of each of these (route or query). The strategies and implementations are described as follows:

- **Route Parameters**: Route parameters are used to specify what resource should be accessed when performing a request (e.g.: supplying the ID of an agent to get its configuration). The Wazuh manager has endpoints that require up to 3 route parameters. For this reason the *RouteParameter* class has 3 constructors, for supplying either one, two or three route parameters. This design is similar to the telescoping pattern (Examples, 2022). As such each constructor receives a key value pair that then will be added to the parameter Hashmap and subsequently replaced within the route parameters of the endpoints URL;
- **Query Parameters**: Query parameters are used to fine-tune any requests made to the Wazuh manager. Fine tuning may include, though is not limited to, changing the format of the output, limiting the range of results, getting results for a specific resource or sorting the results.

  Initially the implementation of these query parameters was similar to the implementation of the route parameters. Such an implementation was not the most effective for the query parameters since the number of variables used by these kinds of parameters is much greater than the ones used by the route parameters. This was a problem since it was necessary to have a constructor for each new attribute required by each query parameter entity, which resulted in large number of constructors which, in turn, made instantiating a query parameter confusing. Another problem that arose from this implementation, was that when any of the query parameters were not necessary, it had to be set to null, which made the creation of certain parameters even more confusing.

  For this reason it was decided to implement the creation of query parameters through a builder like pattern (HowToDoInJava, 2022). As such to add parameters to a query parameter object, it is only necessary to create an instance of the *QueryParameter* class and add the required parameters by using the parameter name (similar to setters) as can be seen in **Listing 5**. This final solution is more verbose than the previous one, which increases the lines of code. On the other hand, it makes the code more readable and reduces ambiguity when compared to using constructors.

As implemented with requests in this project, there is a class for the parameters of each kind of request, making it easier to decide which parameters to use. When parameters are common among several types of requests, the methods to add such parameters are extracted to the *QueryParameters* base class.

Listing 5: Example for creating a query parameters for a task.

```
Map<String, Object> a = new TaskParameter().command("test123").pretty(false)
↪  .offset(1).getParameters();
```

4.5.1.3 *Request Bodies*

This package contains all the classes that will be used when sending data to the Wazuh Manager. Each class contains the required attributes to either create or update a resource on the Wazuh server. The classes are serialised automatically by the Unirest library. By default Unirest uses the Gson (Google, 2022) object mapper for this purpose, but when implementing the Unirest client it was configured to use the Jackson (FasterXML, 2022) library instead. The decision to use the Jackson library for the purpose of serialization was taken since it is the same one used by the elastic search REST client.

The Wazuh manager requires two forms of data encoding in a request's body, depending on the request, it may accept JSON and binary strings. Because of this, the functions used to call HTTP methods can be configured to include headers defining the content type as required.

4.5.1.4 *Enumerations*

The enums package, in the case of the request to the Wazuh server, contains all the constant strings that are used to perform requests, an example of such strings are the endpoints on the Wazuh manager.

4.6 DATA STORAGE MIGRATION

As seen previously, Wazuh uses Elasticsearch as its database to store all the information generated by the Wazuh server. According to the requirements of this project, it is necessary to remove this storage solution and replace it with Apache

Solr, which is similar to Elasticsearch since it is also based on Lucene indexes. To achieve this, a set of steps were followed. In this section we will look over the main actions undertaken to perform an initial migration of the storage solution.

There are several difficulties that needed to be overcome to perform the migration to the Solr database. The first was to decide whether to install Solr on the same or a different server than that on which Wazuh was installed. Subsequently, it was necessary to obtain data that could be stored within the Solr database. In addition to retrieving such data, it was necessary to create the structure of the database so the information that had previously been collected could be stored. Finally, it was necessary to implement the methods within the GUI application to access the information which was stored in the Solr database.

4.6.1 *Setup*

Installing Solr was the first step in this portion of the project. To do this, it was necessary to decide whether Solr should be installed on the same machine as the Wazuh manager or on a different machine. In this case, for development purposes, the decision was made to install Solr on the same Machine where the Wazuh manager was installed. Although this was a good solution for the development stage of this project, it would not be the optimal solution for a production environment. As such, separating the components of the solution when setting up the production environment is recommended.

As can be seen in the Solr documentation (Solr, 2022a), installing the Solr server for development purposes is a simple process. The steps that must be followed to install Solr on a Linux machine and run it in cloud mode are as follows:

1. Navigate to https://solr.apache.org/downloads.html and fetch the latest release;
2. Extract Solr to the directory where the installation is desired (e.g. /opt/);
3. Add the Solr directory to the $PATH variable so it can be used system-wide;
4. Run Solrs cloud example setup: solr -cloud -force

Once these steps were completed, it was possible to move on to the next stage of the data storage solution migration.

4.6.2 *Sample data*

For Solr to be able to store the information that is generated by the Wazuh manager, it was first necessary for Solr to understand the information it would receive. In this section we will explore where Wazuh stores the logs and alerts it generates. to then understand the format in which this data is stored.

Wazuh among other things stores all the logs that are sent to it and the alerts that it generates when a rule is triggered. Furthermore, Wazuh can be configured to store (or not) such information to accommodate the necessities of the organisation and the amount of storage available. Wazuh stores the alerts that it generates in the */var/ossec/logs/alerts/* directory.

Since we know where the Wazuh manager stores the alerts it generates, it was possible to retrieve alerts from this directory. Such alerts were then used to create the Schema which Solr then used to index data. An example of an alert can be seen in **Listing 6**.

Listing 6: Example of an alert generated by the Wazuh manager.

```
{
  "timestamp": "2021-11-17T11:15:21.939+0000",
  "rule": {
    "level": 3,
    "description": "Ossec agent started.",
    "id": "503",
    "firedtimes": 1,
    "mail": false,
    "groups": ["ossec"],
    "pci_dss": ["10.6.1", "10.2.6"],
    "gpg13": ["10.1"],
    "gdpr": ["IV_35.7.d"],
    "hipaa": ["164.312.b"],
    "nist_800_53": ["AU.6", "AU.14", "AU.5"],
    "tsc": ["CC7.2", "CC7.3", "CC6.8"]
  },
  "agent": { "id": "001", "name": "grom", "ip": "192.168.1.103" },
  "manager": { "name": "wazuh-manager" },
  "id": "1637147721.0",
  "full_log": "ossec: Agent started: 'grom->any'.",
  "decoder": { "parent": "ossec", "name": "ossec" },
  "data": { "extra_data": "grom->any" },
  "location": "ossec"
}
```

As can be seen in **Listing 6** the Wazuh server stores relevant information for an alert in JSON format. Each alert stores the following information:

- **Timestamp**: The date and time when the alert was triggered;
- **Rule**: The rule that triggered the alert, which contains different metadata associated to the alert such as the level of the rule, a description of the alert, the ID of the rule, how many times the rule was triggered and finally information associated to the regulation requirements associated to the rule;
- **Agent**: Information about the agent on which the alert was triggered;
- **Id**: A unique ID to identify the alert;
- **full_log**: The full log message which was sent by the agent;
- **Decoder**: Information about the decoder in charge of interpreting the information sent by the agent;
- **Data**: Any extra information that might be necessary to describe an alert;
- **Location**: The location on the agent of the log which was sent to the manager.

After gathering a set of alerts from the Wazuh server and understanding the format of these alerts it was possible to continue with the creation of the schema that stores this information.

4.6.3 *Schema creation and data import*

Solr describes the structure of a schema through a schema file, within this file there are definitions for field types and fields that Solr can understand, thus describing the structure of the documents it can index. There are several ways of creating a schema for Solr to use. During this stage of the project multiple ways of creating the Solr schema were attempted, and one was selected for its simplicity and effectiveness in the end.

4.6.3.1 *Schema Designer*

This tool is part of the Solr web interface. It provides a interactive method for designing a new schema based on sample data (Solr, 2022b). To use this tool, it is first necessary to create a new Schema by copying the *_default* schema, which is included within the Solr installation. When the new schema is created, sample data can either be copied into a text field or uploaded from a file. Solr evaluates this data and uses it to create the fields that belong to the schema. After Solr concludes this process, it is possible to modify the fields that were created automatically by

Solr to better fit the data that has been uploaded. If necessary, more data can be uploaded to the schema designer to verify if the schema accepts such data

Initially this approach seemed to be the easiest way to create a schema that would work for the data which is created by the Wazuh manager. In practice, though, it did not work very well, since the schemas that were being auto-generated by Solr did not match the sample data that was initially provided. Furthermore, the new schema produced errors when importing more of the sample data.

4.6.3.2 *Writing the schema file based on Elasticserch mappings*

Out of the box Wazuh uses Elasticsearch as its storage solution. For Elasticsearch to be able to understand the data which is created by Wazuh, it uses mappings [4] created by the Wazuh team. These mappings can be found on the Wazuh server or downloaded directly from the Elasticsearch cluster that is run by Wazuh.

This appeared to be a good approach to create the new schema for Solr, since, in this way it would be an exact copy of the schema that is implemented and already works to store the data created by Wazuh. Despite being an optimal solution, this is not a very practical one. Mappings are represented in JSON format and describe the properties of the information an index will store. The mappings created by the Wazuh team are around 14000 lines long, so translating this to the XML format required by Solr would be a difficult task. Furthermore, the syntax used by Elasticsearch and Solr to describe the data which is stored by their indexes is different. As such this solution was not adopted.

4.6.3.3 *Using sample JSON alerts to create the schema*

Solr provides API endpoints to upload data in different formats to its indexes. Furthermore, there are specific endpoints which can be used to upload JSON data, such as the one generated by the Wazuh server **Section 4.6.2**. One of the endpoints that is provided to upload JSON data (*/update/json/docs*), allows parameters to be added which "provide information on how to split a single JSON file into multiple Solr documents and how to map fields to Solr's schema" (Solr, 2022d).

Using this functionality, it was possible to upload examples of alerts generated by the Wazuh server to create the schema automatically. This approach is useful for development purposes, since it provides an easy way to create a schema which supports and works with the information that is generated by the Wazuh server.

4 Mappings are the same as a schema for Elasticsearch.

After following this process, it was possible to create an initial Schema for the alerts created by the Wazuh server (**Apendix A**). In future iterations of this project, this schema should be reviewed and the Solr configuration should be optimised to work just as well as the current data storage solution of the Wazuh system.

4.6.4 *Querying Data*

Finally, after performing this initial migration of the data storage, it was necessary to be able to fetch the information from the Solr instance. For this purpose, SolrJ was used.

SolrJ (Solr, 2022c) is a library which makes it easy for any application written in Java to communicate with a Solr instance. SolrJ provides simple, high-level methods to interact with most of Solrs APIs.

Adding SolrJ to a Java project is straightforward, using Maven in the case of this project. As such it is necessary to add the dependence seen in **Listing 7**.

Listing 7: Maven dependency to add SolrJ to a Java project.

```xml
<dependency>
  <groupId>org.apache.solr</groupId>
  <artifactId>solr-solrj</artifactId>
  <version>8.7.0</version>
</dependency>
```

Within the GUI application, a wrapper for the SolrJ client was created with the purpose of configuring the client before performing requests to the Solr instance. The configuration process involved setting the endpoints to which the client would make the requests and the timeout value for waiting on responses.

Once the client is configured, it was possible to perform requests to the Solr instance, to fetch the required data for each view. An example of a request can be seen in **Listing 8**.

