Abstract

This report is about the description of the development process of the project “Automated Incident Response for Cyber Anomalies (AIRCA)”. It is a system that automatically detects, analyses, and responds to security incidents and is fully containerised which makes it lightweight, easy to deploy and ready to use in any environment. This report consists of 9 chapters: Introduction, Background and Literature Review, Development, Testing and Analysis, Conclusion and Future Works, Project Risk Threats and Contingency Plans, Legal Social and Ethical Issues, References and Bibliography, and Appendix.

Keywords: Cyber Anomalies, Cyber Threat Actors, Security Events and Incidents, Security Orchestration, Security Automation, Security Threat Detection and Response, Threat Intelligence, Vulnerability Identification, Correlation Analysis, Containerization.

Table of Contents

Table of Figures

Table of Tables

Chapter I : Introduction

1.1. Topic Introduction

In today's digital landscape where everything relates to each other, people face an ever-increasing volume and complexity of cyber threats. The rapid evolution of cyberattacks demands a proactive and efficient approach to incident response to safeguard sensitive data, protect critical systems, and ensure the continuity of business operations. A system with Automated Incident Response enriched with the various capabilities of **Security Orchestration, Automation, and Response (SOAR)** (IBM, 2015), emerges as a crucial solution to meet this issue.

Figure 1 Elements of SOAR (Palo Alto Networks, 2020)

Traditional manual incident response processes, while effective to some extent, are often overwhelmed by the large scale and sophistication of modern cyber threats. This limitation necessitates a paradigm shift in incident response, where automation and orchestration play a very important role because cybersecurity incidents can have devastating consequences, from financial losses to reputational damage and various legal liabilities.

1.2. Problem Scenario

Cybercrime has become more prevalent as technology is advancing. Following the recent changes prior to the pandemic, the usage of technology and the internet has skyrocketed. At the same time, new types of cyber threats and anomalies emerge every day, increasing the damage. These threats include **Malware Attacks, Phishing Campaigns, Data Breaches, Advanced Persistent Threats (APTs)** and many more. The challenge lies not only in the frequency and sophistication of these attacks but also in the complexity of managing and responding to them effectively.

The escalating volume of daily emerging cyber threats, particularly those attributed to malware, poses a pervasive cybersecurity challenge worldwide. This statement is sharply illustrated through the comprehensive data statistics presented in *Figure 2* and *Figure 3* of SonicWall's threat report titled "**2024 SonicWall Cyber Threat Report**".

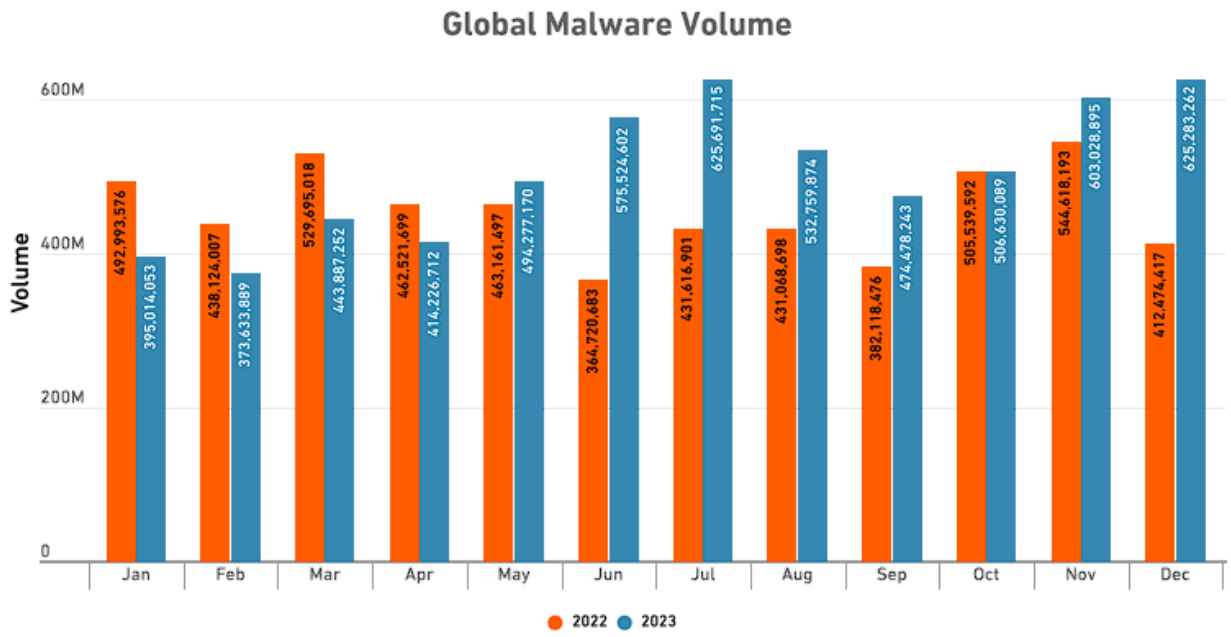


Figure 2 Global Malware Volume (SonicWall, Inc, 2024).

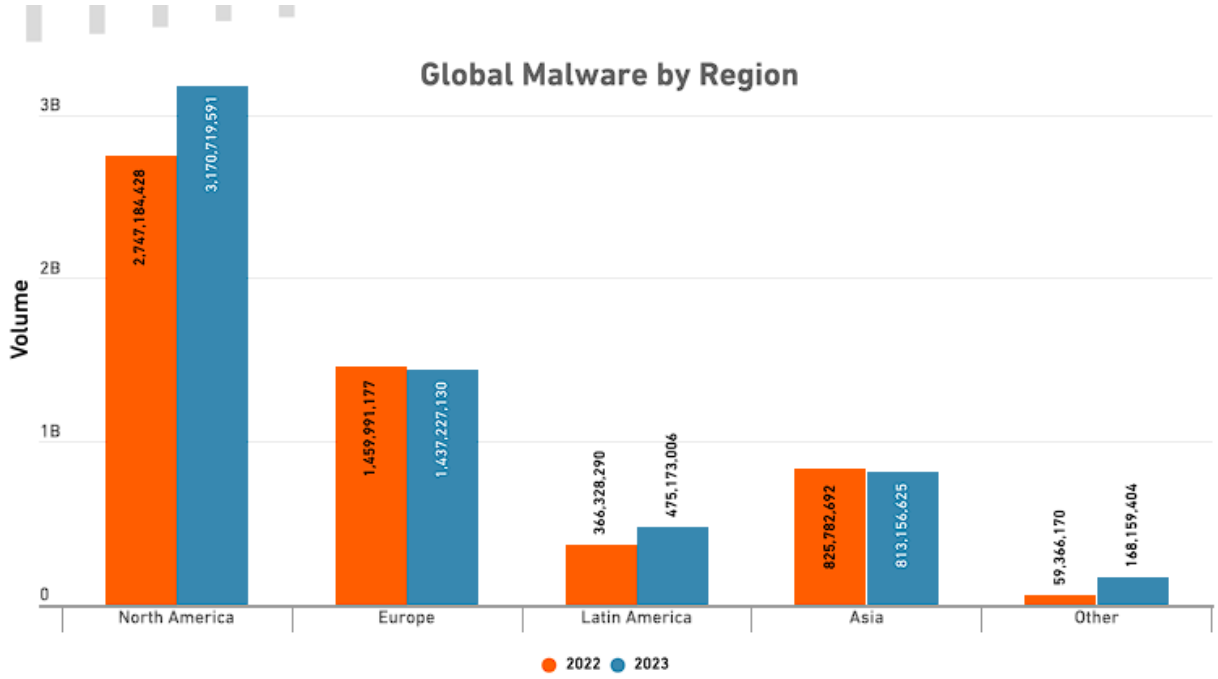


Figure 3 Global Malware by Region (SonicWall, Inc, 2024) .

According to SonicWall, “The 2024 SonicWall Cyber Threat Report gathered intelligence from real-world data collected by SonicWall Capture Labs. This data is securely obtained from devices worldwide, including over 1.1 million security sensors in 215 countries and territories. It encompasses various threat-related information from SonicWall security systems, malware/IP reputation data from firewalls and email security devices, shared intelligence from the cybersecurity community, and more.”

1.3. Project as a Solution

In response to the increasingly complex and relentless cybersecurity anomalies on organizations that reside in the systems totally being undetected which can be stated after looking through the data statistics in *Figure 2* and *Figure 3*, the development and implementation of an **Automated Incident Response for Cyber Anomalies (AIRCA)** system is proposed. This solution offers a comprehensive approach to detect different cyber anomalies and actively respond to them.

AIRCA will help in incident response by building a platform that will detect cyber anomalies on any devices in a network and automates the response for the detected threat. This automation not only alleviates the burden on security teams, but will accelerate response times, and can be setup in any environment. Moreover, the system will integrate with various security tools and platforms, including **Intrusion Detection System (IDS), Security Information and Event Management (SIEM)**, **Threat Intelligence Platform (TIP)** and many more, as per the project requires (SANS, 2023). This approach equips any environment with a better defence against cyber anomalies, strengthening cybersecurity posture and mitigating risks associated with data breaches and various losses.

1.4. Aim and Objectives

1.4.1. Aim

The main aim of this project is to develop a system with the proactive capability to swiftly respond to any detected cyber anomalies during comprehensive monitoring, ensuring immediate and effective incident resolution.

1.4.2. Objectives

The objective to achieve the aim of this project are as follows:

i. Designing a virtual system and network architecture to develop AIRCA.
ii. Implementing existing security infrastructure and integrate them with each other.
iii. Developing automation modules for the security infrastructure to actively monitor, detect and respond to any threat indicators.
iv. Integrating threat intelligence feeds for real-time data sharing and comprehensive threat analysis.
v. Logging each devices network and system activities for further investigations.

1.5. Report Structure

This section provides structure of the whole report.

S.N.	Title	Contents
1.	Chapter I : Introduction	• Topic Introduction • Problem Scenario • Project as a Solution • Aim and Objectives • Report Structure
2.	Chapter II : Background and Literature Review	• Cyber Threat and Anomaly • About End Users • Understanding the Project - Project Elaboration • Similar Projects Review and Comparison Analysis • Resource Requirement - Hardware Requirement - Software Requirement
3.	Chapter III : Development	• Project Methodology • Phases of DSDM Methodology in accordance with project - Phase 1: Pre-Project Phase - Phase 2: Project - Life Cycle Phase - Phase 3: Post-Project Phase • Justification for selecting DSDM Methodology • Survey Analysis • Flowchart of the System • Development of the System
4.	Chapter IV: Testing & Analysis	• Unit Testing • System Testing • Security Testing • Critical Analysis
5.	Chapter V: Project Risks, Threats and Contingency Planning	• Project Risks and Threats • Contingency Planning
6.	Chapter VI: Conclusion	• Project Review/Summary • Advantages • Limitations • Future Works

7.	Chapter VII: Legal, Ethical, and Social Issues	• Legal Issus • Ethical Issues • Social Issues
8.	Chapter VII: Appendix	• Appendix A: Definitions • Appendix B: Pre-Survey • Appendix C: Post-Survey • Appendix D: System Development Phase Evidence • Appendix E: Collection of Charts • Appendix F: Progress Review Table

Table 1 Structure of Report

Chapter II : Background and Literature Review

2.1. Cyber Threat and Anomaly

A cyber threat or anomaly refers to any potential and unusual event or action in a network that can compromise the confidentiality, integrity, or availability of digital information and resources. These threats are often perpetrated by malicious actors who exploit vulnerabilities in computer systems, networks, or software applications. Cyber threat and anomaly can manifest in various forms, including but not limited to:

- **Malware:** Malicious software designed to disrupt, damage, or gain unauthorized access to computer systems. Examples of malware include viruses, worms, Trojans, ransomware, and spyware.
- **Phishing:** Deceptive techniques used to trick individuals into divulging sensitive information such as passwords, usernames, or financial data by posing as a trustworthy entity.
- **Denial of Service (DoS) Attacks:** Deliberate attempts to make a machine or network resource unavailable to users by overwhelming it with a flood of traffic or resource requests.
- **Data Breaches:** Unauthorized access to confidential or sensitive data, often resulting in its theft, modification, or destruction.
- **Insider Threats:** Malicious activities perpetrated by individuals within an organization, such as employees or contractors, with privileged access.

2.2. About End Users

The project is focused on end users who stand at the forefront of cybersecurity challenges in today's digital landscape, facing an array of threats such as malware attacks, phishing campaigns, and data breaches. AIRCA emerges as a solution for defence, offering automatic detection, analysis, and response to these security incidents. Its lightweight, containerized architecture ensures easy deployment in any environment, empowering organizations of all sizes to strengthen their cybersecurity defences with minimal complexity.

Integration with open-source unified SIEM and XDR platform, threat intelligence platform and pattern-based file analysis extends end users' threat detection capabilities, providing real-time insights into emerging cyber threats and enabling proactive mitigation strategies. This comprehensive approach empowers end users to make informed decisions regarding incident response, enhancing their cybersecurity posture and resilience. In essence, AIRCA not only streamlines incident response processes but also fosters a culture of proactive cybersecurity awareness and readiness among end users, ensuring a safer and more secure digital future.

2.3. Understanding the Solution

2.3.1. Project Elaboration

AIRCA is a system that automatically detects, analyses, and responds to security incidents and is fully containerised which makes it lightweight, easy to deploy and ready to use in any environment. AIRCA is based on Wazuh and takes advantage of its multiple features such as vulnerability identification, file integrity monitoring, Mitre TTP monitoring, security alerts generation, active response automation.

Wazuh is a free and open-source unified XDR and SIEM used for security threat prevention, detection, and response that can protect workloads across on-premises, virtualized, containerized, and cloud-based environments (Wazuh, 2023). The agent is one of the core features of Wazuh. AIRCA makes use of this feature because it helps actively monitor the endpoints' network and system behaviour and look for any anomalies and send this information back for analysis.

If suspicious activities are found or any alerts are triggered for the pre-written rules configured in Ossec which is another feature provided by Wazuh then, the script files that are specifically written for such rules will automatically run and perform necessary actions to respond to such threats. This feature to respond to threats is what Wazuh refers as "**Active Response**".

For better correlation analysis, Wazuh is integrated with MISP which is another open-source software solution for collecting, storing, distributing, and sharing cyber security indicators and threats about cyber security incidents analysis and malware analysis (MISP, 2023). Moreover, AIRCA also facilitates the feature of Yara analysis by using File Integrity Monitoring module, and if any of the monitored directory contains any file that matches the Yara rules at any time in Windows Endpoint, then an active-response file runs to quarantine the file into temporary folder.

2.4. Similar Projects

This section includes three projects which are somehow like AIRCA. They are briefly described below:

2.4.1. Project 1 : OpenEDR

OpenEDR is a free and open-source tool for detecting and responding to cyber threats on your computer. It helps analyse and understand potential threats by keeping an eye on activities in real-time. With features like Mitre ATT&CK visibility, it helps you connect the dots and figure out the root cause of any suspicious cyber behaviour. It's like a watchdog for your computer, available for everyone to use, no matter the size of their organization.

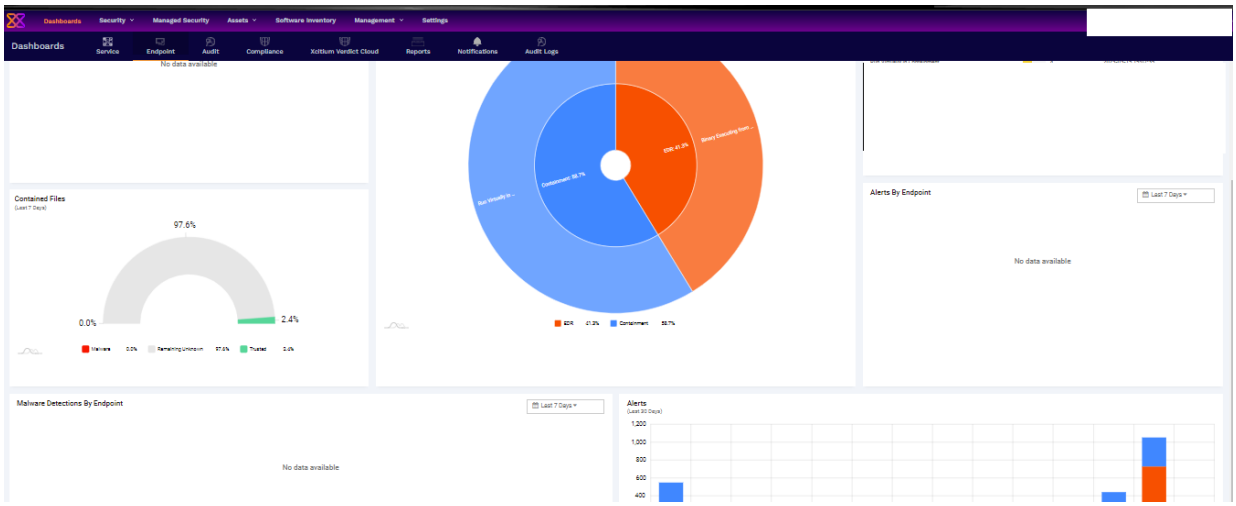


Figure 4 Dashboard of OpenEDR (CyberArts Bilişim A.Ş, 2023)

2.4.1.1. Features Comparison

S.N.	Features	OpenEDR	AIRCA
1.	Vulnerability Identification	✓	✓
2.	Collects, Correlates and Visualizes Logs	✓	✓
3.	Integrated Threat Intelligence	×	✓
4.	Automated Active Response	×	✓
5.	Open-Source	✓	✓

Figure 5 OpenEDR Comparison with AIRCA

2.4.1.2. Comparison Analysis of OpenEDR and AIRCA

Referring to the comparison table above, features like vulnerability identification, log collection, correlation and visualization, open source come by default in OpenEDR but features like Integrated Threat Intelligence with another third-party threat intelligence feeds and automation of active response to threats found don't come by default which make this project AIRCA have a upper hand on these parts than OpenEDR.

[*Learn more about OpenEDR at https://www.openedr.com/.*]

2.4.2. Project 2 : Graylog

Graylog is a log management platform designed to collect, index, and analyses both structured and unstructured data from nearly any source. Graylog serves as a robust platform, facilitating the efficient handling of logs for comprehensive insights. It enables users to make sense of diverse data types, whether structured or unstructured, from a wide range of sources.

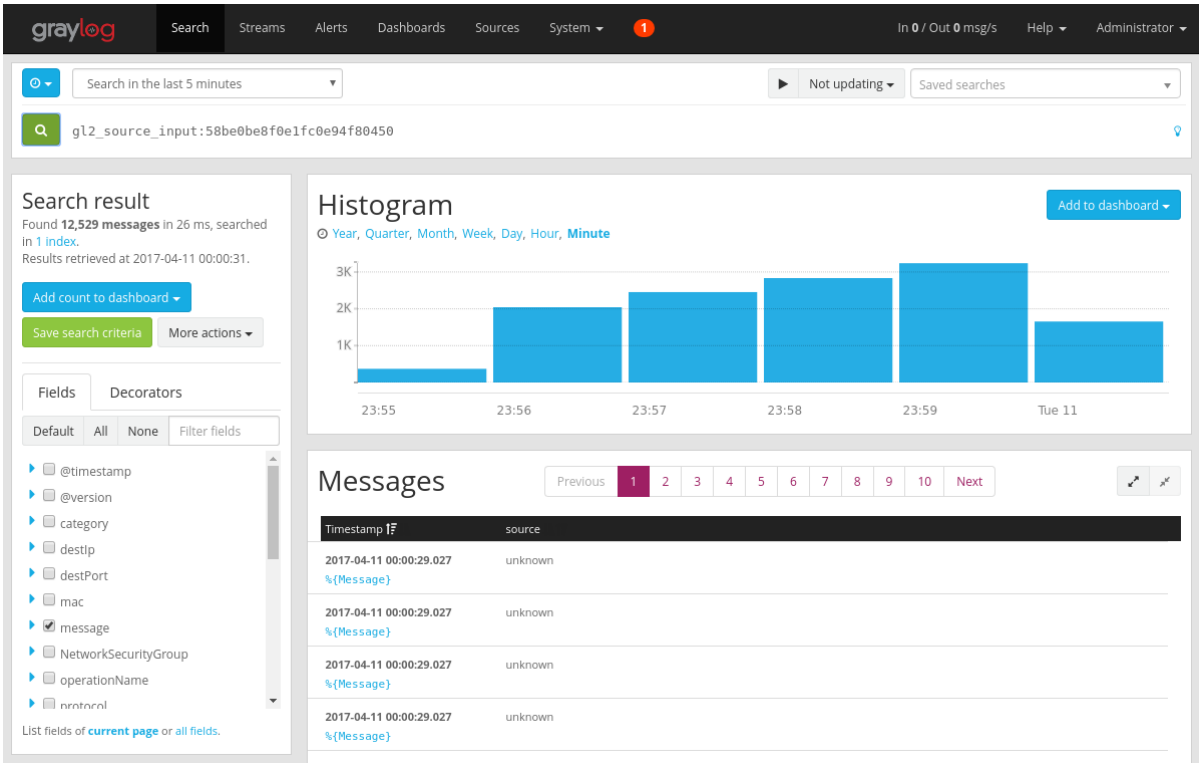


Figure 6 User Interface of GrayLog (Microsoft, 2024)

2.4.2.1. Features Comparison

S.N.	Features	Graylog	AIRCA
1.	Vulnerability Identification	✗	✓
2.	Collects, Correlates and Visualizes Logs	✓	✓
3.	Integrated Threat Intelligence	✗	✓
4.	Automated Active Response	✗	✓
5.	Open-Source	✓	✓

Figure 7 Graylog Comparison with AIRCA

2.4.2.2. Comparison Analysis of Graylog and AIRCA

Referring to the comparison table above, features like log collection, correlation and visualization, open source come by default in Graylog but features like Vulnerability Identification, Integrated Threat Intelligence with another third-party threat intelligence feeds and automation of active response to threats found don't come by default which make this project AIRCA have a upper hand on these parts than Graylog.

[*Learn more about Graylog at https://graylog.org/.*]

2.4.3. Project 3 : Splunk

Splunk is a large-scale data platform designed to streamline the process of gathering and overseeing extensive amounts of machine-generated data, making it easier to search for specific information. This technology finds applications in business and web analytics, as well as in managing applications, ensuring compliance, and enhancing security measures.

Figure 8 User Interface of Splunk Enterprise (Splunk Inc., 2024)

2.4.3.1. Features Comparison

S.N.	Features	Splunk	AIRCA
1.	Vulnerability Identification	×	✓
2.	Collects, Correlates and Visualizes Logs	✓	✓
3.	Integrated Threat Intelligence	×	✓
4.	Automated Active Response	×	✓
5.	Open-Source	×	✓

Figure 9 Splunk Comparison with AIRCA

2.4.3.2. Comparison Analysis of Splunk and AIRCA

Referring to the comparison table above, features like log collection, correlation and visualization come by default in Splunk but features like Vulnerability Identification, Integrated Threat Intelligence with another third-party threat intelligence feeds and automation of active response to threats found don't come by default which make this project AIRCA have an upper hand on these parts than Splunk. Also, it is an enterprise level product which makes it costly.

[*Learn more about Splunk at https://www.splunk.com/.*]

2.5. Resource Requirement

2.5.1. Hardware Requirements

Some hardware recommendation for safer and smooth operation of the system have been listed below:

i. 8GB RAM
ii. 40GB Hard Disk
iii. 2.4 GHz Processor
iv. Network Interface Card (NIC)

2.5.2. Software Requirements

The software required for the system have been listed below:

i. **VMware:** It will be used for creating virtual machines that will serve as server and client for the project.

ii. **Linux:** It will be used to host the servers for the project.

iii. **Windows:** It will be used as an endpoint device which will be monitored for any threat detection.

iv. **Python:** It will be used for installing various libraries that will be needed for our ELK stack deployments. It will also be used for writing automation scripts for active response.

v. **Yara:** It will be used to write rules that will be used for finding any malicious patterns in a file.

vi. **Wazuh:** It is an ELK stack based open-source SIEM and XDR project that is widely used all over the world due to its stable and unique features for detection and response. It will be implemented for this project since it's already free to use.

vii. **Malware Information Sharing Platform (MISP):** It is yet another widely used and popular open-source platform that is used by cyber threat intelligence teams all over the

world to share malware feeds which will be used in this project as well for malware detection.

viii. **Docker:** It will be used later in this project to deploy the servers for scalability and to reduce large hardware consumption.

Chapter III : Development

3.1. Project Methodology

DSDM, or Dynamic Systems Development Method, is an agile project management and software development framework that prioritizes user involvement, incremental delivery, flexibility, continuous testing, and reversible changes to enhance adaptability and responsiveness throughout the development lifecycle (Aiman Khan Nazir, 2017).

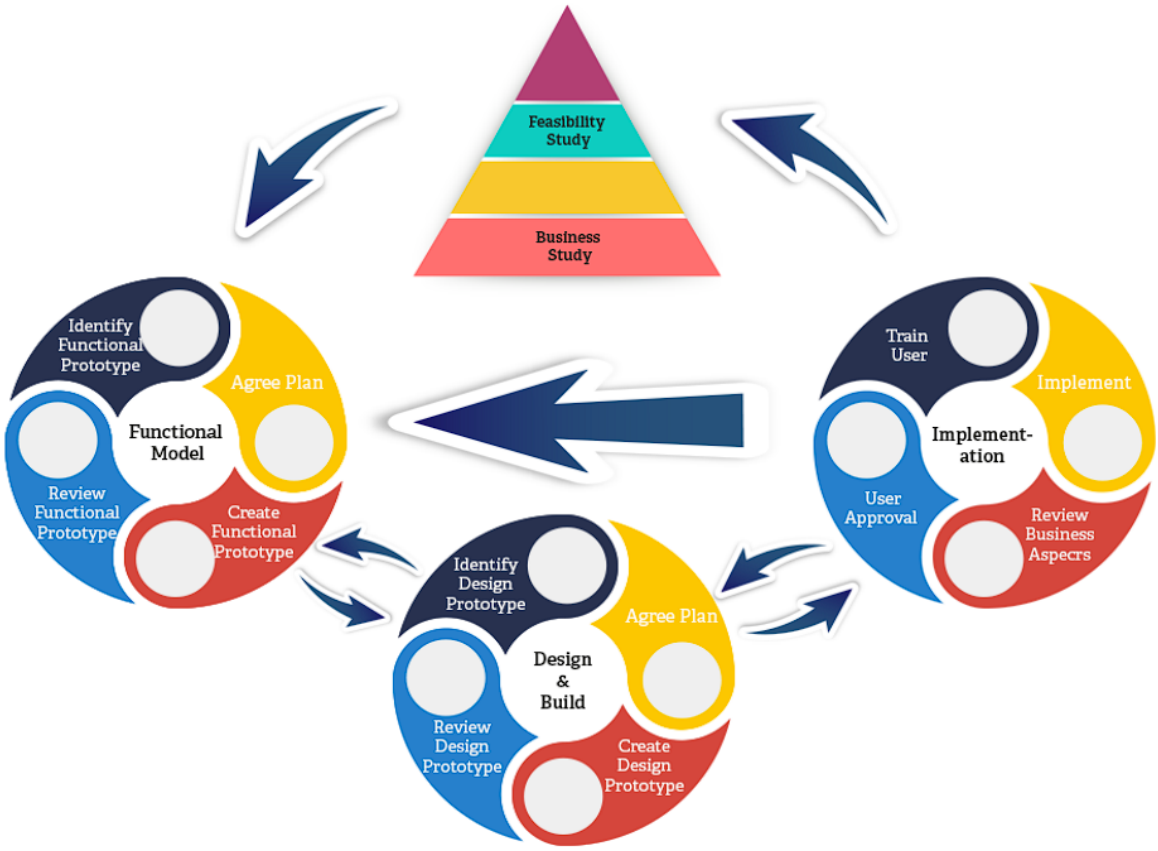


Figure 10 DSDM Process Model (Aiman Khan Nazir, 2017)

For this project, DSDM project was selected since it is driven by its user-centric approach, supporting active involvement throughout development, and its flexibility, enabling better adaptation to evolving cyber threat scenarios, aligning well with the project's dynamic and security-sensitive nature.

[*Note: For better understanding the DSDM model, please refer to "Defining DSDM".*]

3.2. Phases of DSDM Methodology in accordance with project

3.2.1. Phase 1: Pre-Project Phase

In the pre-project phase, Project suggestion and selection activities was undertaken. During this stage, the topic was reviewed by the supervisors, and the tools required were validated. Following the topic selection, which centred around Automated Incident Response for Cyber Anomalies, Insights on incident response automation from various sources were collected. After thorough analysis of the gathered resources and documents, a project proposal was crafted. Upon approval from the supervisor, the preparation for the project started.

3.2.2. Phase 2: Project - Life Cycle Phase

The second stage of the DSDM approach consists of five sub-phases. The actions carried out within these sub-phases to create the suggested solution are outlined below.

3.2.2.1. Sub-Phase 1: Feasibility Study

During this sub-phase, an evaluation was conducted to assess if DSDM is the most suitable approach for the project, along with estimating its potential costs and technical viability. Additionally, a feasibility study was undertaken, focusing on legal considerations, time constraints, and technical feasibility of the proposed solution to find out the likelihood of project success.

Critical issues and viable solutions from the supervisors were received. Consequently, the necessary resources for the automated system were initially identified, with the entire cost not being considered as the FYP was an individual project. This phase entailed developing the outline plan for the project, as well as defining the project's end products.

3.2.2.2. Sub-Phase 2: Business Study

During this sub-phase, Identification of the business domain of the project was done. Based on the research and survey results, it was found that the proposed solution could greatly benefit businesses by providing a centralized system with incident response capabilities. This system would help minimize human errors, gather threat intelligence feeds, present results in a more

user-friendly format, and save users time and effort. Therefore, during this phase, the team discussed the proposed solution and its potential applications with the supervisors.

3.2.2.3. Sub-Phase 3: Functional Model Iteration

In this sub-stage, the project's development officially started, involving tasks such as analysis, coding, and prototyping. The knowledge gained from previous phases and sub-phases informed the initiation of building a functional prototype for the proposed solution. Once all the essential features of the solution were implemented, a comprehensive flowchart of the program and the entire system was created using the 'draw.io' application.

The main development and configuration phase was carried out by dividing it into multiple iterations, which are listed below.

- First Iteration – Selection of required tools and system resources.
- Second Iteration - Deployment of Wazuh and MISP.
- Third Iteration - Integration of Wazuh and MISP.
- Fourth Iteration - Development of detection rules and active response.
- Fifth Iteration - Customization of Wazuh and completion of project.

3.2.2.4. Sub-Phase 4: System Design and Build Iteration

During this sub-phase, the supervisors reviewed the design and functionality of the system, and subsequent enhancements were made based on their feedback. Following an evaluation of the process and source code of the applications that were used, adjustments were made as per the supervisors' recommendations. Additionally, various types of testing, such as unit testing, system testing, and security testing, were conducted using successful sample data.

3.2.2.5. Sub-Phase 5: Implementation

Due to time constraints, the project couldn't be implemented in a production environment. Nevertheless, a brief demonstration of AIRCA was presented to a group of 20 individuals, comprising individuals from different companies and students from Islington College. Following the demonstration, a post-survey form was distributed to gather their feedback.

3.2.3. Phase 3: Post-Project Phase

This was the final phase of the DSDM methodology, which involved measuring how the system performed after deployment and determining any additional changes needed with final documentation for the project in hand.

3.2.4. Justification for selecting DSDM Methodology

The reasons for using the DSDM methodology for developing this project are as follows:

- It can be utilized for any project, whether its business related or technical.
- One important idea in DSDM is that things aren't perfect at the start, so the methodology can make them better with supervisor’s feedback.
- At each iteration, a minimum usable solution and basic version of what is done is presented to the supervisor, making sure the project runs smoothly.
- The iterative structure of DSDM allows to go back to any of the previous phases to make any further adjustments to the system.
- It enables regular and clear communication which aids in project quality improvement.

3.3. Survey Analysis

3.3.1. Pre-Survey Analysis

The pre-survey for the project was carried out among various people from different background from people with no IT background to people working in Cyber Security as their professions. The survey form was majorly filled by students at Islington College and cyber security professionals from Nepali Cyber Security Companies such as Vairav Tech, Cryptogen Nepal, etc. These companies have been using different vendors' SIEM and other security solutions as well as their own in-house developed solutions with different features for incident response to threats.