Listing 8: Example Java code for fetching data from a Solr instance.

```java
SolrQuery query = new SolrQuery("*:*")
    .addFacetQuery("*:*")
    .addFacetQuery("rule.level:[7 TO *]")
    .addFacetQuery("rule.groups:authentication_success")
    .addFacetQuery("rule.groups:win_authentication_failed authentication_failed
    ↪  authentication_failures")
    .setRows(0);

QueryResponse response4 = client.query(query);
```

4.6.5 *Data used for the development of the project*

There were two main sources of information (data sets) used for this project. The first was the data collected from the organisation's machines by using the Wazuh agents. The second was the example dataset of logs provided by the Wazuh team. The latter was necessary since it was not guaranteed that working from the office would be possible because of the current pandemic situation.

For the development of the project, only log data from Windows, Mac OS and Linux systems was used, and these logs were formatted in the native log format for each of these systems. Subsequently Wazuh used its set of decoders to normalise such logs, to then provide meaningful information about each system which could then be represented by using different formats within the GUI. The logs used for the development of the project both included normal logs (i.e.: logs of normal system functions) and logs that represent adversary security events.

In the case of the logs collected within the VOID organisation, there were some restrictions. The restrictions are due to the nature of the logs. Since these belong to the different hosts and servers within the organisation, they may contain either sensitive information such as information about the operating system, programs installed on the system or even vulnerabilities that are present on a system, which is information that could reveal how certain processes are carried out by different hosts.

Finally, all data collected throughout the development of the project, as requested by VOID, was stored in case it is necessary for future use.

4.7 WAZUHFX DEVELOPMENT

The development of the GUI fulfils an important objective of this project, which is to remove the Kibana plugin that has already been developed by the Wazuh team and replace it with a new application, in this case WazuhFX.

During the development of the GUI application two tasks were carried out. The first task was to perform an analysis on the already existing Kibana dashboard, the goal of which was to identify the core functionalities of this application to then define which of these should be implemented within the new GUI. The second task was the development of the WazuhFX application.

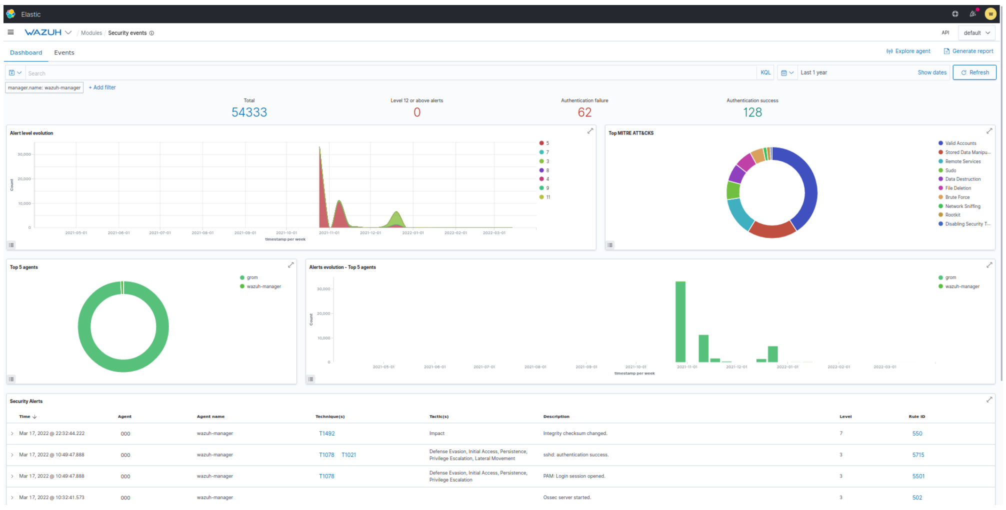
4.7.1 *Kibana dashboard analysis*

Wazuh uses Kibana alongside a dedicated plugin as its UI to view the information generated by the Wazuh manager which is stored in Elasticsearch indexes. This plugin doesn't only present such information, but also has the capability to communicate with the Wazuh manager's REST API, allowing users to administer the different entities and configurations which are stored on the manager.

The plugin is divided into different windows that correspond to actions that can be taken and data that can be accessed from the interface. These windows are:

- **Modules**: The modules includes a menu, from which the user can view and analyze information related to the alerts generated by the different modules of the Wazuh manager, an example of such view can be seen in **Figure 13**. In other words, this window gives a summary of the data that can be accessed, which is related to: the security events; the auditing and policy monitoring results; the threats which could be present on a system; and information related to regulatory compliance;
- **Management**: The management window allows the user to administer different attributes on the Wazuh manager and also provides information about the status of the Wazuh instance. In this window a user can select to manage the rules, decoders, groups and configurations present on the Wazuh manager;
- **Agents**: The agents window presents a list of all the agents which are registered to a manager, as well as presenting different information related to such agents. It is also possible to access a agent's summary information and configuration files from this window;

- **Tools**: This window includes tools that aid the process of working with Wazuh. It presents a tool to interact with the Wazuh API and another to test rules before creating them;
- **Security**: The security windows allows the user to create, edit or delete users, roles and policies, to limit the actions that can be performed on the Wazuh manager;
- **Settings**: The settings window allows a user to modify a set of configurations on the Wazuh manager or configurations related to the ELK Stack.



Total 54333 | Level 12 or above alerts 0 | Authentication failure 62 | Authentication success 128

Security Alerts

Time	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Mar 17, 2022 @ 22:32:44.222	000	wazuh-manager	T1492	Impact	Integrity checksum changed.	7	550
Mar 17, 2022 @ 10:49:47.888	000	wazuh-manager	T1078 T1021	Defense Evasion, Initial Access, Persistence, Privilege Escalation, Lateral Movement	sshd: authentication success.	3	5715
Mar 17, 2022 @ 10:49:47.888	000	wazuh-manager	T1078	Defense Evasion, Initial Access, Persistence, Privilege Escalation	PAM: Login session opened.	3	5501
Mar 17, 2022 @ 10:32:41.573	000	wazuh-manager			Ossec server started.	3	502

Figure 13: Security alerts presented in the Kibana plugin.

Within each of these windows there are sub sections for more specific actions. In the case of more complex actions, such as visualizing data or editing entities on the manager (Rules, decoders, groups and configurations), dedicated windows exist for these actions. In the cases of more simple tasks there are tabs for each sub-section.

One of the main objectives of this project is to develop a GUI for the Wazuh system using JavaFX. Consequently, to initiate the development of the project it was necessary to define the initial goals for developing this new GUI. As such, the goal of reproducing the main capabilities of the Wazuh dashboard was set. The initial step to accomplish this goal was to perform an analysis on a group of the main screens of the Wazuh plugin for Kibana.

4.7.1.1 *Login screen*

Kibana requires a user to authenticate before being able to access its web interface. Because of this, the initial screen presented to a user when accessing the IP or

URL of the interface is the login screen. To further use the application, the user is required to provide a username and a password, which are then verified against those stored by Kibana.

4.7.1.2 *Data presentation*

A useful feature of the Wazuh plugin is its ability to present information stored in the Elasticsearch indexes in different formats. This is an important aspect of the tool since it grants visualization capabilities upon the information that has previously been created from the data that is processed on the Wazuh manager.

Information is presented in several ways depending on the kind of information it has. Generally, information is presented in three ways:

- **Summary of information** is represented in terms of key values. Key values are numbers that describe the number of times an event has occurred. Some examples of these values are total number of events, number of disconnected agents and number of alerts with a value above a threshold. The summary of information gives the user a quick overview of the data that has been collected up to a certain point in time;
- **Graphical information** is represented by graphics. Graphics present information to the user in a visual format, making it easier to get a general overview of the state of the information as a whole. Furthermore, graphics help categorise the information based on different parameters, relate information on certain values, compare information against other data and correlate different pieces of information to mention just a few;
- **Concise information** is represented in tables where all the information about an alert can be found. Concise information allows the user to look in detail into a specific event or events. When information is presented in a tabular format, key data points are shown (for example timestamp, agent and description) to help identify the event which is being searched for. When an event of interest is identified a user can click on such an event to get all of the information about the event.

In addition to providing ways of viewing information, the UI allows a user to search through the information using the Kibana Query Language (KQL) or Lucene syntax. It also provides methods to filter data based on the timestamps of the alerts or other fields that are associated with an event.

4.7.1.3 *Server management*

A useful feature of the Kibana plugin, is the ability to communicate with the Wazuh REST API. This allows the application to administer the resources that are stored on the Wazuh manager or check the status of the server, giving the user the ability to interact with the serverdespite not having direct access to it.

The entities that can be modified from the Kibana interface are: Rules, Decoders, Constant Database (CDB) lists, Groups and configurations. Each of these entities is stored on the Wazuh server, thus the web interface allows a user to manage the underlying entities. Entities can either be managed locally on the Wazuh server or through the Kibana plugin. The latter option offers two possibilities: upload files with the configurations; or manage the configuration through the web interface.

Entities are presented to the user in a list format where the user can read a description of the entity and the file to which it is associated. Wazuh has pre-created entities which can only be viewed by users. On the other hand, users can create their own files, in which case the user has full permissions (read, write, delete) over these files. The following entities can be managed by a user:

- **Rules and decoders** are entities stored on the Wazuh manager used to process and create alerts from logs. These entities are defined in XML markup;
- **CBD lists** are regular text files that contain key value pairs[5];
- **Agent groups** are folders containing configuration and other files for a specific agent group;
- **Configuration files** are XML files containing the configurations for either agents, a group of agents or the Wazuh manager.

It is also possible to search through these entities with a search bar, making it easy to find entities when needed.

4.7.1.4 *Listing agents and configuring agents*

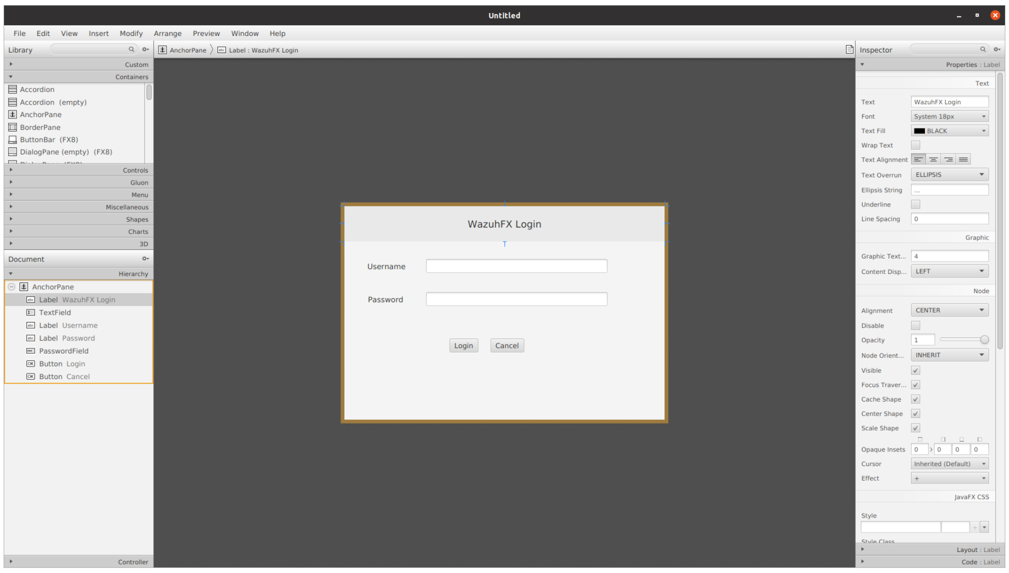
Since the Wazuh server usually communicates and manages several agents simultaneously, it is important to keep track of those agents. Agents are listed by the Kibana plugin, presenting different information related to each of them. By clicking on an item in this list, it is also possible to manage an agent's configuration and access specific information related to that agent.