Most of the survey participants were familiar with all the tools and technologies mentioned in the survey questions. All of them agreed that a system implementing the tools and technologies would probably help better increase defence against various cyber anomalies. They suggested some of their idea for the success of the project and mentioned that the system would help prevent some of the incidents that they faced in recent years. And through the survey form responses, it was concluded that automated incident response with integrated threat intelligence for responding to cyber threat and anomalies is very important and can help organizations or even individuals protect their assets from cyber threat actors.

The survey form questions can be found at *Pre-Survey Questions*, survey form sample can be found at *Pre-Survey Sample* and the responses can be found at *Pre-Survey Responses*. Please note that some of the survey responses were cleaned out because they contained unexpected responses.

3.3.2. Post-Survey Analysis

The post-survey for the project was carried out among the people who were previously involved in pre-survey phase of the project. However this time, people with some level of cybersecurity knowledge were involved and feedback was received from 20 participants. Most of the people had experience of using different vendors' SIEM and other security solutions as well which was the reason why the post-survey was carried out from them for better feedback.

All the participants were satisfied with the level of visibility/correlation between security events provided by AIRCA. Majority of them agreed that the integration of the threat intelligence platform and malware pattern analysis with Yara rules with the SIEM improved the threat detection and mitigation capabilities. And through the survey form responses, it was concluded that with all the integration, detection, response features for responding to cyber threat and anomalies is very important and can help organizations or even individuals protect their assets from cyber threat actors.

The survey form questions can be found at *Post-Survey Questions*, survey form sample can be found at *Post-Survey Sample* and the responses can be found at *Post-Survey Responses*. Please note that some of the survey responses were cleaned out because they contained unexpected responses.

3.4. Flowchart of the System

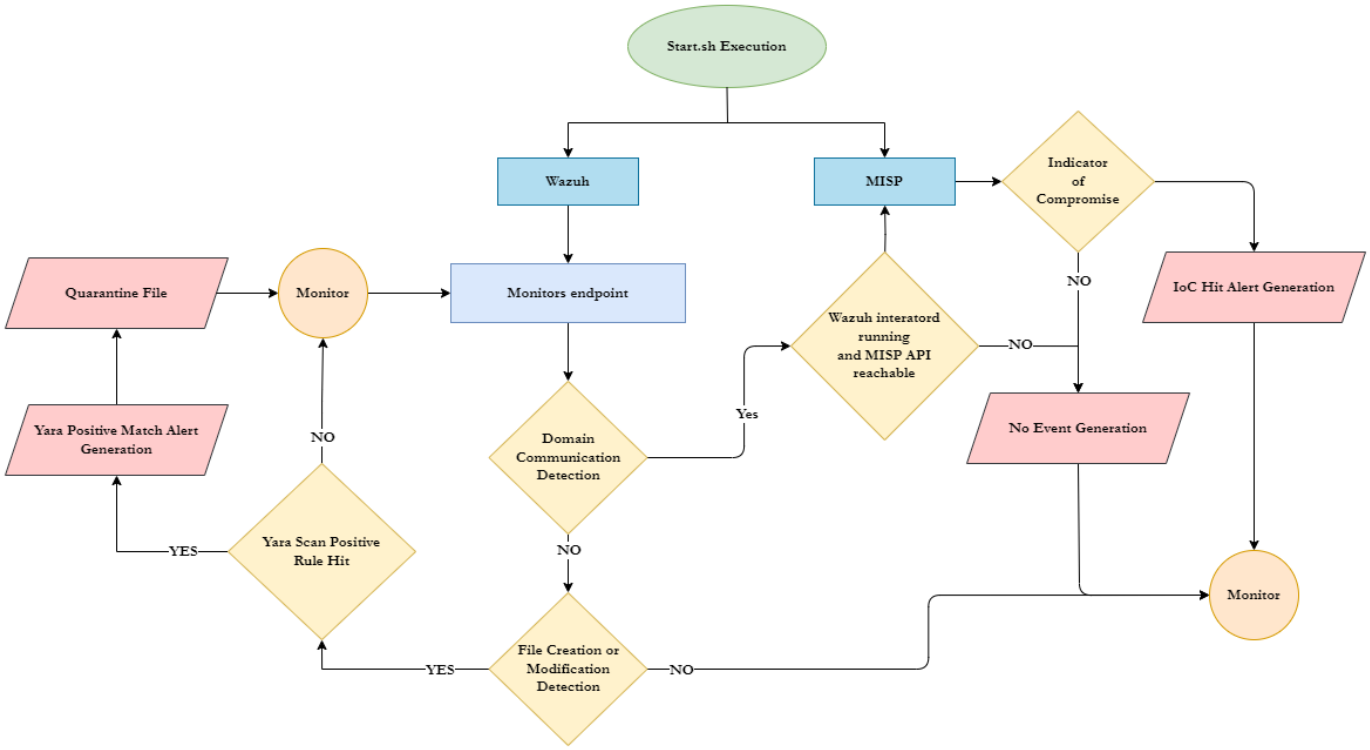


Figure 11 Event monitoring and handling flow of the system

3.5. Development of the System

Following the completion of the flowcharts of the system, the development, or coding, phase of the proposed solution proceeded. As previously stated, the proposed solution was created using the DSDM (Dynamic System Development Method) software development methodology. Development of the configured systems was completed during the Functional Modal / Prototype Iteration sub-phase of the DSDM process. The development of any application using this methodology must be carried out in distinct iterations until the result is completed within the timeframe specified. This specific project was built in five different iterations which are listed as follows:

- First Iteration - Selection of required tools and system resources.
- Second Iteration - Deployment of Wazuh and MISP.
- Third Iteration - Integration of Wazuh with MISP.
- Fourth Iteration - Development of detection rules and active response.
- Fifth Iteration - Customization of Wazuh and completion of project.

Each iteration produces a fully functional piece of application that has been discussed and adjusted with supervisors. Moreover, the topics discussed below provides details about each iteration and the main outcomes generated throughout the project. Each step of progress has been clearly described, and its unique features have been thoroughly explained.

3.5.1. First Iteration - Selection of required tools and system resources

In this iteration of system development, setting up and testing all the necessary tools for building the proposed solution were focused. The focus was on choosing the right tools and techniques, as they play a crucial role in shaping how the project progresses and what the result will look like. These tools and techniques are vital for laying a strong foundation and ensuring the success of the project.

The primary tools that were installed and used in this iteration are as follows:

- Three virtual machines in VMware were deployed where the two machines were Linux, and one was Windows.
- Required hardware resource were provided for each of the virtual machines. The machines included: windows-endpoint, ubuntu-endpoint and Aircа.
- In Aircа machine, tools like docker and docker-compose were installed and tested.

The basic system/network diagram of the machines can be found at *System & Network Diagram* and virtual machines resource information can be found at *Machines Resource Information*.

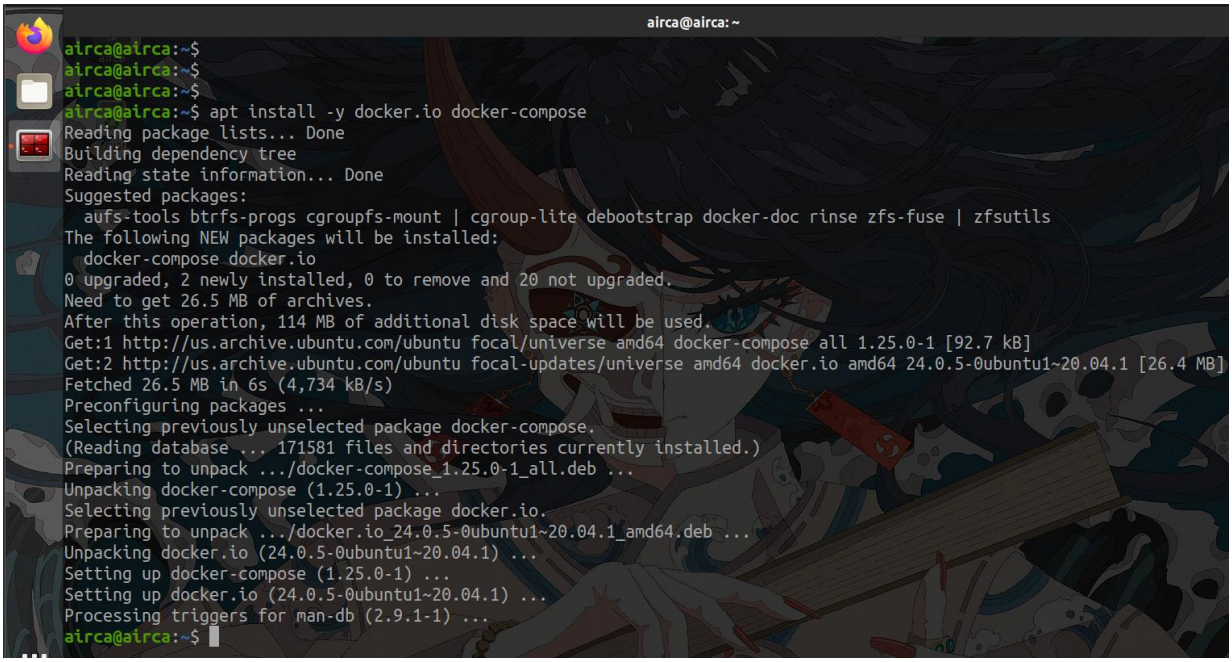


Figure 12 Installing docker and docker-compose

In the above figure, docker and docker-compose were both installed using apt in Aircа machine, this will be needed for later development and deployment iterations of the system.

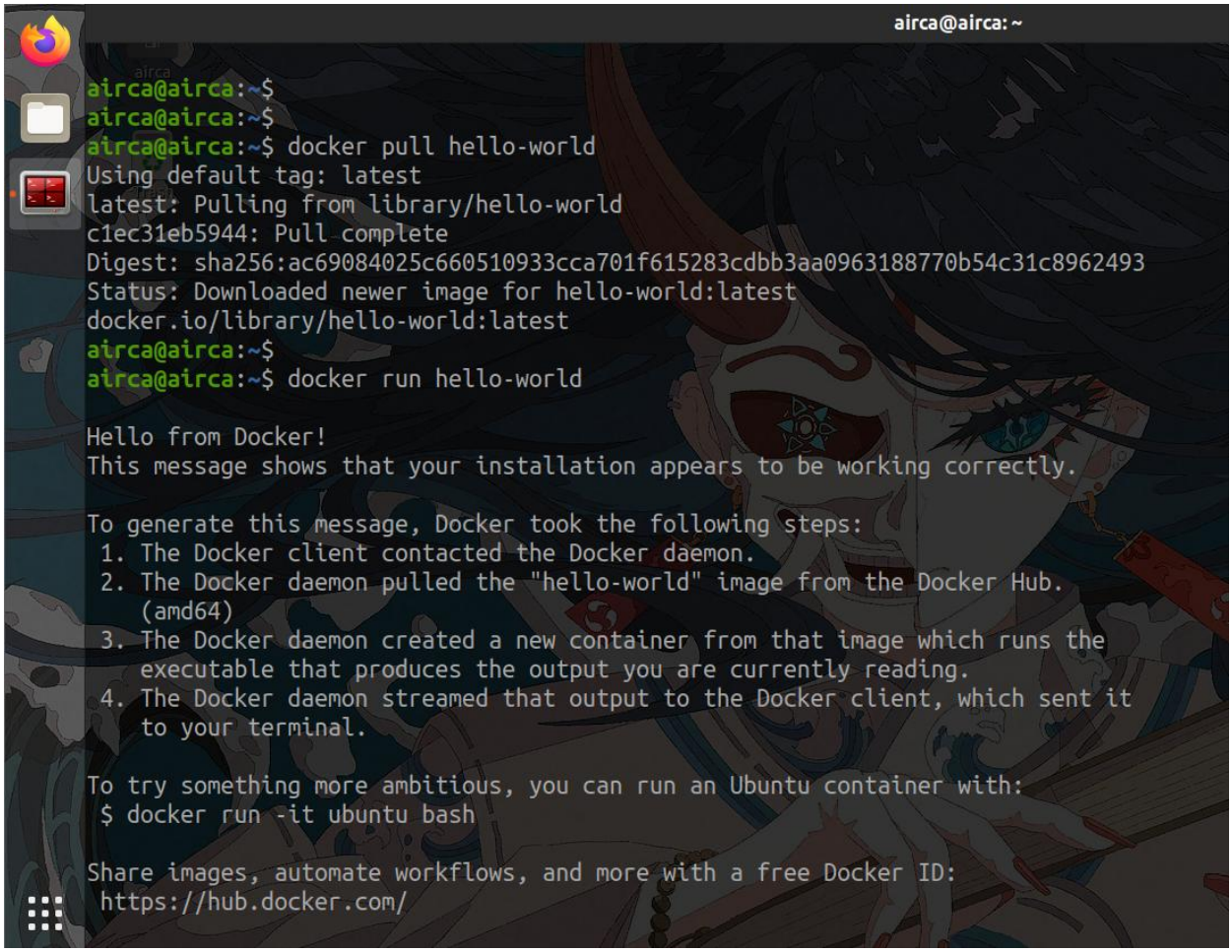


Figure 13 Verifying installation of docker

In the above figure, docker was used to pull an image named “hello-world” and run a container in Airca machine to test if docker installation was a success.

3.5.2. Second Iteration - Deployment of Wazuh and MISP

In this iteration of the system development, setting up and deploying Wazuh and MISP was prioritized as they were the main applications used in this project.

The main works that were done in this iteration are as follows:

- Git was pre-installed like docker and docker-compose from first iteration. To install git, “apt install git” can be used.
- Wazuh and MISP repositories were cloned from GitHub using Git clone and deployed each one of them independently in this iteration.
- Agents were installed in the windows and ubuntu endpoint after deploying Wazuh.

The deployment process of Wazuh and MISP can be found in figures below.

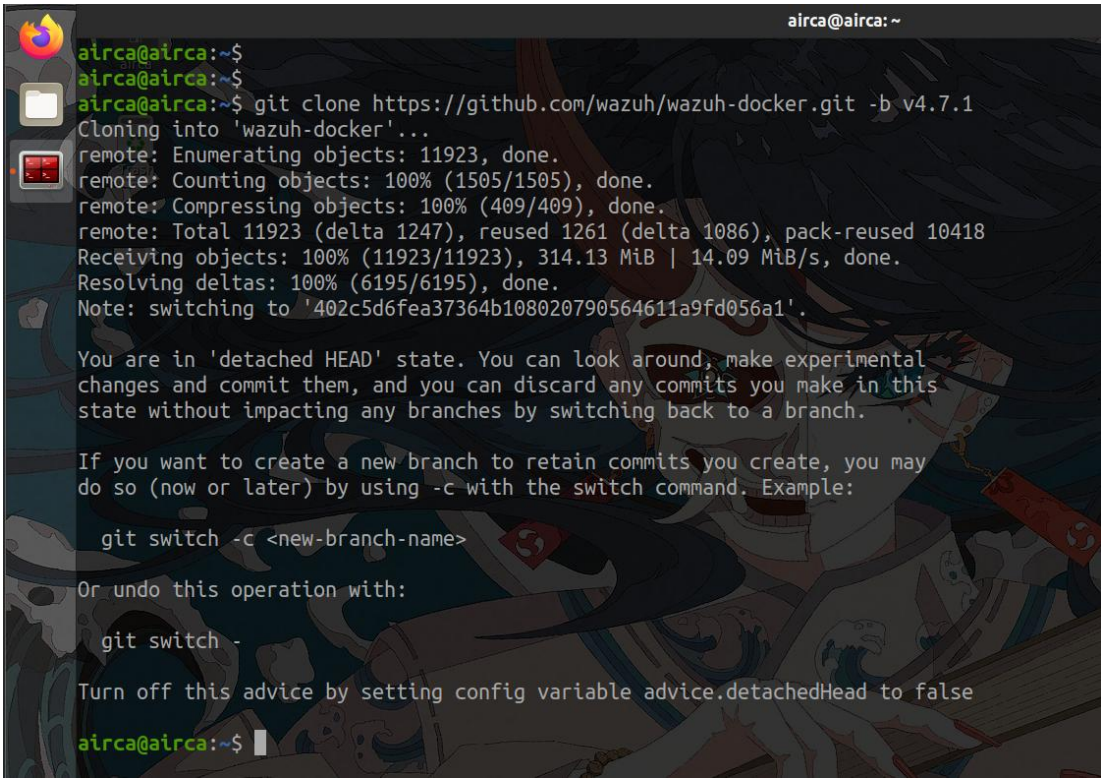


Figure 14 Cloning wazuh-docker repository

In the above figure, git was used to clone wazuh-docker from Wazuh’s official GitHub repository. The repository contained most of the default configuration files.