5 The format of the key values is - Key:Value. Values are optional

4.7.2 *Development*

When developing the GUI application, the pieces that had to be created and put together for each view can be generally separated into two categories: the visual components; and the logic components. As such during the development process of the WazuhFX application, separate workflows for producing the elements in each of these categories were followed:

- **Graphical components**: The structure of a graphical component in JavaFX is described by using FXML (Section 2.1). In the case of this project the scene builder tool was used to create the markup for the different views, an example of such usage can be seen in **Figure** 14. All views were created by using specific components that can be used to arrange other components, present information or receive inputs from the user;
- **Logic Components**: The logic behind each view of the application was done in Java code. The structure of the code was done following the MVVM pattern (Section 2.2).



AnchorPane
- Label WazuhFX Login
- TextField
- Label Username
- Label Password
- PasswordField
- Button Login
- Button Cancel

Text
Text: WazuhFX Login
Font: System 18px
Text Fill: BLACK
Wrap Text: ☐
Text Overrun: ELLIPSIS
Ellipsis String: ...
Underline: ☐
Line Spacing: 0

Graphic
Graphic Text...: 4
Content Disp...: LEFT

Node
Alignment: CENTER
Disable: ☐
Opacity: 1
Node Orient...: INHERIT
Visible: ✓
Focus Traver...: ✓
Cache Shape: ✓
Center Shape: ✓
Scale Shape: ✓
Opaque Insets: 0 0 0 0
Cursor: Inherited (Default)
Effect: +

JavaFX CSS
Style
Style Class

Figure 14: Scene builder being used to create the login view.

During the process of creating a new view for the application, the first step was to define the layout for the components that will be presented to the user. Because the initial objective during the development phase of this project was to replicate the functionalities of the Wazuh Kibana plugin, it was not necessary to define such a layout for the views and the layouts from the plugin were used as a guide.

In the following sections, the process carried out to create each view within the WazuhFX application will be summarised.

4.7.2.1 *Creating the view*

After defining the layout of the view, the next step was to create it by using the components that were available. As mentioned previously, the Scene Builder tool was used to accomplish this. This tool allows users to drag and drop the desired components onto a canvas where they can be previewed. Scene builder allows the user to create a view by using default JavaFX components. Additionally, other libraries can be imported, adding components and functionalities to the initial set of components defined by JavaFX.

Two types of elements were used to create a view for this application:

- **Layout components** were used to organise other components included within view. These components provide typical arrangements for visual components such as rows, columns and tiles. Thus, these components serve as containers which arrange their contents depending on the properties set;
- **UI controls** are the elements which a user can view and interact with. As such UI controls are used to present information and receive inputs from a user. For these controls to be viewed it is necessary to place them within a layout component.

All the views within the WazuhFX application are added or removed from the root container, which is the main layout component for the different views within the application. The layout of the root container is defined as a *BorderPane*, and this layout divides the screen into 5 sections: top; bottom; left; right; and center. As can be seen in **Figure 15**. The top section of the *BorderPane* corresponds to the toolbar of the application and is static. The left section corresponds to the actions that can be taken from within the application (Section 4.7.1), and this section is also static. Finally the central section of the *BorderPane* contains the content of the window which the user is viewing, and the content of this section is not static, changing depending on the user's interactions.

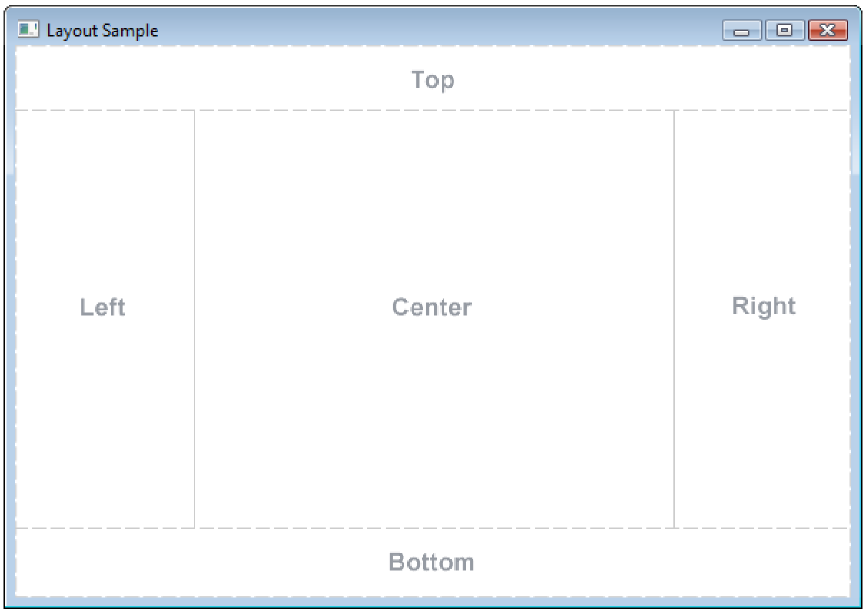


Figure 15: The BorderPane layout.

The navigation between the different views of the GUI application is performed by the *ViewController*. This controller uses the path of a FXML file to then set the center of the *BorderPane* with the contents of the FXML file. Additionally, the controller keeps a cache and a history of the views that have already been accessed, so that the files associated to each of these views do not have to be loaded every time a view is visited again and to allow the user to go back to views that have previously been visited. The code in charge of switching from view to view can be seen in **Listing** 9.

Listing 9: Code used to switch between different FXML views.

```java
public static void switchTo(ApplicationView view){
    Parent center;
    try {
        if (cache.containsKey(view)) {
            center = cache.get(view);
        }
        else {
            center = FXMLLoader.load(ViewController.class.getResource(view.getFi
            ↪  leName()));
            cache.put(view, center);
        }
        history.push(view);
        root.setCenter(center);
    } catch (IOException e) {
        //Handle exceptions
        e.printStackTrace();
    }
}
```

Each view has a main layout which defines how the view looks. The layouts that were mainly used for the views of the application were:

- **VBox**: This layout allows a set of nodes to easily be arranged in a column;
- **HBox**: This layout allows a set of nodes to easily be arranged in a row;
- **GridPane**: This layout allows us to create a grid with any number of rows and columns, where nodes can be arranged based on the coordinates of such grid and can span any number of cells;
- **TilePane**: This layout allows us to create a grid where each cell has the same size, the contents of this container are then wrapped either horizontally or vertically depending on the configuration for the tile pane;
- **AnchorPane**: This layout allows a node or set of nodes to be anchored to the top, left, bottom, right or center of the view. The nodes will then maintain the position that was set when the windows resizes.

For the view to then present information or receive inputs from the user, UI controls needed to be added to the view. These controls are components such as buttons, labels, text boxes and list views. There are many other UI controls that were used for this project, all of which were either part of the JavaFX library or belong to one of the libraries that were added to the project: controlsFX (ControlsFX, 2022); materialFX (Parisi, 2022); and tilesFX (Grunwald, 2022). These libraries add UI further controls in addition to those that already exist in the JavaFX library.

4.7.2.2 *Adding the logic*

Once a view was setup using Scene Builder, it was necessary to add the logic which augments functionalities to the view. Since the MVVM pattern was used during this project, three different files for each view were created to add the logic.

The view controller is a Java class that is directly associated to the FXML representation of the *View*. The FXML file references the controller through one of its properties. Furthermore, the view controller can access the UI controls and layouts of the FXML view through their IDs which are also set as properties within the FXML file. The main purpose of this file is to have reference to the UI components so they can be used within the Java code, providing the ability to modify their properties and values. As such the view controller and the FXML file together represent the *View*. Finally the view controller contains a reference to its corresponding *ViewModel* from which it can fetch information to display in the view or to which it can send information to update the underlying models. This data exchange is set up within the initialise method of the controller, by binding to the property variables of the *ViewModel*. An example of the view controller can be seen in **Listing 10**.

Listing 10: Example Java code of the controller associated to a FXML file.

```java
public class ModulesMenuController extends Controller {
    private final ModuleMenuViewModel moduleMenuViewModel = new
    ↪  ModuleMenuViewModel();
    @FXML
    private Label labelTotalAgents;
    @FXML
    private Button btnSecurityEvents;

    ...

    @FXML
    private void initialize() {
        labelTotalAgents.textProperty().bind(moduleMenuViewModel.totalAgentsProp⌋
        ↪  erty());

        ...

        btnSecurityEvents.setOnAction(actionEvent ->
        ↪  ViewController.switchTo(ApplicationView.VIEW_SECURITY_EVENTS));
        moduleMenuViewModel.getData();
    }
}
```

The *ViewModel* is a Java class which is responsible for keeping the state of the *View* and synchronizing it with the *Model*. To accomplish this the *ViewModel* model uses three different strategies: JavaFXs property variables; a reference to the pub/sub Notification implementation; and a reference to the *Model* of the view. The JavaFX property variables are used to store the information that is generated by the *Model* and update the visual components of the *View*, which is done by using the bind method of the property variables. The *ViewModel* subscribes to the Notification singleton so that it can know when the value of the model has changed and update its values accordingly. Finally, it uses the reference to the *Model* of the view to be able to invoke the methods that can be used to store or fetch information and to retrieve these values. An example of the *ViewModel* can be seen in **Listing 11**.

Listing 11: Example Java code used to synchronise the *View* and the *Model* by using the *ModelView class.*

```java
public class ModuleMenuViewModel {
    private final StringProperty totalAgents = new SimpleStringProperty("");

    ...

    private final Notifications notifications = Notifications.INSTANCE;
    private final AgentsSummaryModel moduleMenuModel = new AgentsSummaryModel();

    private final Service<Void> getDataCommand = new Service<Void>() {

        ...

        //Call moduleMenuModel.fetchSummary() on a separate thread

        ...
    };

    public ModuleMenuViewModel() {
        notifications.subscribe(ApplicationEvents.UPDATE_MODULES_MENU_VIEW.getEv
        ↪  ent(), this,
        ↪  this::update);

    }

    private void update(String event) {
        moduleMenuModel.getAgentsSummary().ifPresent(
                agentsSummary -> {
                    totalAgents.set(String.valueOf(agentsSummary.getTotal()));

                    ...
                }
        );
    }
    public StringProperty totalAgentsProperty() {
        return totalAgents;
    }

    ...

    public void getData() {
        getDataCommand.restart();
    }
}
```

The *Model* is a Java class which has the ability to either persist the data that is generated through the inputs of the application or retrieve all the data that will be presented through the view. To achieve this the *Model* interacts with the

Wazuh REST API through the *managerCommunication* module (Section 4.5). Since the *Model* has no references to the *View* or the *ViewModel* it uses a pub/sub implementation to notify other components when data has been modified or fetched successfully. The *Model* is also responsible, when necessary, for mapping JSON objects to Java classes or vice versa. To accomplish this, the Jackson object mapper was used. An example of the *Model* can be seen in **Listing 12**.