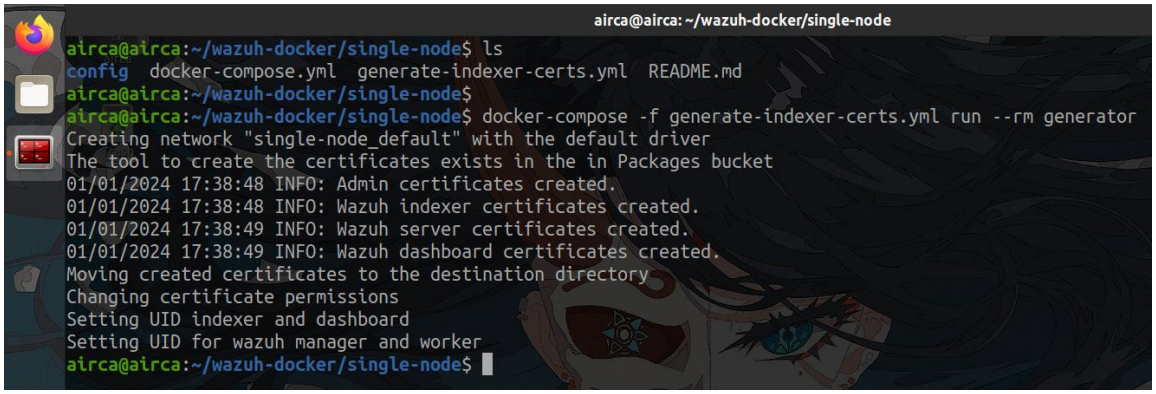


Figure 15 Generating certificates for Wazuh indexer, server, and dashboard

Docker-compose was used to generate self-signed certificate for Wazuh's components using generate-indexer-certs.yml file which was already present in default configuration file.

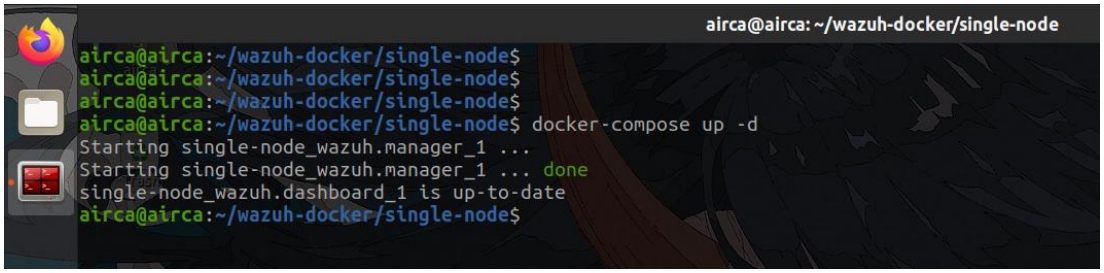


Figure 16 Starting Wazuh via docker-compose

Docker-compose was again used to run the containers from default docker-compose.yml file that was residing in single-node directory. Note that, docker-compose up picks up docker-compose.yml by default to run the containers from the images in the compose file.

After the deployment of Wazuh was completed, the dashboard could be accessed from "https://localhost" itself. Wazuh dashboard overview can be found at *Wazuh Dashboard Overview*.

After verifying that dashboard was accessible across the network using the IP of the Airca machine. Agents were installed in the endpoints who which the process for installation can be found at *Agent Installation Process for Endpoints*. Then, Git was again used to clone the MISP repository from GitHub of which the figures can be found below.

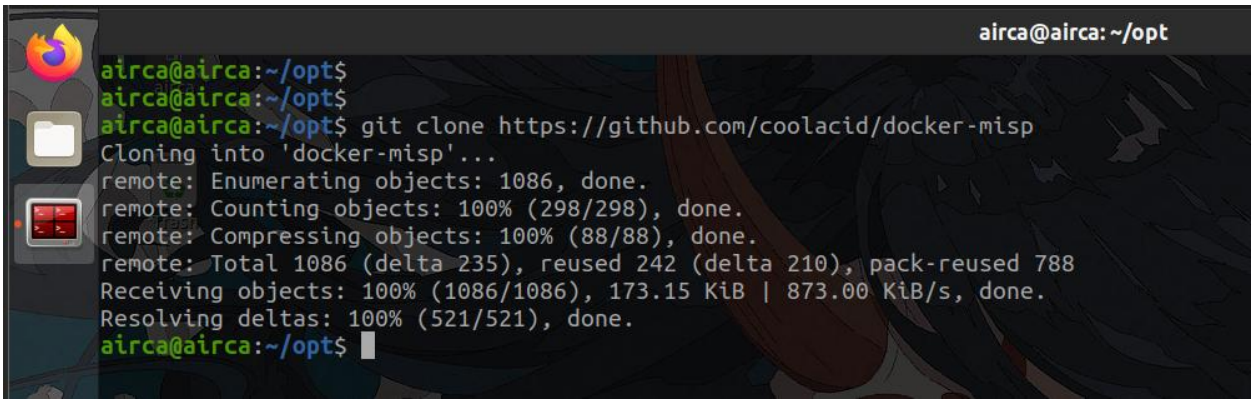


Figure 17 Cloning docker-misp repository

In the above figure, git was used to clone docker-misp from coolacid's GitHub repository. The repository contained pre-configured files that were production-ready.

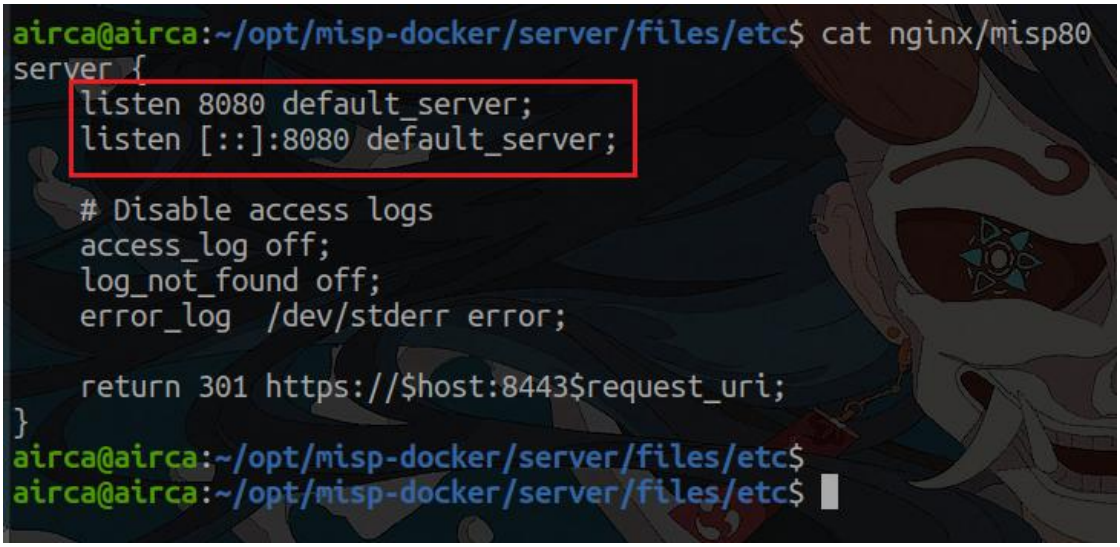


Figure 18 Changing MISP http port to 8080

The default port for MISP were changed to 8080 for HTTP and 8443 for HTTPS respectively because Wazuh was also using the default port for HTTP and HTTPS i.e. 80 and 443. This was done since both Wazuh and MISP were going to be deployed in the same machine using docker.

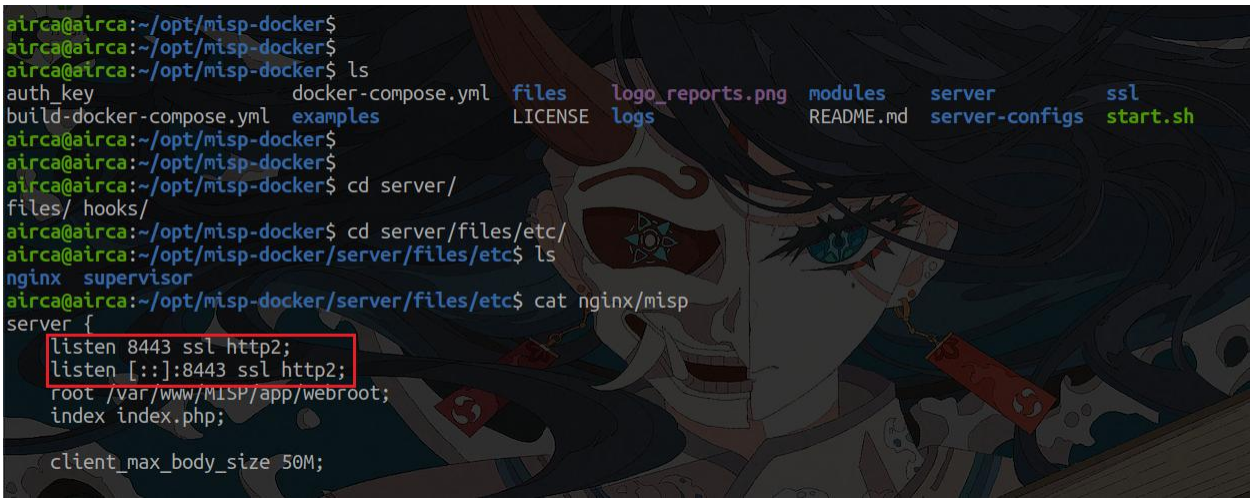


Figure 19 Changing MISP https port to 8443

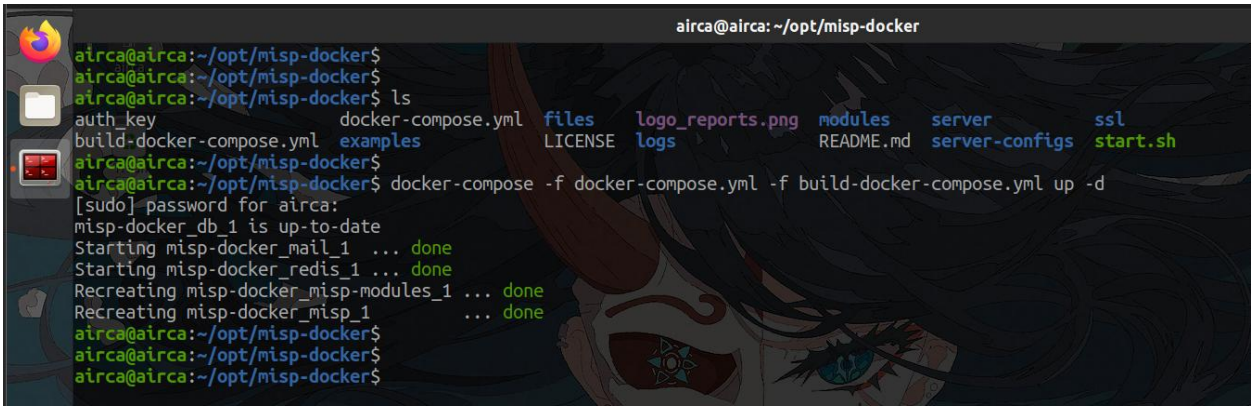


Figure 20 Starting MISP via docker-compose

After the deployment of MISP was completed, the dashboard could be accessed from "https://localhost" itself. MISP dashboard overview can be found at *MISP Dashboard Overview*.

3.5.3. Third Iteration - Integration of Wazuh and MISP

In this iteration of the system development, integration of Wazuh and MISP for threat intelligence analysis correlation was done.

The main works that were done in this iteration are as follows:

- A single docker-compose file was created to deploy Wazuh and MISP with a custom docker network which made communication between Wazuh and MISP possible.
- Integrator module was enabled in Wazuh for MISP.
- MISP Alerts were generated in Wazuh.
- Start.sh and Stop.sh script files were developed throughout third to fifth iteration of the project where Start.sh automates setting up all the configurations in the containers and Stop.sh just stops all the containers.

The single docker-compose file that was used to setup the required resources and configurations to deploy necessary containers for Wazuh and MISP can be found at *Docker Compose Configuration*.

Figure 21 MISP Integration rule in ossec.conf

In the figure above, custom-misp.py has been set in the integration field in ossec.conf which gets triggered after some of the Sysmon events are generated in Wazuh. This python script then tries to connect with MISP API and triggers success rule id, or if API fails to connect, it will also throw a rule id for that. MISP API integration tests can be found at *Test Case 4* and *Test Case 5* whereas Sysmon events test can be found at *Test Case 3*.

Before integration with MISP was configured, Sysmon was configured in the Windows endpoint with default configuration file. This process was done throughout third and fourth iteration of development phase.

The installation process for Sysmon can be found at: *https://www.blumira.com/enable-sysmon/* and the Sysmon rulesets used can be found at *Sysmon xml rule* whereas the custom-misp.py can be found at *Custom MISP python script*.

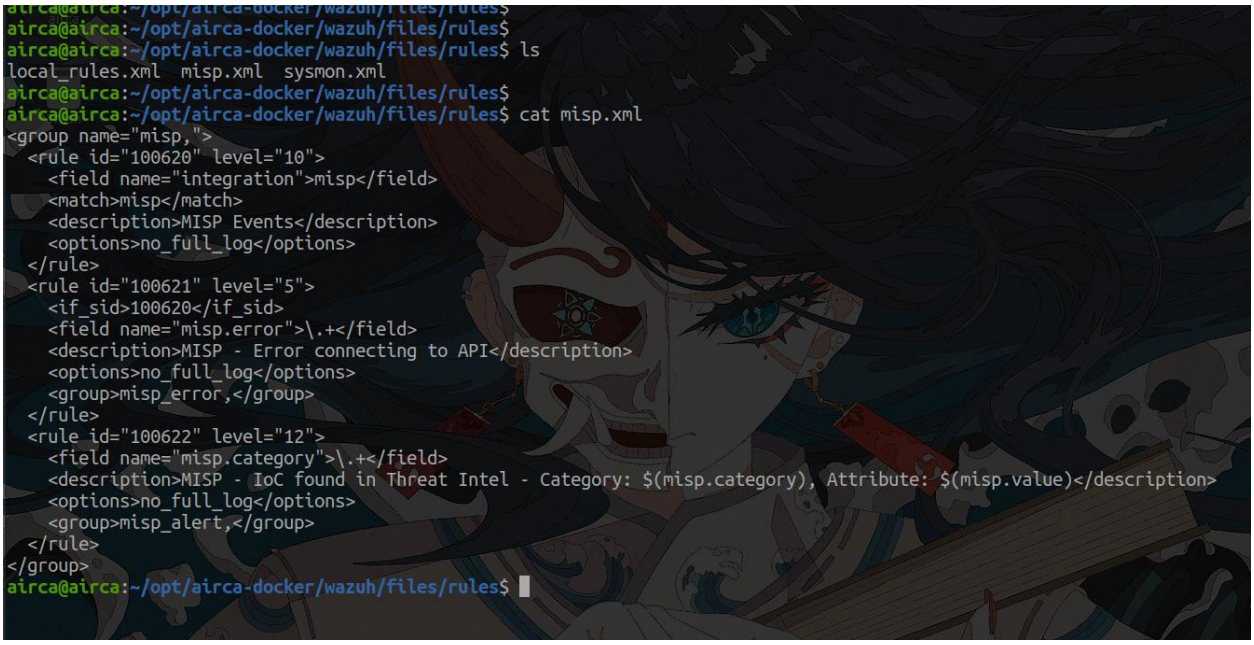


Figure 22 MISP Ruleset for alert generation in Wazuh

The ruleset xml file above in the figure handles the alert generation for each rule id that the integration custom-misp.py throws as result.

The final Start.sh script and Stop.sh script file that was developed over the time of development phase iterations can be found at *Start.sh script* and *Stop.sh script*.

Throughout the iterations, every file and folder were structured and organized which can be seen in the figure below.

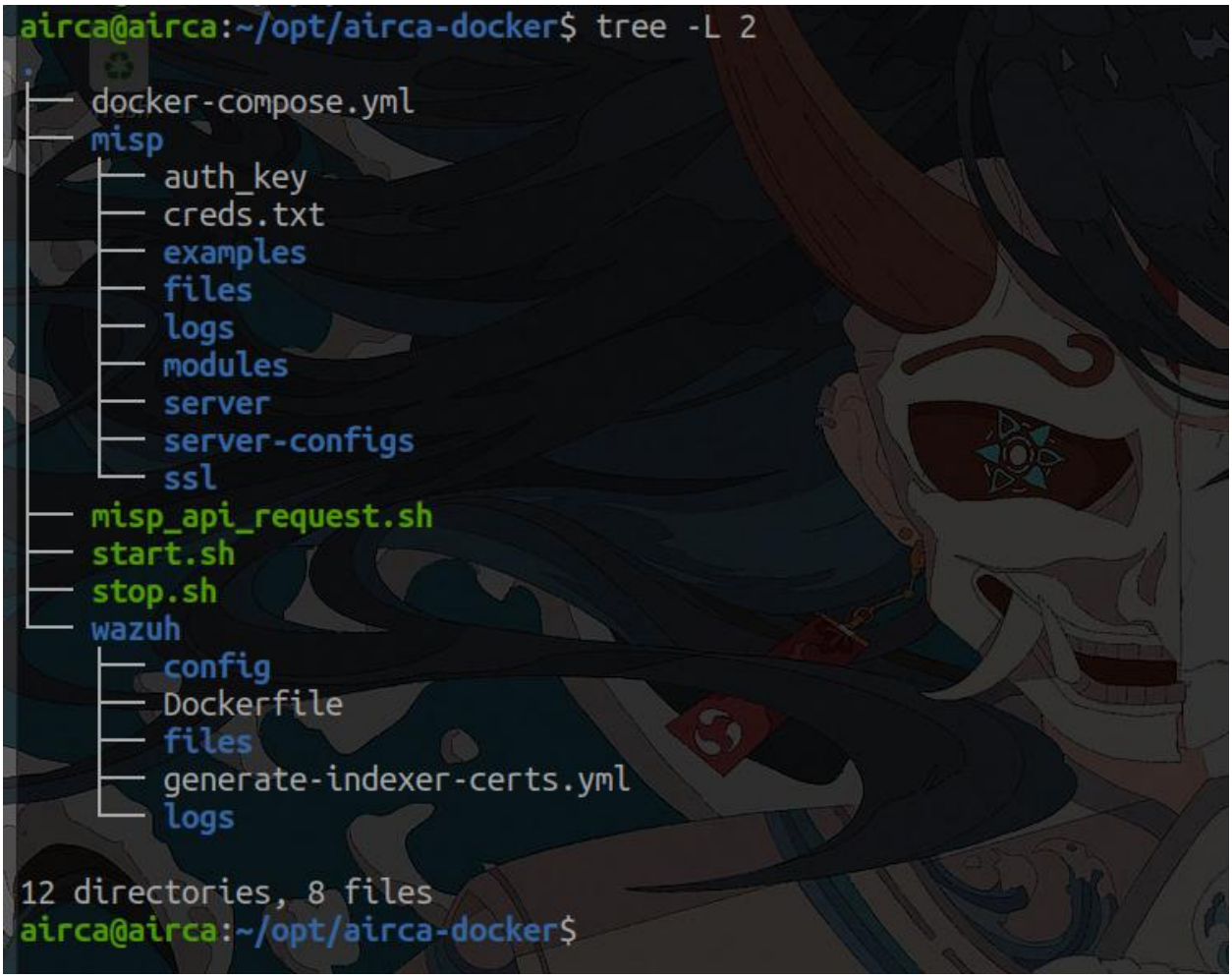


Figure 23 Folder hierarchy of the project

3.5.4. Fourth Iteration - Development of detection rules and active response

In this iteration of the system development, detection rules and active response was prioritized and was developed parallelly with third iteration as they were the main objective of this project.

The main works that were done in this iteration are as follows:

- MISP IoC positive alert hit generation.
- Yara scan positive alert hit generation.
- Quarantine suspicious files after positive Yara hit using active response.

MISP API integration tests can be found at *Test Case 4* and *Test Case 5* whereas Sysmon events test can be found at *Test Case 3*. The development of it can be found in the third iteration.

For Yara analysis, local_decoder xml file and local_rule xml file was created in Airca machine. To setup the overall Yara detection for windows and ubuntu endpoint, the official documentation of Wazuh was followed which can be found at: *https://documentation.wazuh.com/current/proof-of-concept-guide/detect-malware-yara-integration.html*.

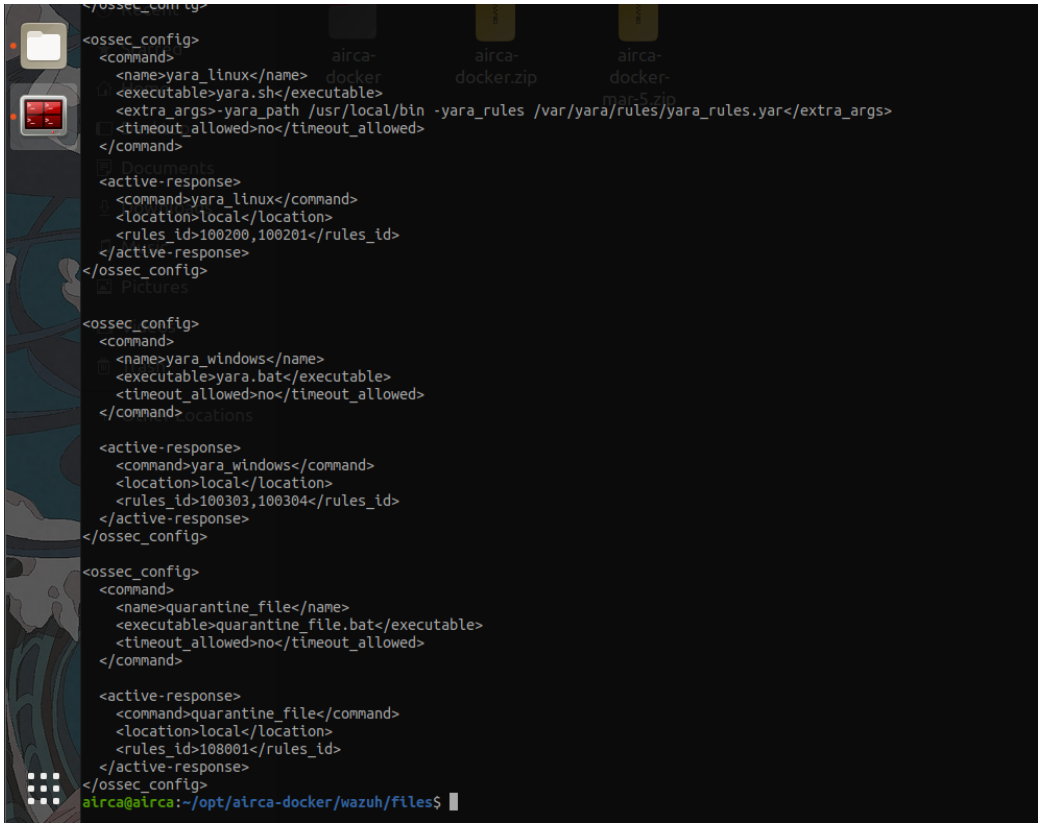


Figure 24 Enabling Yara in ossec.conf

To enable the Yara scan and active response script for the suspicious file, rules were written in ossec.conf which can be seen in the figure above. In the rule, yara_windows command runs if rule id matches 100303 or 100304 in windows and for yara_linux if rule id matches 100200, 100201 in ubuntu. After the Yara scan completes in windows and if the file is a positive Yara rule match then, the quarantine_file script runs to quarantine the file.

Yara analysis scan alert and file quarantine tests can be found at *Test Case 8* and *Test Case 9*.

The Yara batch file for windows is at *Yara scan script for windows* and Yara script file for ubuntu is at *Yara scan script for ubuntu* where as active response batch file for windows is at *Active response script*.

3.5.5. Fifth Iteration - Customization of Wazuh and completion of project

In the final iteration of this project, customization of Wazuh was done where custom logos of AIRCA was placed in loading screens, login page, and in other places as well. Customization of dashboards was also tested but unfortunately, that wasn't possible because all modules and plugins in the dashboard configuration were interconnected with each other and since, time constraints was there in the project, it was left out.

The customization of the dashboard can be found at *Test Case 10* and customization process can be found at *Customized Screen Process* and Overview at *Customized Screen Overview*.

Chapter IV : Testing and Analysis

4.1. Testing

For validating the system's functionality, multiple types of testing must be conducted to address any faults or problems that may arise throughout the testing process. Testing is critical in system development and cannot be under looked. Several test cases must be done to ensure that the system is proper and reliable. As a result, several types of testing, including as unit testing, system testing, and security testing have been carried out for testing the overall resilience of AIRCA.

4.1.1. Unit Testing

Unit testing is a method that checks individual parts to make sure they work correctly. Its main aim is to confirm that every piece of code in the software works as it should (Amazon Web Services, Inc. , 2024). This system underwent twenty individual unit tests, as detailed below.

4.1.1.1. Test Plan

Test Cases	Objectives	Results
Case 1	To test that Wazuh and MISP starts with no errors.	Successful
Case 2	To test if the Wazuh UI can be changed to dark mode.	Successful
Case 3	To test if new user can be added in Wazuh.	Successful
Case 4	To test if newly added can login into Wazuh.	Successful
Case 5	To test if admin can modify user's role in Wazuh.	Successful
Case 6	To test if admin can delete users in Wazuh.	Successful
Case 7	To test if new user can be added in MISP.	Successful
Case 8	To test if newly added can login into MISP.	Successful
Case 9	To test if admin can modify user's role in MISP.	Successful
Case 10	To test if admin can delete users in MISP.	Successful

Table 2 Test Plans for Unit Testing

4.1.1.1.1. Test Case 1

Unit Testing	Objectives
Objective	To test that Wazuh and MISP starts with no errors.
Action	Start.sh script and docker-compose was executed.
Expected Result	Containers should start without any errors.
Actual Result	Containers started without any errors.
Conclusion	Test successful.

Table 3 Unit Testing - Test Case 1

Evidence

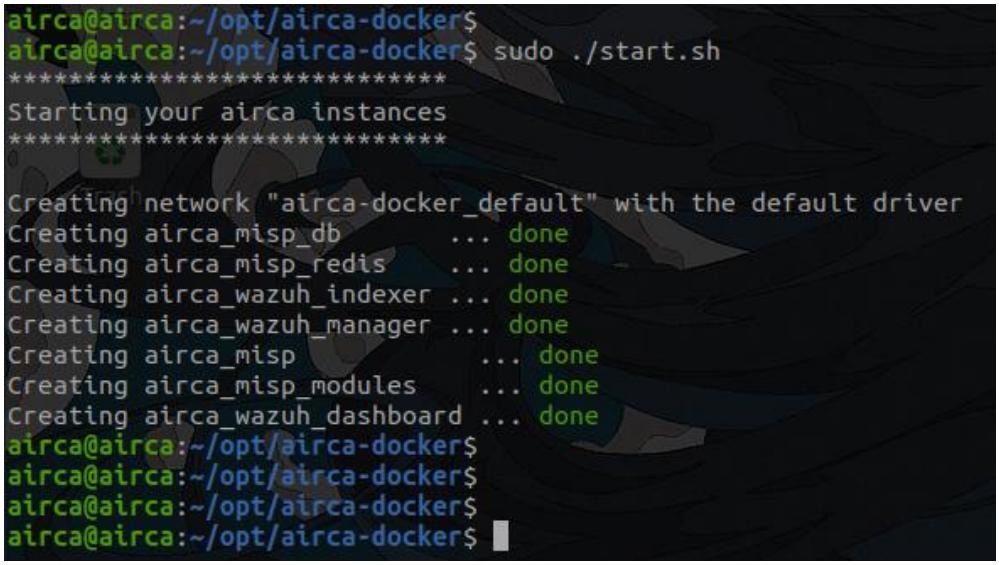


Figure 25 Executing start.sh script file

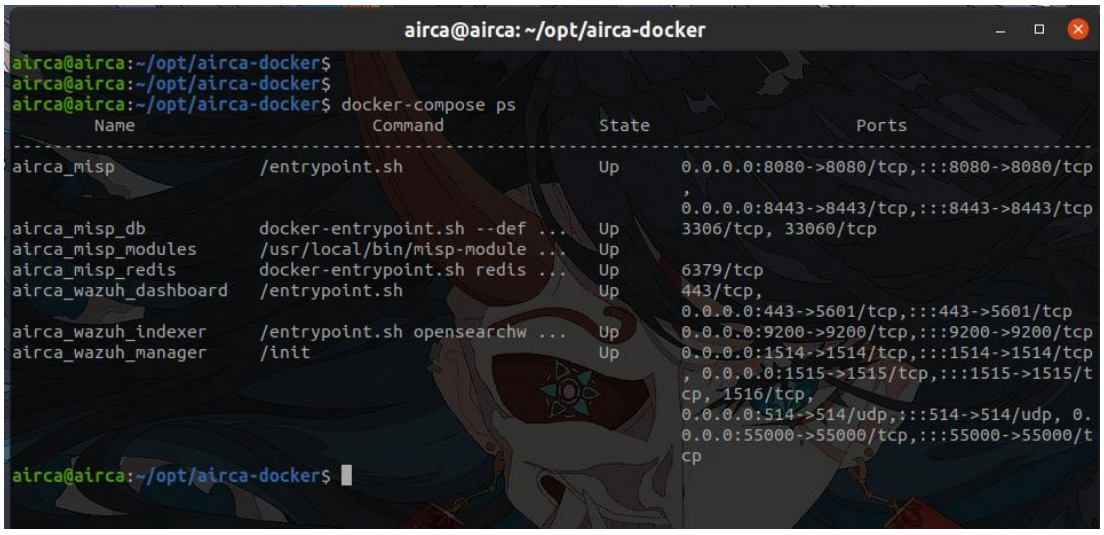


Figure 26 Checking created docker containers of the system

4.1.1.1.2. Test Case 2

Unit Testing	Objectives
Objective	To test if the Wazuh UI can be changed to dark mode.
Action	Toggled dark mode theme in Advanced Settings.
Expected Result	Dark mode theme should be enabled.
Actual Result	Dark mode theme was enabled.
Conclusion	Test successful.