Listing 12: Example of the Java code used to fetch information with the *Model* class.

```java
 public class AgentsSummaryModel {
    private final Notifications notifications = Notifications.INSTANCE;
    private  Optional<AgentsSummary> agentsSummary = Optional.empty();

    ...

    public Optional<AgentsSummary> fetchSummary() throws JsonProcessingException
    ↪  {
        QueryParameters qp = new QueryParameters();
        JSONObject data =
        ↪  Agents.summarizeStatus(qp).getJSONObject(ResponseKey.DATA.getKey());
        ObjectMapper mapper = new ObjectMapper();
        AgentsSummary as = mapper.readValue(data.toString(),
        ↪  AgentsSummary.class);
        agentsSummary = Optional.of(as);
        notifications.publish(ApplicationEvents.UPDATE_MODULES_MENU_VIEW.getEven⌋
        ↪  t());
        return Optional.of(as);
    }
}
```

4.8 WAZUHFX RESULTS

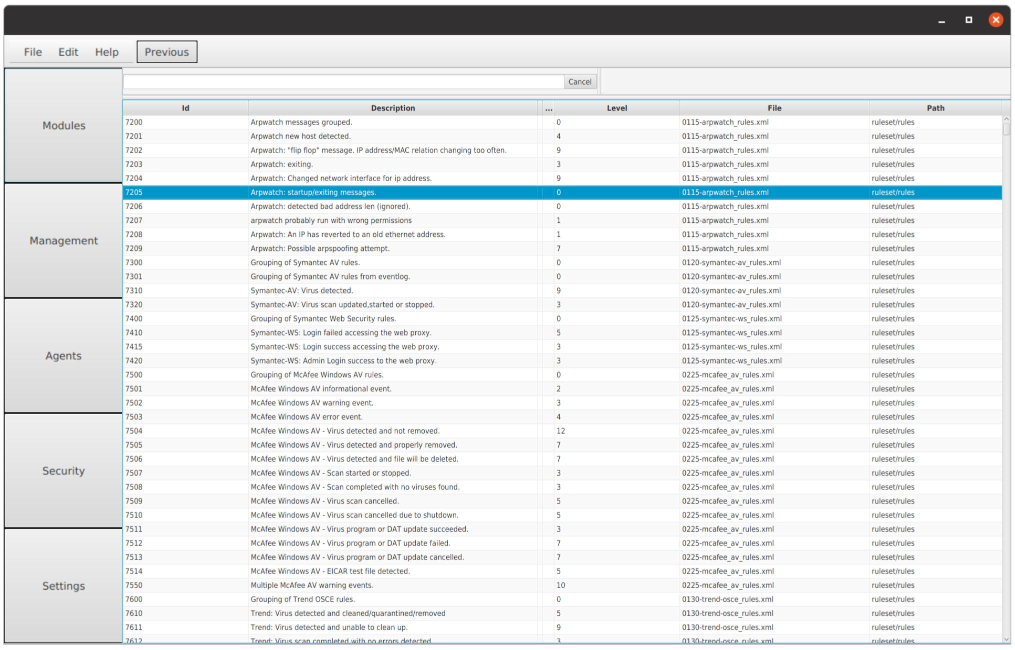
The application produced during the internship can be used on Linux, Mac OS and Windows machines, as long as they have the Java runtime environment installed on them.

When a user first executes the application a login screen is presented, the user can then introduce a username and password. These credentials are used to perform a basic authentication request to Wazuh manager, if successful the manager will respond with a JWT which will then be used to perform other requests to the Manager. If at any point in time the JWT expires, no more actions can be performed by the user and to keep on using the application the user must re-authenticate.

Once authenticated, the user is presented with a screen from which it is possible to perform different actions, namely list and view alerts, list and edit rules or decoders, edit the Wazuh managers configuration, overview agents and export data to Comma Separated Values (CSV) format.

The GUI allows the user to list the alerts that have been generated by the Wazuh manager, which can be filtered or sorted based on different parameters. If the user wants to get more information about any of the items on the list, it is possible to select one by clicking on it, which then presents a screen with extra information about that alert. Since the feature to present graphs within the application has not been implemented, the user can export the most important information about the alerts to CSV format, so that he can create a graphical representation of these alerts in another tool (e.g. Microsoft Excel).

The GUI allows the user to list all the rules or decoders which are stored on the Wazuh manager, an example of such list can be seen in **Figure** 16. The lists containing the rules or decoders can also be sorted and filtered. If the user selects one of the items on the list another view is presented, this view contains the XML representation of the rule or decoder. If the rule or decoder does not belong to the set of rules and decoders pre-created by the Wazuh team, then the user can also edit and update it, which then modifies the corresponding entity on the Wazuh manager.

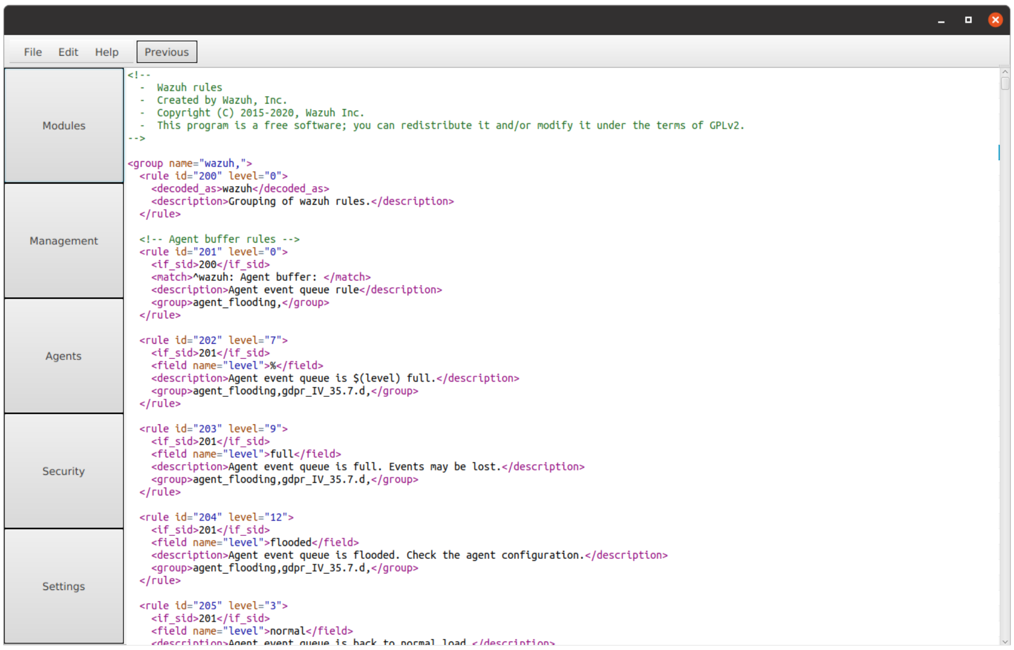


Id	Description	...	Level	File	Path
7200	Arpwatch messages grouped.		0	0115-arpwatch_rules.xml	ruleset/rules
7201	Arpwatch new host detected.		4	0115-arpwatch_rules.xml	ruleset/rules
7202	Arpwatch: "flip flop" message. IP address/MAC relation changing too often.		9	0115-arpwatch_rules.xml	ruleset/rules
7203	Arpwatch: exiting.		3	0115-arpwatch_rules.xml	ruleset/rules
7204	Arpwatch: Changed network interface for ip address.		9	0115-arpwatch_rules.xml	ruleset/rules
7205	Arpwatch: startup/exiting messages.		0	0115-arpwatch_rules.xml	ruleset/rules
7206	Arpwatch: detected bad address len (ignored).		0	0115-arpwatch_rules.xml	ruleset/rules
7207	arpwatch probably run with wrong permissions		1	0115-arpwatch_rules.xml	ruleset/rules
7208	Arpwatch: An IP has reverted to an old ethernet address.		1	0115-arpwatch_rules.xml	ruleset/rules
7209	Arpwatch: Possible arpspoofing attempt.		7	0115-arpwatch_rules.xml	ruleset/rules
7300	Grouping of Symantec AV rules.		0	0120-symantec-av_rules.xml	ruleset/rules
7301	Grouping of Symantec AV rules from eventlog.		0	0120-symantec-av_rules.xml	ruleset/rules
7310	Symantec-AV: Virus detected.		9	0120-symantec-av_rules.xml	ruleset/rules
7320	Symantec-AV: Virus scan updated,started or stopped.		3	0120-symantec-av_rules.xml	ruleset/rules
7400	Grouping of Symantec Web Security rules.		0	0125-symantec-ws_rules.xml	ruleset/rules
7410	Symantec-WS: Login failed accessing the web proxy.		5	0125-symantec-ws_rules.xml	ruleset/rules
7415	Symantec-WS: Login success accessing the web proxy.		3	0125-symantec-ws_rules.xml	ruleset/rules
7420	Symantec-WS: Admin Login success to the web proxy.		3	0125-symantec-ws_rules.xml	ruleset/rules
7500	Grouping of McAfee Windows AV rules.		0	0225-mcafee_av_rules.xml	ruleset/rules
7501	McAfee Windows AV informational event.		2	0225-mcafee_av_rules.xml	ruleset/rules
7502	McAfee Windows AV warning event.		3	0225-mcafee_av_rules.xml	ruleset/rules
7503	McAfee Windows AV error event.		4	0225-mcafee_av_rules.xml	ruleset/rules
7504	McAfee Windows AV - Virus detected and not removed.		12	0225-mcafee_av_rules.xml	ruleset/rules
7505	McAfee Windows AV - Virus detected and properly removed.		7	0225-mcafee_av_rules.xml	ruleset/rules
7506	McAfee Windows AV - Virus detected and file will be deleted.		7	0225-mcafee_av_rules.xml	ruleset/rules
7507	McAfee Windows AV - Scan started or stopped.		3	0225-mcafee_av_rules.xml	ruleset/rules
7508	McAfee Windows AV - Scan completed with no viruses found.		3	0225-mcafee_av_rules.xml	ruleset/rules
7509	McAfee Windows AV - Virus scan cancelled.		5	0225-mcafee_av_rules.xml	ruleset/rules
7510	McAfee Windows AV - Virus scan cancelled due to shutdown.		5	0225-mcafee_av_rules.xml	ruleset/rules
7511	McAfee Windows AV - Virus program or DAT update succeeded.		3	0225-mcafee_av_rules.xml	ruleset/rules
7512	McAfee Windows AV - Virus program or DAT update failed.		7	0225-mcafee_av_rules.xml	ruleset/rules
7513	McAfee Windows AV - Virus program or DAT update cancelled.		7	0225-mcafee_av_rules.xml	ruleset/rules
7514	McAfee Windows AV - EICAR test file detected.		5	0225-mcafee_av_rules.xml	ruleset/rules
7550	Multiple McAfee AV warning events.		10	0225-mcafee_av_rules.xml	ruleset/rules
7600	Grouping of Trend OSCE rules.		0	0130-trend-osce_rules.xml	ruleset/rules
7610	Trend: Virus detected and cleaned/quarantined/removed		5	0130-trend-osce_rules.xml	ruleset/rules
7611	Trend: Virus detected and unable to clean up.		9	0130-trend-osce_rules.xml	ruleset/rules
7612	Trend: Virus scan completed with no errors detected.		3	0130-trend-osce_rules.xml	ruleset/rules

Figure 16: WazuhFX view listing the rules which are stored on the Wazuh manager.

Another operation that can be performed from the GUI is to access a list of the agents which are registered to the Wazuh manager. This list can be sorted by the user. If the user clicks on one of the items of the list, a new view is presented where extra information about an agent is given.

Finally, the user can access the Wazuh manager configuration, which is presented in an XML editor, such as the one seen in **Figure** 17. Within this editor the user can modify the different parameters of the Wazuh managers configuration. When the user saves these modifications, the respective configurations are updated on the Wazuh manager..



```
<!--
  -  Wazuh rules
  -  Created by Wazuh, Inc.
  -  Copyright (C) 2015-2020, Wazuh Inc.
  -  This program is a free software; you can redistribute it and/or modify it under the terms of GPLv2.
-->

<group name="wazuh,">
  <rule id="200" level="0">
    <decoded_as>wazuh</decoded_as>
    <description>Grouping of wazuh rules.</description>
  </rule>

  <!-- Agent buffer rules -->
  <rule id="201" level="0">
    <if_sid>200</if_sid>
    <match>^wazuh: Agent buffer: </match>
    <description>Agent event queue rule</description>
    <group>agent_flooding,</group>
  </rule>

  <rule id="202" level="7">
    <if_sid>201</if_sid>
    <field name="level">%</field>
    <description>Agent event queue is $(level) full.</description>
    <group>agent_flooding,gdpr_IV_35.7.d,</group>
  </rule>

  <rule id="203" level="9">
    <if_sid>201</if_sid>
    <field name="level">full</field>
    <description>Agent event queue is full. Events may be lost.</description>
    <group>agent_flooding,gdpr_IV_35.7.d,</group>
  </rule>

  <rule id="204" level="12">
    <if_sid>201</if_sid>
    <field name="level">flooded</field>
    <description>Agent event queue is flooded. Check the agent configuration.</description>
    <group>agent_flooding,gdpr_IV_35.7.d,</group>
  </rule>

  <rule id="205" level="3">
    <if_sid>201</if_sid>
    <field name="level">normal</field>
    <description>Agent event queue is back to normal load.</description>
```

Figure 17: WazuhFX XML editor view.

5 CONCLUSIONS

The internship at VOID software provided a great opportunity to expand and consolidate the skills and knowledge gained throughout the Master's degree.

An important takeaway from this experience was the experience of working within an organisation which specialises in developing software. Insights into how the development process works, the people that are part of it and the methodology that are required, were both observed and put into practice. The result of this process was the ability to take an idea and transform it into a tangible result.

As a cybersecurity student it is pertinent to highlight the importance of knowing how to carry out research and applying it to accomplish a goal. Many of the challenges that were faced while completing this Master's degree had to do with researching topics to gather enough information and reaching a reasonable conclusion with regard to the topic at hand. This internship was no different: while performing the tasks leading up to the proposed project completion it was necessary to put into practice these research skills by looking into new technologies, concepts and ideas throughout this internship.

More specifically, it was possible to put into practice the skills related to software development and cybersecurity. Throughout the project an agile methodology was followed (namely FDD) with the objective of implementing the functionalities of a security tool one by one. In order to support this methodology, *agile* tools were used to keep track of the task at hand and to retain an overview of the versions of the project that had been delivered. Furthermore it was necessary to gain an understanding of the various processes which are followed within VOID software; and to learn about the systems and network architecture of the organisation, to then be able to set up a security tool in the best way possible which monitors system and network events, fulfilling some of the goals the organisation has set out to accomplish regarding the security of their digital assets.

By using the JavaFX platform alongside other Java libraries it was possible to start and advance the development of a security tool. Although this tool was not finished by the end of the internship and more development is required for it to be a finalised product, the project has started and a solid base was delivered, so the tool can be further worked on if VOID software desires so. Knowledge about JavaFX was gained and used as best as possible to create a new GUI for the Wazuh system. This tool was thought of and developed to ensure that its development can be continued at any point in time, such objective was made possible by using a mature and well known development pattern (MVVM). At the end of the internship it was possible to produce a tool that is able to communicate with the Wazuh API, fetch information that was produced by the Wazuh server and present it to the user.

The Wazuh system was set up within the VOID software organisation, allowing them to monitor several of their systems in case a security incident occurs. Implementing this system helped to learn about the inner workings of a security tool (EDR) and the uses it may have (IR) for an organisation; although not being the main objective of this task, this knowledge was then applied while developing the GUI. To set up Wazuh it was also necessary to understand how core systems are distributed on the different networks of the organisation, providing a general overview of the systems architecture of a medium sized company and the digital assets which are part of it. After having an understanding of these two concepts it was possible to take decisions regarding the main criteria related to the Wazuh system installation.

Finally the communication flows of the different components of the Wazuh system were understood, to then be able to modify its initial architecture. It was possible to migrate the data storage solution (Elasticsearch) to a new one (Solr); it was also possible to replace the current GUI (Kibana) with a new one (WazuhFX); and it was possible to modify the tool (Filebeat) which forwarded alerts to the database, with a new one (post-tool). These modifications were only carried out in a development environment and require to be worked on before performing them on a production system. Although this was the case, the investigation supporting these changes and initial development modifications are an excellent starting point to carry out the final implementation of the new VOID flavoured Wauzuh system.

5.0.1 *Future work*

Although the internship has already been finished, the projects that were developed could still be continued. As such, the following improvements and increments are suggested:

- Carry out tests on the deployed application at VOID to certify that it works as intended;

- Add graphs to represent data within the GUI;
- Fine tune the Wazuh manager so that i does not create unnecessary alerts;
- Implement the new Solr storage solution for a production environment;
- Improve the configuration system of the WazuhFX application;
- Verify the XML structure of rules, decoders and configurations within the front-end application.

BIBLIOGRAPHY

Chin, Stephen (2019). *The Definitive Guide to Modern Java Clients with JavaFX : Cross-Platform Mobile and Cloud Development*. Berkeley, CA: Apress. ISBN: 978-1-4842-4925-3.

Chuvakin, Anton (July 26, 2013). *Named: Endpoint Threat Detection & Response*. Gartner. URL: https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/ (visited on 09/23/2021).

Comodo (Apr. 2021). *What is enpoint security? And why is it crucial today?* Comodo Seecurity. URL: https://enterprise.comodo.com/blog/what-is-endpoint-security/ (visited on 09/20/2021).

ControlsFX (2022). *ControlsFX*. Ed. by Jonathan Giles. URL: https://github.com/controlsfx/controlsfx (visited on 03/14/2022).

Cranor, Lorrie Faith and Norbou Buchler (Nov. 2014). "Better Together: Usability and Security Go Hand in Hand". In: 12.6, pp. 89–93. DOI: 10.1109/msp.2014.109.

David Kim, Michael G. Solomon (Oct. 26, 2016). *Fundamentals of Information Systems Security: Print Bundle*. JONES & BARTLETT PUB INC. 548 pp. ISBN: 128411645X. URL: https://www.ebook.de/de/product/26228820/david_kim_michael_g_solomon_fundamentals_of_information_systems_security_print_bundle.html.

Examples, Java by (2022). *Telescoping Constructor in Java*. URL: http://www.javabyexamples.com/telescoping-constructor-in-java/ (visited on 02/17/2022).

FasterXML (2022). *Jackson Databind*. URL: https://github.com/FasterXML/jackson-databind (visited on 03/08/2022).

Gartner (2021). *Endpoint protection platform*. Gartner. URL: https://www.gartner.com/en/information-technology/glossary/endpoint-protection-platform-epp (visited on 09/23/2021).

Google (2022). *Gson*. URL: https://github.com/google/gson (visited on 03/08/2022).

Grunwald, Gerrit (2022). *TilesFX*. URL: https://github.com/HanSolo/tilesfx (visited on 03/01/2022).

Gunson, Nancie et al. (June 2011). “User perceptions of security and usability of single-factor and two-factor authentication in automated telephone banking”. In: 30.4, pp. 208–220. DOI: 10.1016/j.cose.2010.12.001.

HowToDoInJava (Jan. 3, 2022). *Builder Design Pattern*. URL: https://howtodoinjava.com/design-patterns/creational/builder-pattern-in-java/ (visited on 02/17/2022).

Karl, Patrick (Feb. 21, 2012). *Incident Handler’s Handbook*.

Kong (2022). *Unirest-Java Documentation*. URL: http://kong.github.io/unirest-java/ (visited on 02/17/2022).

Liao, Hung-Jen et al. (2013). “Intrusion detection system: A comprehensive review”. In: *Journal of Network and Computer Applications* 36.1, pp. 16–24. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2012.09.004. URL: https://www.sciencedirect.com/science/article/pii/S1084804512001944.

McFee (Sept. 2020). *What is ednpoint security?* McFee. URL: https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint.html (visited on 09/23/2021).

Microsoft (Apr. 10, 2012). *The MVVM Pattern*. URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10)?redirectedfrom=MSDN.

— (Apr. 15, 2021). *Desktop Guide (WPF .NET)*. Ed. by DCtheGeek adego. URL: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-6.0 (visited on 02/15/2022).

Miller, David (2011). *Security information and event management (SIEM) implementation*. New York: McGraw-Hill. ISBN: 9780071701082.

Möller, Sebastian et al. (June 2011). “Modeling the behavior of users who are confronted with security mechanisms”. In: 30.4, pp. 242–256. DOI: 10.1016/j.cose.2011.01.001.

Oracle (2014). *JavaFX: Getting Started with JavaFX*. URL: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm.

Parisi, Alessadro (2022). *MaterialFX*. URL: https://github.com/palexdev/MaterialFX (visited on 02/16/2022).

Podzins, Oskars and Andrejs Romanovs (Apr. 2019). “Why SIEM is Irreplaceable in a Secure IT Environment?” In: *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. DOI: 10.1109/eStream.2019.8732173.

Rudolph, Carsten and Andreas Fuchs (May 2012). “Redefining Security Engineering”. In: IEEE. DOI: 10.1109/ntms.2012.6208773.

Solr (2022a). *Installing Solr*. URL: https://solr.apache.org/guide/8_8/installing-solr.html (visited on 03/01/2022).

— (2022b). *Schema Designer*. URL: https://solr.apache.org/guide/8_10/schema-designer.html (visited on 03/01/2022).

— (2022c). *SolrJ*. URL: https://solr.apache.org/guide/8_7/using-solrj.html (visited on 03/08/2022).

— (2022d). *Transforming and Indexing Custom JSON*. URL: https://solr.apache.org/guide/8_8/transforming-and-indexing-custom-json.html (visited on 03/01/2022).

Verizon (2021). *2021 Data Breach Investigation Report*. Tech. rep. Verizon.

Wazuh (2022a). *Architecture*. URL: https://documentation.wazuh.com/current/getting-started/architecture.html (visited on 03/08/2022).

— (2022b). *Docker*. URL: https://documentation.wazuh.com/current/docker/index.html (visited on 03/08/2022).

— (2022c). *Installation Guide*. URL: https://documentation.wazuh.com/current/installation-guide/index.html (visited on 03/08/2022).

— (2022d). *Server Installation*. URL: https://documentation.wazuh.com/current/installation-guide/open-distro/index.html (visited on 03/08/2022).

— (2022e). *Virtual Machine (OVA)*. URL: https://documentation.wazuh.com/current/virtual-machine/virtual-machine.html (visited on 03/08/2022).

— (2021). *Wazuh Docs*. URL: https://documentation.wazuh.com/current/index.html (visited on 11/08/2021).

Weir, Catherine S. et al. (Feb. 2009). "User perceptions of security, convenience and usability for ebanking authentication tokens". In: 28.1-2, pp. 47–62. DOI: 10.1016/j.cose.2008.09.008.