Table 4 Unit Testing - Test Case 2

Evidence

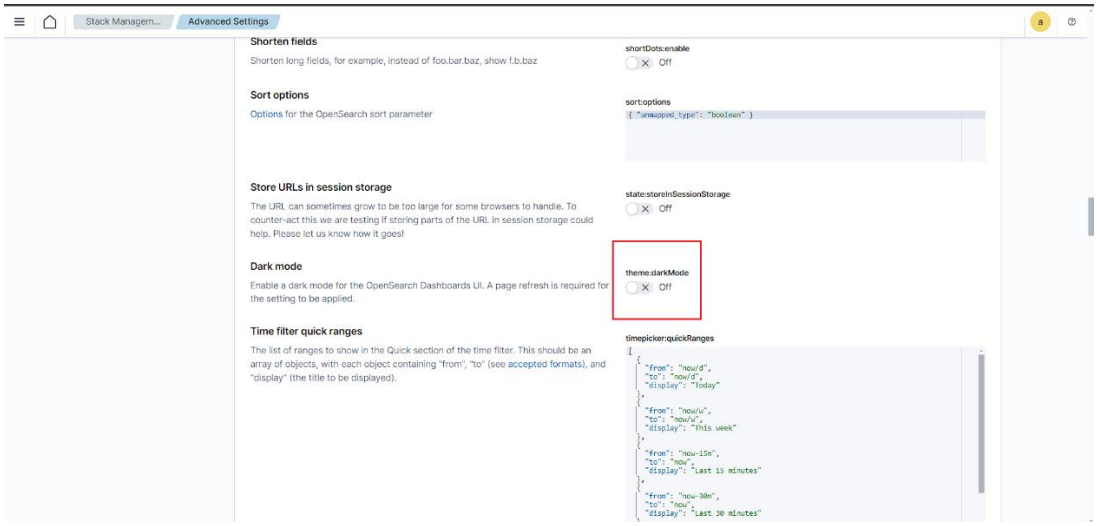


Figure 27 Wazuh - Dark Mode Disabled

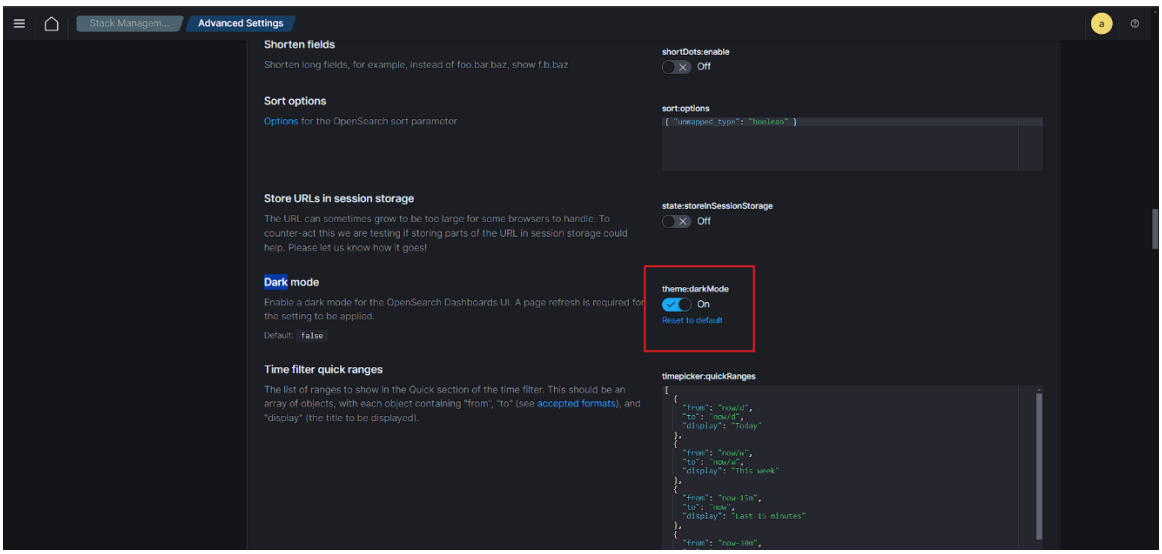


Figure 28 Wazuh - Dark Mode Enabled

4.1.1.1.3. Test Case 3

Unit Testing	Objectives
Objective	To test if new user can be added in Wazuh.
Action	Added new user named analyst in Internal Users settings with readall permission.
Expected Result	User analyst should be created.
Actual Result	User analyst was created successfully.
Conclusion	Test successful.

Table 5 Unit Testing - Test Case 3

Evidence

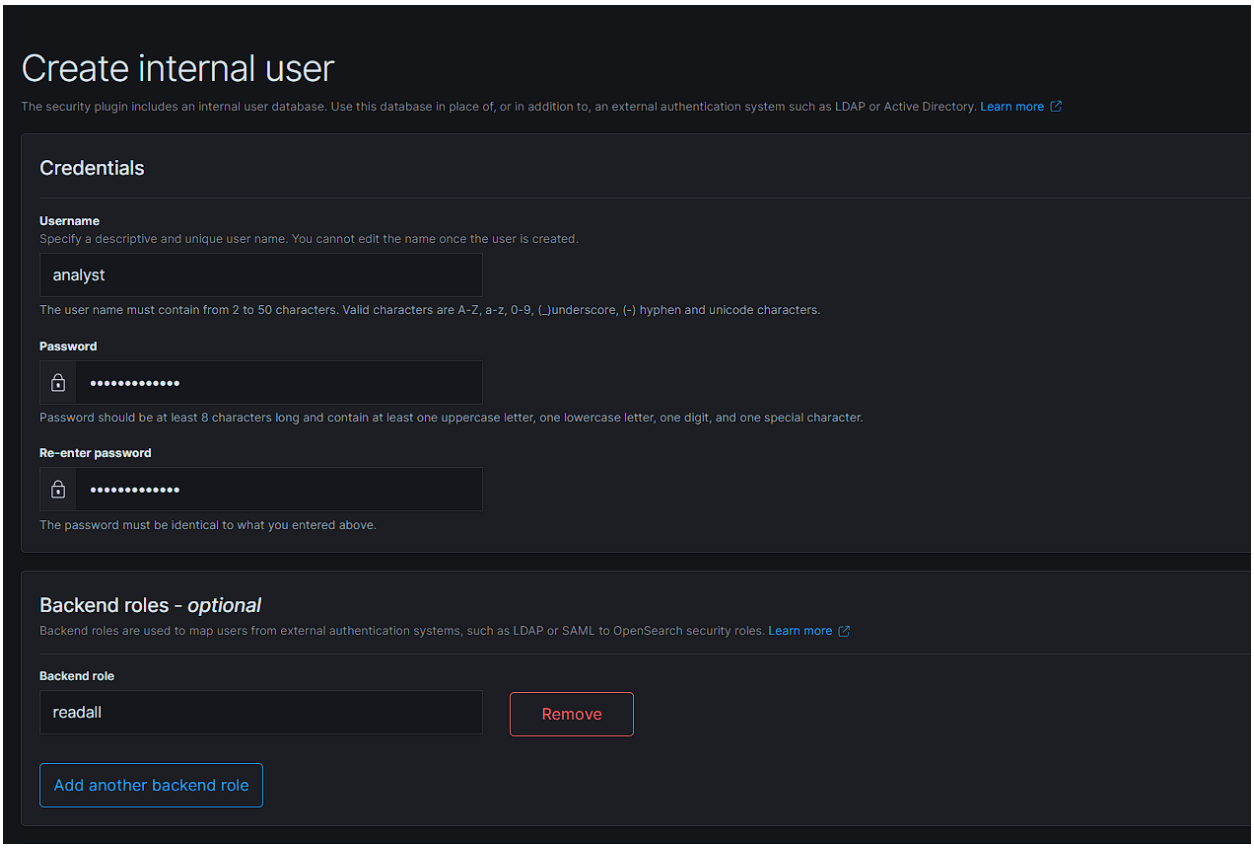


Figure 29 Creating user named analyst

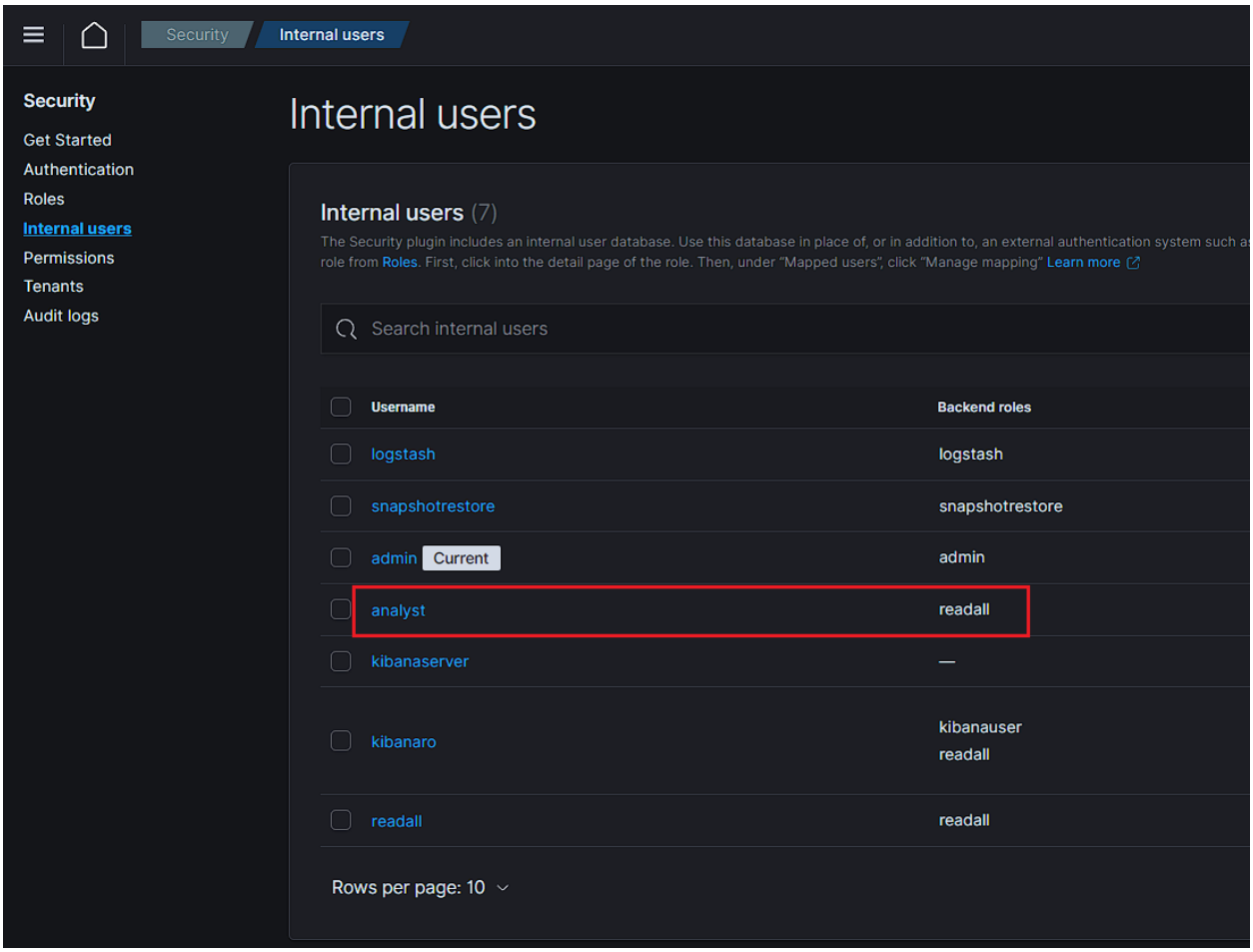


Figure 30 User named analyst created

4.1.1.1.4. Test Case 4

Unit Testing	Objectives
Objective	To test if newly added can login into Wazuh.
Action	Trying to login into Wazuh from analyst user.
Expected Result	User should be able to login.
Actual Result	User was able to login successfully.
Conclusion	Test successful.

Table 6 Unit Testing - Test Case 4

Evidence

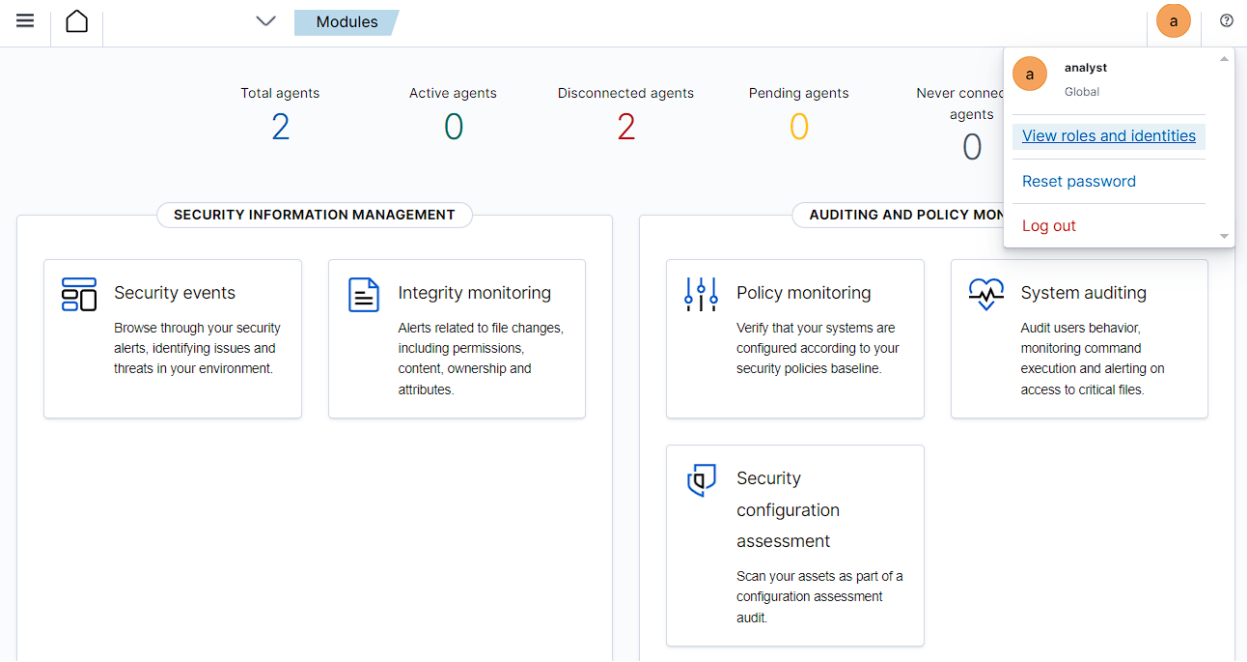


Figure 31 Logged in as analyst in Wazuh

4.1.1.1.5. Test Case 5

Unit Testing	Objectives
Objective	To test if admin can modify user's role in Wazuh.
Action	Map all_read role to user analyst.
Expected Result	Role should be mapped to user analyst.
Actual Result	Role was mapped to user analyst.
Conclusion	Test successful.

Table 7 Unit Testing - Test Case 5

Evidence

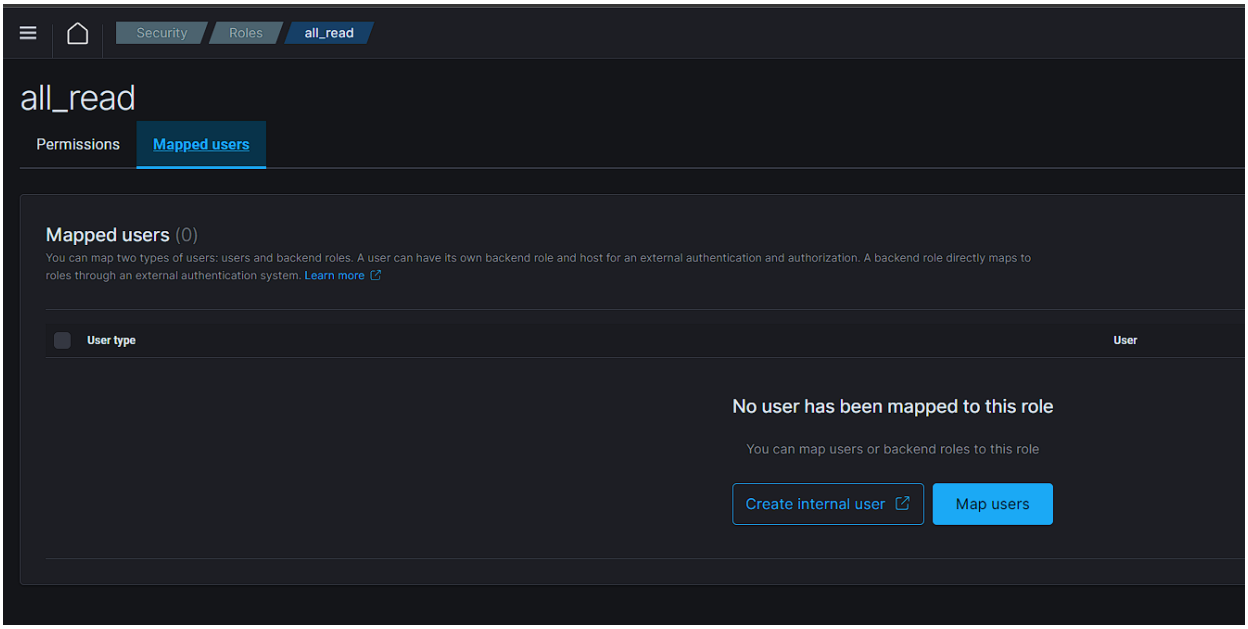


Figure 32 Before mapping all_read role to user analyst

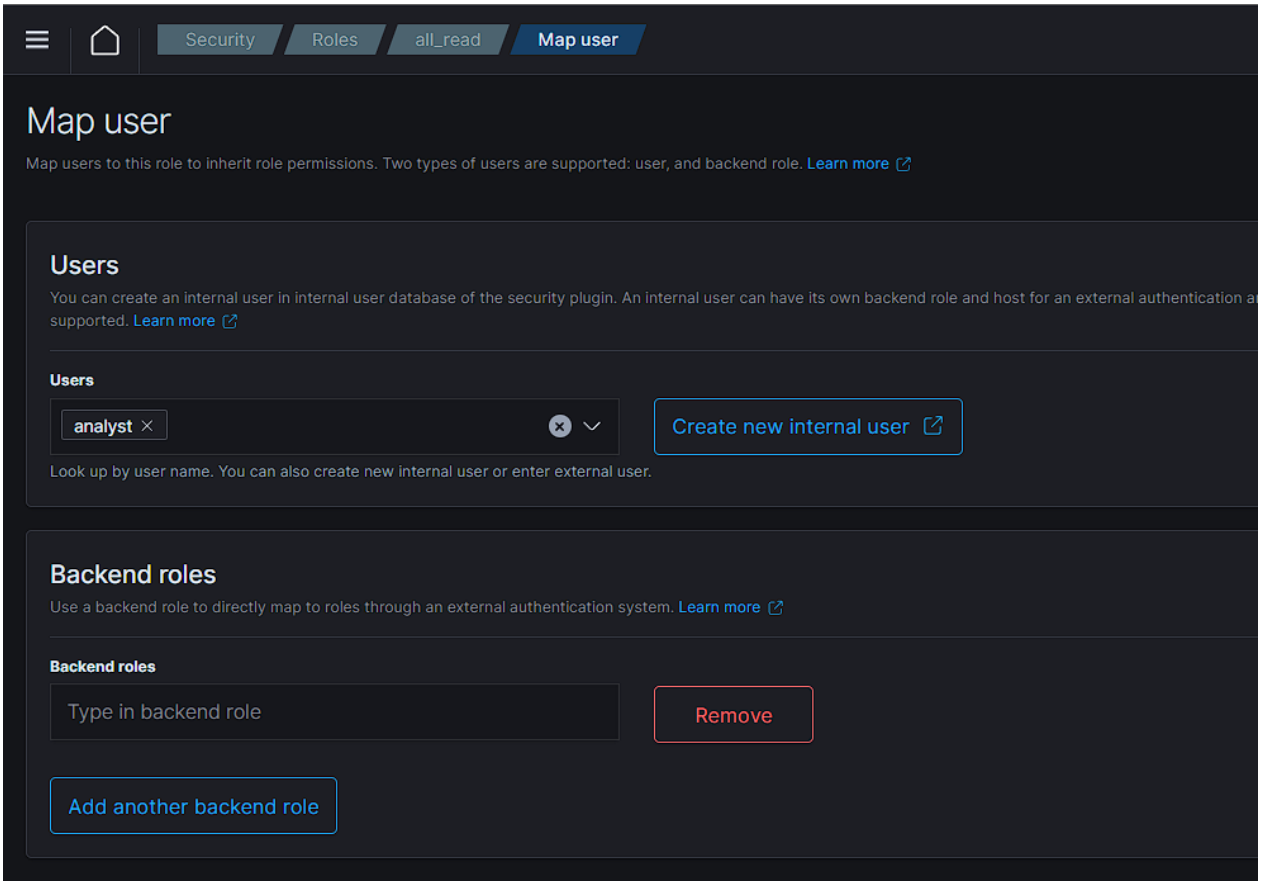


Figure 33 Mapping all_read role to user analyst

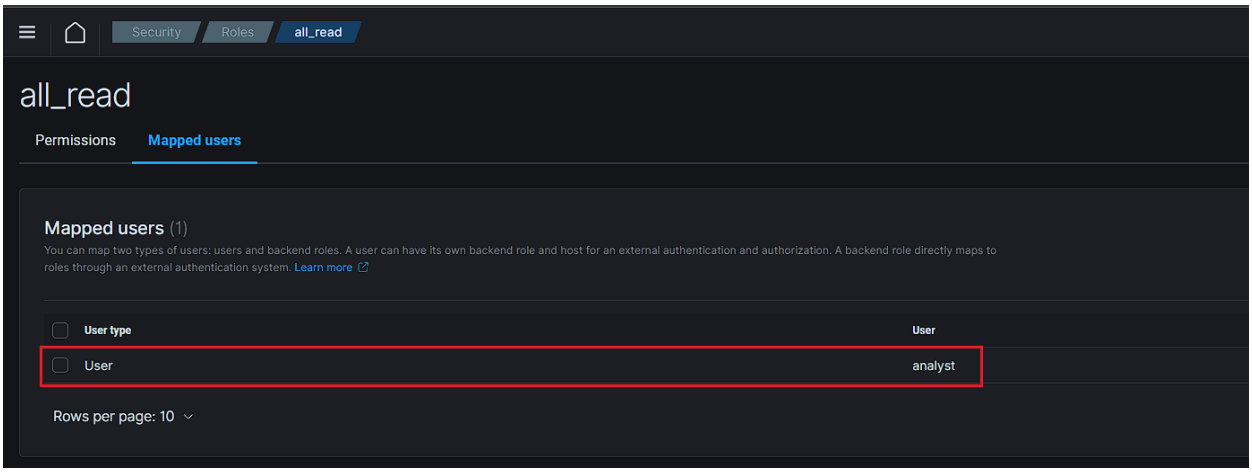


Figure 34 After mapping all_read role to user analyst

4.1.1.1.6. Test Case 6

Unit Testing	Objectives
Objective	To test if admin can delete users in Wazuh.
Action	Remove user analyst from internal users.
Expected Result	User analyst should be removed.
Actual Result	User analyst was removed.
Conclusion	Test successful.

Table 8 Unit Testing - Test Case 6

Evidence

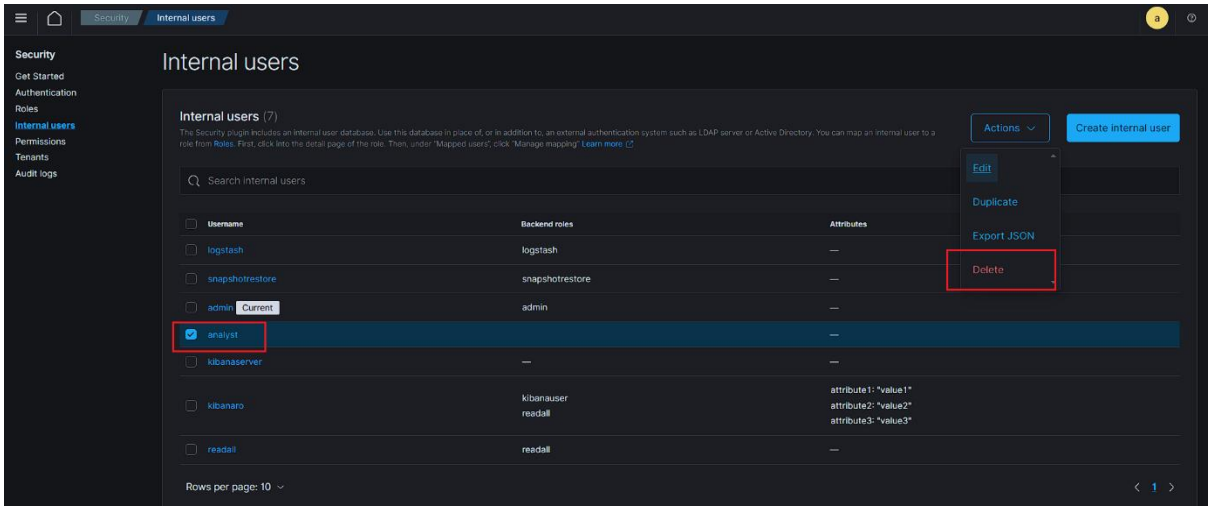


Figure 35 Before deleting user analyst

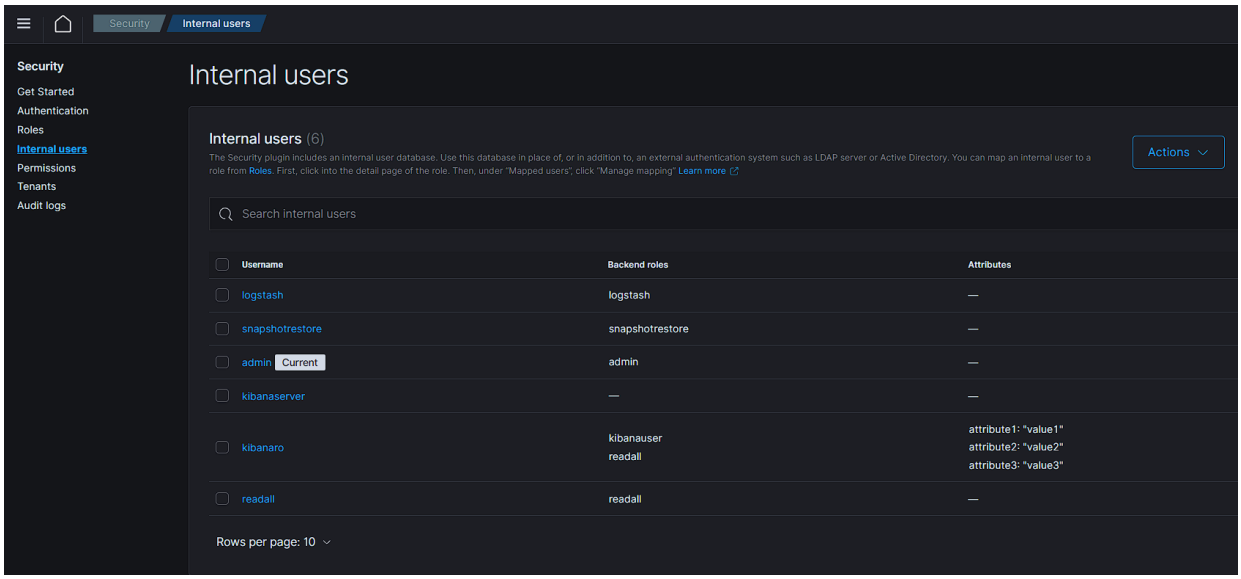


Figure 36 After deleting user analyst

4.1.1.1.7. Test Case 7

Unit Testing	Objectives
Objective	To test if new user can be added in MISP.
Action	Add user hunter from Administration settings.
Expected Result	User hunter should be added.
Actual Result	User hunter was added.
Conclusion	Test successful.

Table 9 Unit Testing - Test Case 1

Evidence

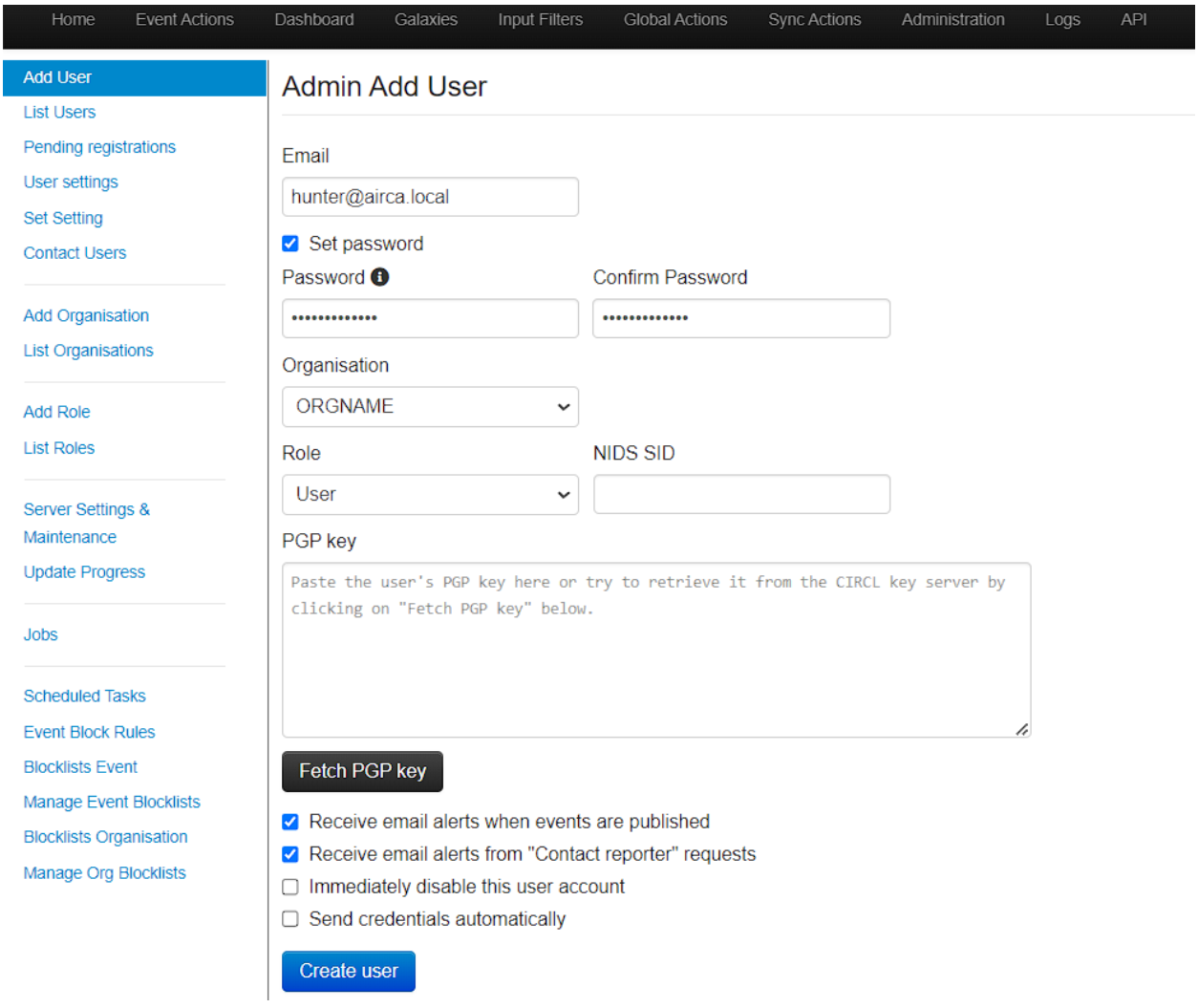


Figure 37 Creating user hunter

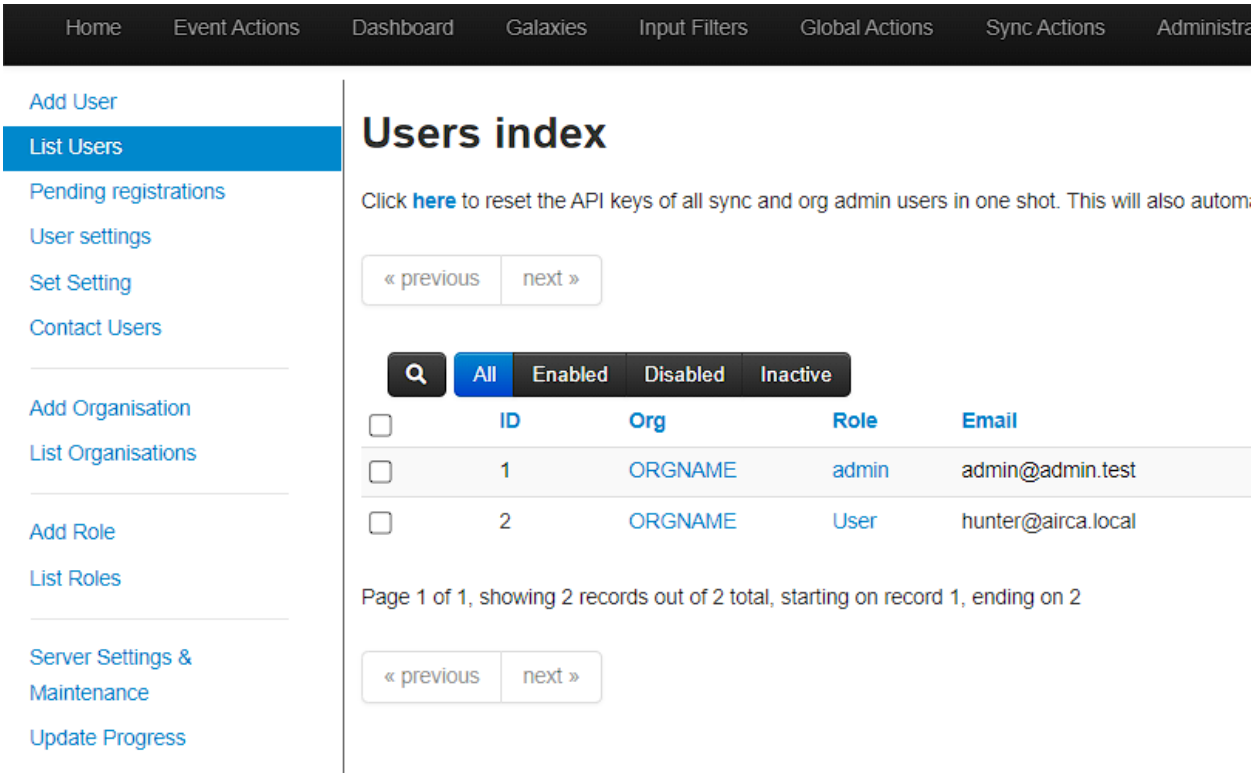


Figure 38 User hunter created

4.1.1.1.8. Test Case 8

Unit Testing	Objectives
Objective	To test if newly added can login into MISP.
Action	Trying to login from user hunter.
Expected Result	User hunter should be able to login.
Actual Result	User hunter was able to login.
Conclusion	Test successful.