APPENDIXES

A

SOLR SCHEMA CONFIGURATION

The schema file for the solr test configuration.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Solr managed schema - automatically generated - DO NOT EDIT -->
<schema name="default-config" version="1.6">
  <uniqueKey>id</uniqueKey>
  <fieldType name="_nest_path_" class="solr.NestPathField"
  ↪ maxCharsForDocValues="-1" omitNorms="true" omitTermFreqAndPositions="true"
  ↪ stored="false" multiValued="false"/>
  <fieldType name="ancestor_path" class="solr.TextField">
    <analyzer type="index">
      <tokenizer class="solr.KeywordTokenizerFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/"/>
    </analyzer>
  </fieldType>
  <fieldType name="binary" class="solr.BinaryField"/>
  <fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>
  <fieldType name="booleans" class="solr.BoolField" sortMissingLast="true"
  ↪ multiValued="true"/>
  <fieldType name="delimited_payloads_float" class="solr.TextField"
  ↪ indexed="true" stored="false">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.DelimitedPayloadTokenFilterFactory" encoder="float"/>
    </analyzer>
  </fieldType>
  <fieldType name="delimited_payloads_int" class="solr.TextField" indexed="true"
  ↪ stored="false">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.DelimitedPayloadTokenFilterFactory" encoder="integer"/>
    </analyzer>
  </fieldType>
  <fieldType name="delimited_payloads_string" class="solr.TextField"
  ↪ indexed="true" stored="false">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.DelimitedPayloadTokenFilterFactory"
      ↪ encoder="identity"/>
    </analyzer>
```

```
</fieldType>
<fieldType name="descendent_path" class="solr.TextField">
  <analyzer type="index">
    <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.KeywordTokenizerFactory"/>
  </analyzer>
</fieldType>
<fieldType name="ignored" class="solr.StrField" indexed="false" stored="false"
↪  multiValued="true"/>
<fieldType name="location" class="solr.LatLonPointSpatialField"
↪  docValues="true"/>
<fieldType name="location_rpt"
↪  class="solr.SpatialRecursivePrefixTreeFieldType" geo="true"
↪  omitNorms="true" omitTermFreqAndPositions="true" maxDistErr="0.001"
↪  termOffsets="false" distErrPct="0.025" distanceUnits="kilometers"
↪  termPositions="false" omitPositions="true"/>
<fieldType name="lowercase" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.KeywordTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="pdate" class="solr.DatePointField" docValues="true"/>
<fieldType name="pdates" class="solr.DatePointField" docValues="true"
↪  multiValued="true"/>
<fieldType name="pdouble" class="solr.DoublePointField" docValues="true"/>
<fieldType name="pdoubles" class="solr.DoublePointField" docValues="true"
↪  multiValued="true"/>
<fieldType name="pfloat" class="solr.FloatPointField" docValues="true"/>
<fieldType name="pfloats" class="solr.FloatPointField" docValues="true"
↪  multiValued="true"/>
<fieldType name="phonetic_en" class="solr.TextField" indexed="true"
↪  stored="false">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.DoubleMetaphoneFilterFactory" inject="false"/>
  </analyzer>
</fieldType>
<fieldType name="pint" class="solr.IntPointField" docValues="true"/>
<fieldType name="pints" class="solr.IntPointField" docValues="true"
↪  multiValued="true"/>
<fieldType name="plong" class="solr.LongPointField" docValues="true"/>
<fieldType name="plongs" class="solr.LongPointField" docValues="true"
↪  multiValued="true"/>
<fieldType name="point" class="solr.PointType" subFieldSuffix="_d"
↪  dimension="2"/>
<fieldType name="random" class="solr.RandomSortField" indexed="true"/>
<fieldType name="rank" class="solr.RankField"/>
<fieldType name="string" class="solr.StrField" sortMissingLast="true"
↪  docValues="true"/>
<fieldType name="strings" class="solr.StrField" sortMissingLast="true"
↪  docValues="true" multiValued="true"/>
```

```
<fieldType name="text_ar" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_ar.txt"
    ↪  ignoreCase="true"/>
    <filter class="solr.ArabicNormalizationFilterFactory"/>
    <filter class="solr.ArabicStemFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="text_bg" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_bg.txt"
    ↪  ignoreCase="true"/>
    <filter class="solr.BulgarianStemFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="text_ca" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.ElisionFilterFactory"
    ↪  articles="lang/contractions_ca.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_ca.txt"
    ↪  ignoreCase="true"/>
    <filter class="solr.SnowballPorterFilterFactory" language="Catalan"/>
  </analyzer>
</fieldType>
<fieldType name="text_cjk" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.CJKWidthFilterFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.CJKBigramFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="text_cz" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_cz.txt"
    ↪  ignoreCase="true"/>
    <filter class="solr.CzechStemFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="text_da" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" format="snowball"
    ↪  words="lang/stopwords_da.txt" ignoreCase="true"/>
    <filter class="solr.SnowballPorterFilterFactory" language="Danish"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_de" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪  words="lang/stopwords_de.txt" ignoreCase="true"/>
      <filter class="solr.GermanNormalizationFilterFactory"/>
      <filter class="solr.GermanLightStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_el" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.GreekLowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_el.txt"
      ↪  ignoreCase="false"/>
      <filter class="solr.GreekStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
    <analyzer type="index">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.EnglishPossessiveFilterFactory"/>
      <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
      <filter class="solr.PorterStemFilterFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.SynonymGraphFilterFactory" expand="true"
      ↪  ignoreCase="true" synonyms="synonyms.txt"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.EnglishPossessiveFilterFactory"/>
      <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
      <filter class="solr.PorterStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_en_splitting" class="solr.TextField"
  ↪  autoGeneratePhraseQueries="true" positionIncrementGap="100">
    <analyzer type="index">
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.WordDelimiterGraphFilterFactory" catenateNumbers="1"
      ↪  generateNumberParts="1" splitOnCaseChange="1" generateWordParts="1"
      ↪  catenateAll="0" catenateWords="1"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
      <filter class="solr.PorterStemFilterFactory"/>
      <filter class="solr.FlattenGraphFilterFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.SynonymGraphFilterFactory" expand="true"
      ↪  ignoreCase="true" synonyms="synonyms.txt"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.WordDelimiterGraphFilterFactory" catenateNumbers="0"
      ↪  generateNumberParts="1" splitOnCaseChange="1" generateWordParts="1"
      ↪  catenateAll="0" catenateWords="0"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
      <filter class="solr.PorterStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_en_splitting_tight" class="solr.TextField"
  ↪  autoGeneratePhraseQueries="true" positionIncrementGap="100">
    <analyzer type="index">
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.SynonymGraphFilterFactory" expand="false"
      ↪  ignoreCase="true" synonyms="synonyms.txt"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.WordDelimiterGraphFilterFactory" catenateNumbers="1"
      ↪  generateNumberParts="0" generateWordParts="0" catenateAll="0"
      ↪  catenateWords="1"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
      <filter class="solr.EnglishMinimalStemFilterFactory"/>
      <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
      <filter class="solr.FlattenGraphFilterFactory"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      <filter class="solr.SynonymGraphFilterFactory" expand="false"
      ↪  ignoreCase="true" synonyms="synonyms.txt"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.WordDelimiterGraphFilterFactory" catenateNumbers="1"
      ↪  generateNumberParts="0" generateWordParts="0" catenateAll="0"
      ↪  catenateWords="1"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
      <filter class="solr.EnglishMinimalStemFilterFactory"/>
      <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_es" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪ words="lang/stopwords_es.txt" ignoreCase="true"/>
      <filter class="solr.SpanishLightStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_et" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_et.txt"
      ↪ ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Estonian"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_eu" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_eu.txt"
      ↪ ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Basque"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_fa" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <charFilter class="solr.PersianCharFilterFactory"/>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ArabicNormalizationFilterFactory"/>
      <filter class="solr.PersianNormalizationFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_fa.txt"
      ↪ ignoreCase="true"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_fi" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪ words="lang/stopwords_fi.txt" ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Finnish"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_fr" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.ElisionFilterFactory"
      ↪ articles="lang/contractions_fr.txt" ignoreCase="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪ words="lang/stopwords_fr.txt" ignoreCase="true"/>
      <filter class="solr.FrenchLightStemFilterFactory"/>
    </analyzer>
  </fieldType>
```

```
<fieldType name="text_ga" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.ElisionFilterFactory"
    ↪ articles="lang/contractions_ga.txt" ignoreCase="true"/>
    <filter class="solr.StopFilterFactory" words="lang/hyphenations_ga.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.IrishLowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" words="lang/stopwords_ga.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.SnowballPorterFilterFactory" language="Irish"/>
  </analyzer>
</fieldType>
<fieldType name="text_gen_sort" class="solr.SortableTextField"
↪ positionIncrementGap="100" multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.SynonymGraphFilterFactory" expand="true"
    ↪ ignoreCase="true" synonyms="synonyms.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="text_general" class="solr.TextField"
↪ positionIncrementGap="100" multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.SynonymGraphFilterFactory" expand="true"
    ↪ ignoreCase="true" synonyms="synonyms.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
<fieldType name="text_general_rev" class="solr.TextField"
↪ positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt"
    ↪ ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
```

```
      <filter class="solr.ReversedWildcardFilterFactory" maxPosQuestion="2"
      ↪  maxFractionAsterisk="0.33" maxPosAsterisk="3" withOriginal="true"/>
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.SynonymGraphFilterFactory" expand="true"
      ↪  ignoreCase="true" synonyms="synonyms.txt"/>
      <filter class="solr.StopFilterFactory" words="stopwords.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_gl" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_gl.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.GalicianStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_hi" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.IndicNormalizationFilterFactory"/>
      <filter class="solr.HindiNormalizationFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_hi.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.HindiStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_hu" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪  words="lang/stopwords_hu.txt" ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Hungarian"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_hy" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_hy.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Armenian"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_id" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
```

```
        <filter class="solr.StopFilterFactory" words="lang/stopwords_id.txt"
        ↪  ignoreCase="true"/>
        <filter class="solr.IndonesianStemFilterFactory" stemDerivational="true"/>
      </analyzer>
    </fieldType>
    <fieldType name="text_it" class="solr.TextField" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.ElisionFilterFactory"
        ↪  articles="lang/contractions_it.txt" ignoreCase="true"/>
        <filter class="solr.LowerCaseFilterFactory"/>
        <filter class="solr.StopFilterFactory" format="snowball"
        ↪  words="lang/stopwords_it.txt" ignoreCase="true"/>
        <filter class="solr.ItalianLightStemFilterFactory"/>
      </analyzer>
    </fieldType>
    <fieldType name="text_ja" class="solr.TextField"
    ↪  autoGeneratePhraseQueries="false" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.JapaneseTokenizerFactory" mode="search"/>
        <filter class="solr.JapaneseBaseFormFilterFactory"/>
        <filter class="solr.JapanesePartOfSpeechStopFilterFactory"
        ↪  tags="lang/stoptags_ja.txt"/>
        <filter class="solr.CJKWidthFilterFactory"/>
        <filter class="solr.StopFilterFactory" words="lang/stopwords_ja.txt"
        ↪  ignoreCase="true"/>
        <filter class="solr.JapaneseKatakanaStemFilterFactory" minimumLength="4"/>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
    </fieldType>
    <fieldType name="text_ko" class="solr.TextField" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.KoreanTokenizerFactory"
        ↪  outputUnknownUnigrams="false" decompoundMode="discard"/>
        <filter class="solr.KoreanPartOfSpeechStopFilterFactory"/>
        <filter class="solr.KoreanReadingFormFilterFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
    </fieldType>
    <fieldType name="text_lv" class="solr.TextField" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
        <filter class="solr.StopFilterFactory" words="lang/stopwords_lv.txt"
        ↪  ignoreCase="true"/>
        <filter class="solr.LatvianStemFilterFactory"/>
      </analyzer>
    </fieldType>
    <fieldType name="text_nl" class="solr.TextField" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
        <filter class="solr.StopFilterFactory" format="snowball"
        ↪  words="lang/stopwords_nl.txt" ignoreCase="true"/>
      <filter class="solr.StemmerOverrideFilterFactory"
      ↪  dictionary="lang/stemdict_nl.txt" ignoreCase="false"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Dutch"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_no" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪  words="lang/stopwords_no.txt" ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Norwegian"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_pt" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪  words="lang/stopwords_pt.txt" ignoreCase="true"/>
      <filter class="solr.PortugueseLightStemFilterFactory"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_ro" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_ro.txt"
      ↪  ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Romanian"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_ru" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪  words="lang/stopwords_ru.txt" ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Russian"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_sv" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" format="snowball"
      ↪  words="lang/stopwords_sv.txt" ignoreCase="true"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Swedish"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_th" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.ThaiTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
```

```xml
      <filter class="solr.StopFilterFactory" words="lang/stopwords_th.txt"
      ↪  ignoreCase="true"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_tr" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.TurkishLowerCaseFilterFactory"/>
      <filter class="solr.StopFilterFactory" words="lang/stopwords_tr.txt"
      ↪  ignoreCase="false"/>
      <filter class="solr.SnowballPorterFilterFactory" language="Turkish"/>
    </analyzer>
  </fieldType>
  <fieldType name="text_ws" class="solr.TextField" positionIncrementGap="100">
    <analyzer>
      <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    </analyzer>
  </fieldType>
  <field name="_nest_path_" type="_nest_path_"/>
  <field name="_root_" type="string" docValues="false" indexed="true"
  ↪  stored="false"/>
  <field name="_text_" type="text_general" multiValued="true" indexed="true"
  ↪  stored="false"/>
  <field name="_version_" type="plong" indexed="false" stored="false"/>
  <field name="agent" type="text_general"/>
  <field name="agent.id" type="plongs"/>
  <field name="agent.ip" type="text_general"/>
  <field name="agent.name" type="text_general"/>
  <field name="data" type="text_general"/>
  <field name="data.dstuser" type="text_general"/>
  <field name="data.extra_data" type="text_general"/>
  <field name="data.file" type="text_general"/>
  <field name="data.sca.check.command" type="text_general"/>
  <field name="data.sca.check.compliance.cis" type="pdoubles"/>
  <field name="data.sca.check.compliance.cis_csc" type="text_general"/>
  <field name="data.sca.check.compliance.gdpr_IV" type="text_general"/>
  <field name="data.sca.check.compliance.gpg_13" type="pdoubles"/>
  <field name="data.sca.check.compliance.hipaa" type="text_general"/>
  <field name="data.sca.check.compliance.nist_800_53" type="text_general"/>
  <field name="data.sca.check.compliance.pci_dss" type="pdoubles"/>
  <field name="data.sca.check.compliance.tsc" type="text_general"/>
  <field name="data.sca.check.description" type="text_general"/>
  <field name="data.sca.check.id" type="plongs"/>
  <field name="data.sca.check.previous_result" type="text_general"/>
  <field name="data.sca.check.rationale" type="text_general"/>
  <field name="data.sca.check.reason" type="text_general"/>
  <field name="data.sca.check.remediation" type="text_general"/>
  <field name="data.sca.check.result" type="text_general"/>
  <field name="data.sca.check.status" type="text_general"/>
  <field name="data.sca.check.title" type="text_general"/>
  <field name="data.sca.description" type="text_general"/>
  <field name="data.sca.failed" type="plongs"/>
  <field name="data.sca.file" type="text_general"/>
  <field name="data.sca.invalid" type="plongs"/>
```

```xml
<field name="data.sca.passed" type="plongs"/>
<field name="data.sca.policy" type="text_general"/>
<field name="data.sca.policy_id" type="text_general"/>
<field name="data.sca.scan_id" type="plongs"/>
<field name="data.sca.score" type="plongs"/>
<field name="data.sca.total_checks" type="plongs"/>
<field name="data.sca.type" type="text_general"/>
<field name="data.srcip" type="text_general"/>
<field name="data.status" type="text_general"/>
<field name="data.title" type="text_general"/>
<field name="data.uid" type="plongs"/>
<field name="decoder" type="text_general"/>
<field name="decoder.name" type="text_general"/>
<field name="decoder.parent" type="text_general"/>
<field name="full_log" type="text_general"/>
<field name="id" type="string" multiValued="false" indexed="true"
↪ required="true" stored="true"/>
<field name="location" type="text_general"/>
<field name="manager" type="text_general"/>
<field name="manager.name" type="text_general"/>
<field name="predecoder.hostname" type="text_general"/>
<field name="predecoder.program_name" type="text_general"/>
<field name="predecoder.timestamp" type="text_general"/>
<field name="previous_log" type="text_general"/>
<field name="previous_output" type="text_general"/>
<field name="rule" type="text_general"/>
<field name="rule.cis" type="pdoubles"/>
<field name="rule.cis_csc" type="pdoubles"/>
<field name="rule.description" type="text_general"/>
<field name="rule.firedtimes" type="plongs"/>
<field name="rule.gdpr" type="text_general"/>
<field name="rule.gdpr_IV" type="text_general"/>
<field name="rule.gpg13" type="pdoubles"/>
<field name="rule.gpg_13" type="pdoubles"/>
<field name="rule.groups" type="text_general"/>
<field name="rule.hipaa" type="text_general"/>
<field name="rule.id" type="plongs"/>
<field name="rule.level" type="plongs"/>
<field name="rule.mail" type="booleans"/>
<field name="rule.mitre.id" type="text_general"/>
<field name="rule.mitre.tactic" type="text_general"/>
<field name="rule.mitre.technique" type="text_general"/>
<field name="rule.nist_800_53" type="text_general"/>
<field name="rule.pci_dss" type="text_general"/>
<field name="rule.tsc" type="text_general"/>
<field name="timestamp" type="text_general"/>
<dynamicField name="*_txt_en_split_tight" type="text_en_splitting_tight"
↪ indexed="true" stored="true"/>
<dynamicField name="*_descendent_path" type="descendent_path" indexed="true"
↪ stored="true"/>
<dynamicField name="*_ancestor_path" type="ancestor_path" indexed="true"
↪ stored="true"/>
<dynamicField name="*_txt_en_split" type="text_en_splitting" indexed="true"
↪ stored="true"/>
```

```
<dynamicField name="*_txt_sort" type="text_gen_sort" indexed="true"
↪  stored="true"/>
<dynamicField name="ignored_*" type="ignored"/>
<dynamicField name="*_txt_rev" type="text_general_rev" indexed="true"
↪  stored="true"/>
<dynamicField name="*_phon_en" type="phonetic_en" indexed="true"
↪  stored="true"/>
<dynamicField name="*_s_lower" type="lowercase" indexed="true" stored="true"/>
<dynamicField name="*_txt_cjk" type="text_cjk" indexed="true" stored="true"/>
<dynamicField name="random_*" type="random"/>
<dynamicField name="*_t_sort" type="text_gen_sort" multiValued="false"
↪  indexed="true" stored="true"/>
<dynamicField name="*_txt_en" type="text_en" indexed="true" stored="true"/>
<dynamicField name="*_txt_ar" type="text_ar" indexed="true" stored="true"/>
<dynamicField name="*_txt_bg" type="text_bg" indexed="true" stored="true"/>
<dynamicField name="*_txt_ca" type="text_ca" indexed="true" stored="true"/>
<dynamicField name="*_txt_cz" type="text_cz" indexed="true" stored="true"/>
<dynamicField name="*_txt_da" type="text_da" indexed="true" stored="true"/>
<dynamicField name="*_txt_de" type="text_de" indexed="true" stored="true"/>
<dynamicField name="*_txt_el" type="text_el" indexed="true" stored="true"/>
<dynamicField name="*_txt_es" type="text_es" indexed="true" stored="true"/>
<dynamicField name="*_txt_et" type="text_et" indexed="true" stored="true"/>
<dynamicField name="*_txt_eu" type="text_eu" indexed="true" stored="true"/>
<dynamicField name="*_txt_fa" type="text_fa" indexed="true" stored="true"/>
<dynamicField name="*_txt_fi" type="text_fi" indexed="true" stored="true"/>
<dynamicField name="*_txt_fr" type="text_fr" indexed="true" stored="true"/>
<dynamicField name="*_txt_ga" type="text_ga" indexed="true" stored="true"/>
<dynamicField name="*_txt_gl" type="text_gl" indexed="true" stored="true"/>
<dynamicField name="*_txt_hi" type="text_hi" indexed="true" stored="true"/>
<dynamicField name="*_txt_hu" type="text_hu" indexed="true" stored="true"/>
<dynamicField name="*_txt_hy" type="text_hy" indexed="true" stored="true"/>
<dynamicField name="*_txt_id" type="text_id" indexed="true" stored="true"/>
<dynamicField name="*_txt_it" type="text_it" indexed="true" stored="true"/>
<dynamicField name="*_txt_ja" type="text_ja" indexed="true" stored="true"/>
<dynamicField name="*_txt_ko" type="text_ko" indexed="true" stored="true"/>
<dynamicField name="*_txt_lv" type="text_lv" indexed="true" stored="true"/>
<dynamicField name="*_txt_nl" type="text_nl" indexed="true" stored="true"/>
<dynamicField name="*_txt_no" type="text_no" indexed="true" stored="true"/>
<dynamicField name="*_txt_pt" type="text_pt" indexed="true" stored="true"/>
<dynamicField name="*_txt_ro" type="text_ro" indexed="true" stored="true"/>
<dynamicField name="*_txt_ru" type="text_ru" indexed="true" stored="true"/>
<dynamicField name="*_txt_sv" type="text_sv" indexed="true" stored="true"/>
<dynamicField name="*_txt_th" type="text_th" indexed="true" stored="true"/>
<dynamicField name="*_txt_tr" type="text_tr" indexed="true" stored="true"/>
<dynamicField name="*_point" type="point" indexed="true" stored="true"/>
<dynamicField name="*_srpt" type="location_rpt" indexed="true" stored="true"/>
<dynamicField name="attr_*" type="text_general" multiValued="true"
↪  indexed="true" stored="true"/>
<dynamicField name="*_txt" type="text_general" indexed="true" stored="true"/>
<dynamicField name="*_str" type="strings" docValues="true" indexed="false"
↪  stored="false" useDocValuesAsStored="false"/>
<dynamicField name="*_dts" type="pdate" multiValued="true" indexed="true"
↪  stored="true"/>
```

```xml
<dynamicField name="*_dpf" type="delimited_payloads_float" indexed="true"
↪  stored="true"/>
<dynamicField name="*_dpi" type="delimited_payloads_int" indexed="true"
↪  stored="true"/>
<dynamicField name="*_dps" type="delimited_payloads_string" indexed="true"
↪  stored="true"/>
<dynamicField name="*_is" type="pints" indexed="true" stored="true"/>
<dynamicField name="*_ss" type="strings" indexed="true" stored="true"/>
<dynamicField name="*_ls" type="plongs" indexed="true" stored="true"/>
<dynamicField name="*_bs" type="booleans" indexed="true" stored="true"/>
<dynamicField name="*_fs" type="pfloats" indexed="true" stored="true"/>
<dynamicField name="*_ds" type="pdoubles" indexed="true" stored="true"/>
<dynamicField name="*_dt" type="pdate" indexed="true" stored="true"/>
<dynamicField name="*_ws" type="text_ws" indexed="true" stored="true"/>
<dynamicField name="*_i" type="pint" indexed="true" stored="true"/>
<dynamicField name="*_s" type="string" indexed="true" stored="true"/>
<dynamicField name="*_l" type="plong" indexed="true" stored="true"/>
<dynamicField name="*_t" type="text_general" multiValued="false"
↪  indexed="true" stored="true"/>
<dynamicField name="*_b" type="boolean" indexed="true" stored="true"/>
<dynamicField name="*_f" type="pfloat" indexed="true" stored="true"/>
<dynamicField name="*_d" type="pdouble" indexed="true" stored="true"/>
<dynamicField name="*_p" type="location" indexed="true" stored="true"/>
<copyField source="agent" dest="agent_str" maxChars="256"/>
<copyField source="manager" dest="manager_str" maxChars="256"/>
<copyField source="data" dest="data_str" maxChars="256"/>
<copyField source="rule" dest="rule_str" maxChars="256"/>
<copyField source="location" dest="location_str" maxChars="256"/>
<copyField source="decoder" dest="decoder_str" maxChars="256"/>
<copyField source="timestamp" dest="timestamp_str" maxChars="256"/>
<copyField source="full_log" dest="full_log_str" maxChars="256"/>
<copyField source="agent.ip" dest="agent.ip_str" maxChars="256"/>
<copyField source="rule.tsc" dest="rule.tsc_str" maxChars="256"/>
<copyField source="rule.gdpr" dest="rule.gdpr_str" maxChars="256"/>
<copyField source="rule.hipaa" dest="rule.hipaa_str" maxChars="256"/>
<copyField source="decoder.parent" dest="decoder.parent_str" maxChars="256"/>
<copyField source="rule.nist_800_53" dest="rule.nist_800_53_str"
↪  maxChars="256"/>
<copyField source="rule.description" dest="rule.description_str"
↪  maxChars="256"/>
<copyField source="data.extra_data" dest="data.extra_data_str" maxChars="256"/>
<copyField source="decoder.name" dest="decoder.name_str" maxChars="256"/>
<copyField source="manager.name" dest="manager.name_str" maxChars="256"/>
<copyField source="rule.pci_dss" dest="rule.pci_dss_str" maxChars="256"/>
<copyField source="agent.name" dest="agent.name_str" maxChars="256"/>
<copyField source="rule.groups" dest="rule.groups_str" maxChars="256"/>
<copyField source="previous_log" dest="previous_log_str" maxChars="256"/>
<copyField source="previous_output" dest="previous_output_str" maxChars="256"/>
<copyField source="rule.mitre.id" dest="rule.mitre.id_str" maxChars="256"/>
<copyField source="rule.mitre.technique" dest="rule.mitre.technique_str"
↪  maxChars="256"/>
<copyField source="data.dstuser" dest="data.dstuser_str" maxChars="256"/>
<copyField source="predecoder.program_name" dest="predecoder.program_name_str"
↪  maxChars="256"/>
```

```
  <copyField source="predecoder.timestamp" dest="predecoder.timestamp_str"
   ↪  maxChars="256"/>
  <copyField source="rule.mitre.tactic" dest="rule.mitre.tactic_str"
   ↪  maxChars="256"/>
  <copyField source="predecoder.hostname" dest="predecoder.hostname_str"
   ↪  maxChars="256"/>
  <copyField source="data.status" dest="data.status_str" maxChars="256"/>
  <copyField source="data.title" dest="data.title_str" maxChars="256"/>
  <copyField source="data.file" dest="data.file_str" maxChars="256"/>
  <copyField source="data.srcip" dest="data.srcip_str" maxChars="256"/>
  <copyField source="rule.gdpr_IV" dest="rule.gdpr_IV_str" maxChars="256"/>
  <copyField source="data.sca.check.previous_result"
   ↪  dest="data.sca.check.previous_result_str" maxChars="256"/>
  <copyField source="data.sca.type" dest="data.sca.type_str" maxChars="256"/>
  <copyField source="data.sca.check.remediation"
   ↪  dest="data.sca.check.remediation_str" maxChars="256"/>
  <copyField source="data.sca.policy" dest="data.sca.policy_str" maxChars="256"/>
  <copyField source="data.sca.check.description"
   ↪  dest="data.sca.check.description_str" maxChars="256"/>
  <copyField source="data.sca.check.compliance.cis_csc"
   ↪  dest="data.sca.check.compliance.cis_csc_str" maxChars="256"/>
  <copyField source="data.sca.check.command" dest="data.sca.check.command_str"
   ↪  maxChars="256"/>
  <copyField source="data.sca.check.title" dest="data.sca.check.title_str"
   ↪  maxChars="256"/>
  <copyField source="data.sca.check.compliance.gdpr_IV"
   ↪  dest="data.sca.check.compliance.gdpr_IV_str" maxChars="256"/>
  <copyField source="data.sca.check.compliance.nist_800_53"
   ↪  dest="data.sca.check.compliance.nist_800_53_str" maxChars="256"/>
  <copyField source="data.sca.check.compliance.hipaa"
   ↪  dest="data.sca.check.compliance.hipaa_str" maxChars="256"/>
  <copyField source="data.sca.check.compliance.tsc"
   ↪  dest="data.sca.check.compliance.tsc_str" maxChars="256"/>
  <copyField source="data.sca.check.result" dest="data.sca.check.result_str"
   ↪  maxChars="256"/>
  <copyField source="data.sca.check.rationale"
   ↪  dest="data.sca.check.rationale_str" maxChars="256"/>
  <copyField source="data.sca.file" dest="data.sca.file_str" maxChars="256"/>
  <copyField source="data.sca.policy_id" dest="data.sca.policy_id_str"
   ↪  maxChars="256"/>
  <copyField source="data.sca.description" dest="data.sca.description_str"
   ↪  maxChars="256"/>
  <copyField source="data.sca.check.status" dest="data.sca.check.status_str"
   ↪  maxChars="256"/>
  <copyField source="data.sca.check.reason" dest="data.sca.check.reason_str"
   ↪  maxChars="256"/>
</schema>
```

B WAZUH SERVER INSTALLATION

The following steps must be followed to perform a step by step installation of the Wazuh system.

B.1 INSTALLATION PROCESS FROM THE OFFICIAL REPOSITORIES

The first step to install the Wazuh server, is to add the official Wazuh repository to the system where it will be installed.

Before starting we must install a series of necessary packages. To run all of the commands in the installation process we must be authenticated as the system administrator (root user):

```
$ sudo su -
$ apt update
$ apt install curl apt-transport-https unzip wget libcap2-bin
↪  software-properties-common lsb-release gnupg
```

B.1.0.1 *Adding the Wazuh repository*

Before being able to pull the packages from the Wazuh repositories we must add the Gnu Privacy Guard (GPG) key:

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
```

Now we can add the repository and update the packages information:

```
$ echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
↪  /etc/apt/sources.list.d/wazuh.list
$ apt update
```

B.1.0.2 *Installing the Wazuh manager*

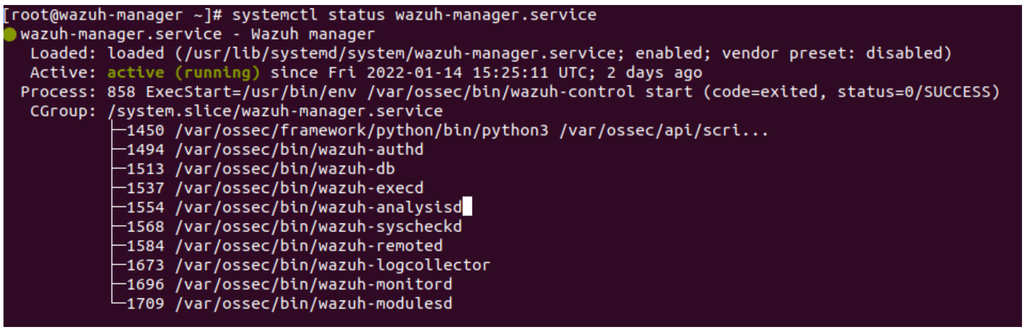
Subsequently we can install the Wazuh manager:

```
$ apt install wazuh-manager
```

After the installation process is completed, we must enable and start the Wazuh manager service:

```
$ systemctl daemon-reload
$ systemctl enable wazuh-manager
$ systemctl start wazuh-manager
```

Finally we check if the Wazuh manager is running:



```
[root@wazuh-manager ~]# systemctl status wazuh-manager.service
● wazuh-manager.service - Wazuh manager
   Loaded: loaded (/usr/lib/systemd/system/wazuh-manager.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-01-14 15:25:11 UTC; 2 days ago
  Process: 858 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/wazuh-manager.service
           ├─1450 /var/ossec/framework/python/bin/python3 /var/ossec/api/scri...
           ├─1494 /var/ossec/bin/wazuh-authd
           ├─1513 /var/ossec/bin/wazuh-db
           ├─1537 /var/ossec/bin/wazuh-execd
           ├─1554 /var/ossec/bin/wazuh-analysisd
           ├─1568 /var/ossec/bin/wazuh-syscheckd
           ├─1584 /var/ossec/bin/wazuh-remoted
           ├─1673 /var/ossec/bin/wazuh-logcollector
           ├─1696 /var/ossec/bin/wazuh-monitord
           └─1709 /var/ossec/bin/wazuh-modulesd
```

C WAZUH AGENT INSTALLATION

The following steps must be followed to install the Wazuh agent.

C.0.1 *Adding the Wazuh repository*

Before being able to pull the packages from the Wazuh repositories we must run commands as the root user, add the Gnu Privacy Guard (GPG) key and add the source repository :

```
$ sudo -su -
$ curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
$ echo "deb https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
↪  /etc/apt/sources.list.d/wazuh.list
$ apt-get update
```

C.0.2 *Deploy the Wazuh agent*

Set the WAZUH_MANAGER variable with the manager IP address and install the Wazuh agent from the Wazuh repository:

```
$ WAZUH_MANAGER="IP_MANAGER" apt-get install wazuh-agent
```

Enable and restart the agent service:

```
$ systemctl daemon-reload
$ systemctl enable wazuh-agent
$ systemctl start wazuh-agent
```

C.0.3 *Disable automatic updates*

Since the Wazuh agents version must be equal or lower than the one of the Wazuh manager, the documentation recommends to disable Wazuh automatic updates:

```
    $ echo "wazuh-agent hold" | dpkg --set-selections
```

DECLARAÇÃO

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"Application Security Testing "*, é original e foi realizado por Tomás Avila Welling (2190381) sob orientação de Professor PhD Marco António de Oliveira Monteiro (marco.monteiro@ipleiria.pt).

Leiria, March of 2022

Tomás Avila Welling

escola superior de tecnologia e gestão
instituto politécnico de leiria

PARECER SOBRE TRABALHO DE ESTÁGIO SUBMETIDO A PROVAS PÚBLICAS SUBSCRITO PELO ORIENTADOR

ORIENTADOR

NOME Marco António de Oliveira Monteiro

CATEGORIA Professor Adjunto **DEPARTAMENTO** Engª Informática

INSTITUIÇÃO ESTG – Instituto Politécnico de Leiria

COORIENTADOR ***(SE APLICÁVEL)***

NOME

CATEGORIA **DEPARTAMENTO**

INSTITUIÇÃO

NA QUALIDADE DE ORIENTADOR(ES) DO ESTUDANTE Tomás Avila Welling

COM O NÚMERO 2190381 DO CURSO DE Mestrado em Cibersegurança e Informática Forense

COM O TEMA Application Security Testing

Declara que o trabalho acima referenciado, por mim revisto, foi concluído com sucesso, respeita as regras aprovadas e reúne a qualidade científica e as condições necessárias para que seja submetido a prova pública de defesa.

DATA 28/03/2022

ORIENTADOR

Assinado por: **Marco António de Oliveira Monteiro**
Num. de Identificação: 09801988
Data: 2022.03.28 19:55:03 +0100

CHAVE MÓVEL