Table 10 Unit Testing - Test Case 1

Evidence

Figure 39 Logging into user hunter

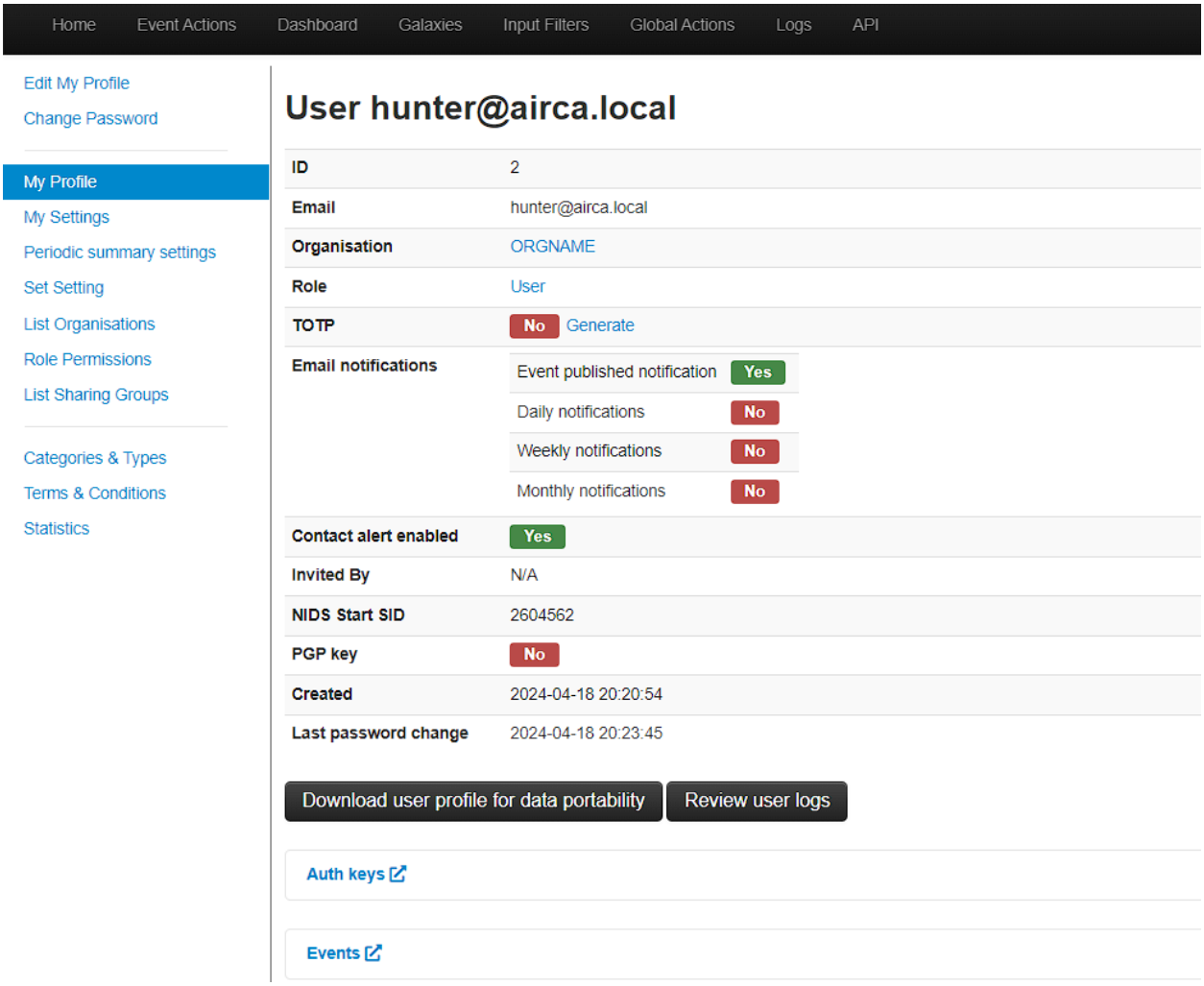


Figure 40 Logged into user hunter

4.1.1.1.9. Test Case 9

Unit Testing	Objectives
Objective	To test if admin can modify user's role in MISP.
Action	Assigning Org Admin role to user hunter.
Expected Result	User hunter should be Org Admin.
Actual Result	User hunter got Org Admin role successfully.
Conclusion	Test successful.

Table 11 Unit Testing - Test Case 1

Evidence

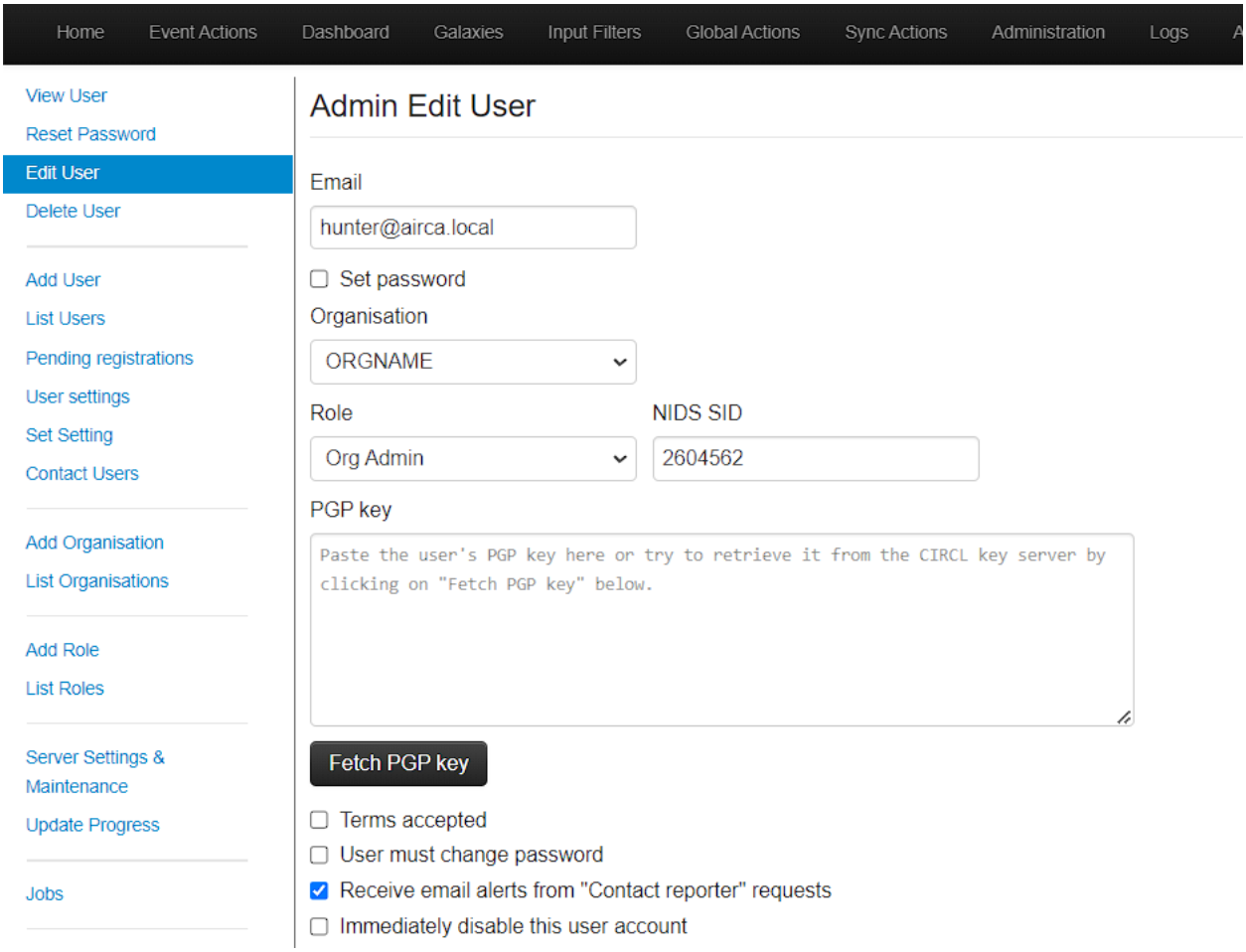


Figure 41 Before making user hunter Org Admin

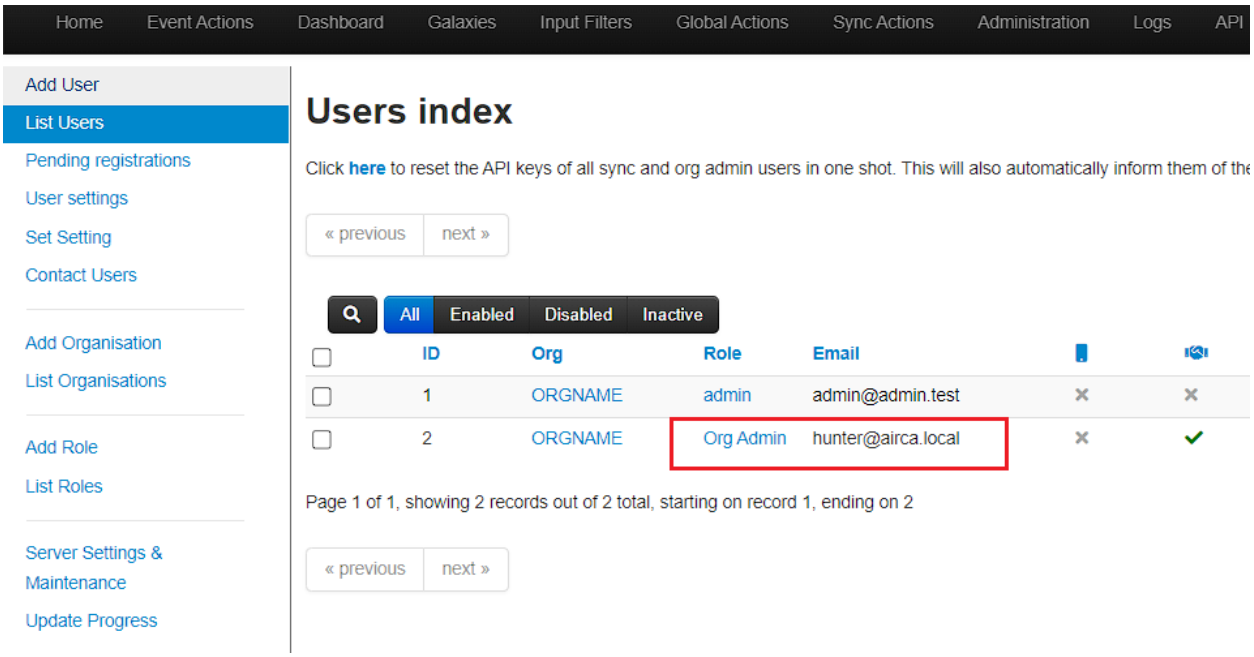


Figure 42 After making user hunter Org Admin

4.1.1.1.10. Test Case 10

Unit Testing	Objectives
Objective	To test if admin can delete users in MISP.
Action	Select and hit remove icon to user hunter.
Expected Result	User hunter should be removed.
Actual Result	User hunter was removed.
Conclusion	Test successful.

Table 12 Unit Testing - Test Case 1

Evidence

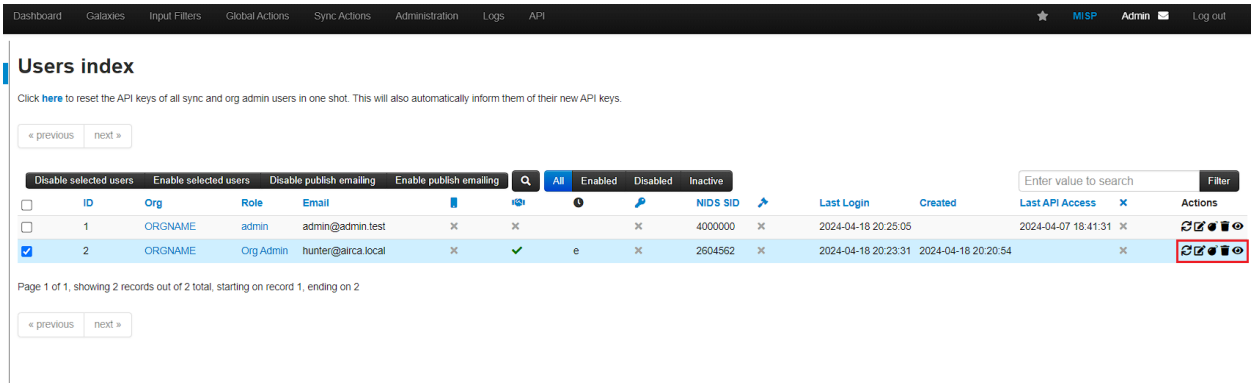


Figure 43 Before removing user hunter

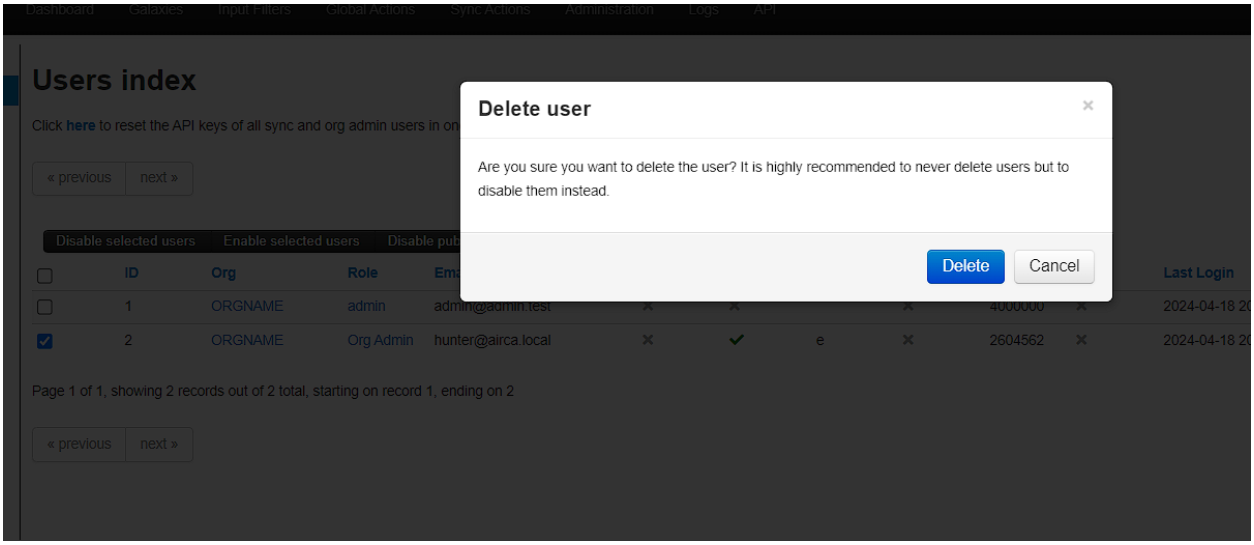


Figure 44 User Delete Confirmation Prompt

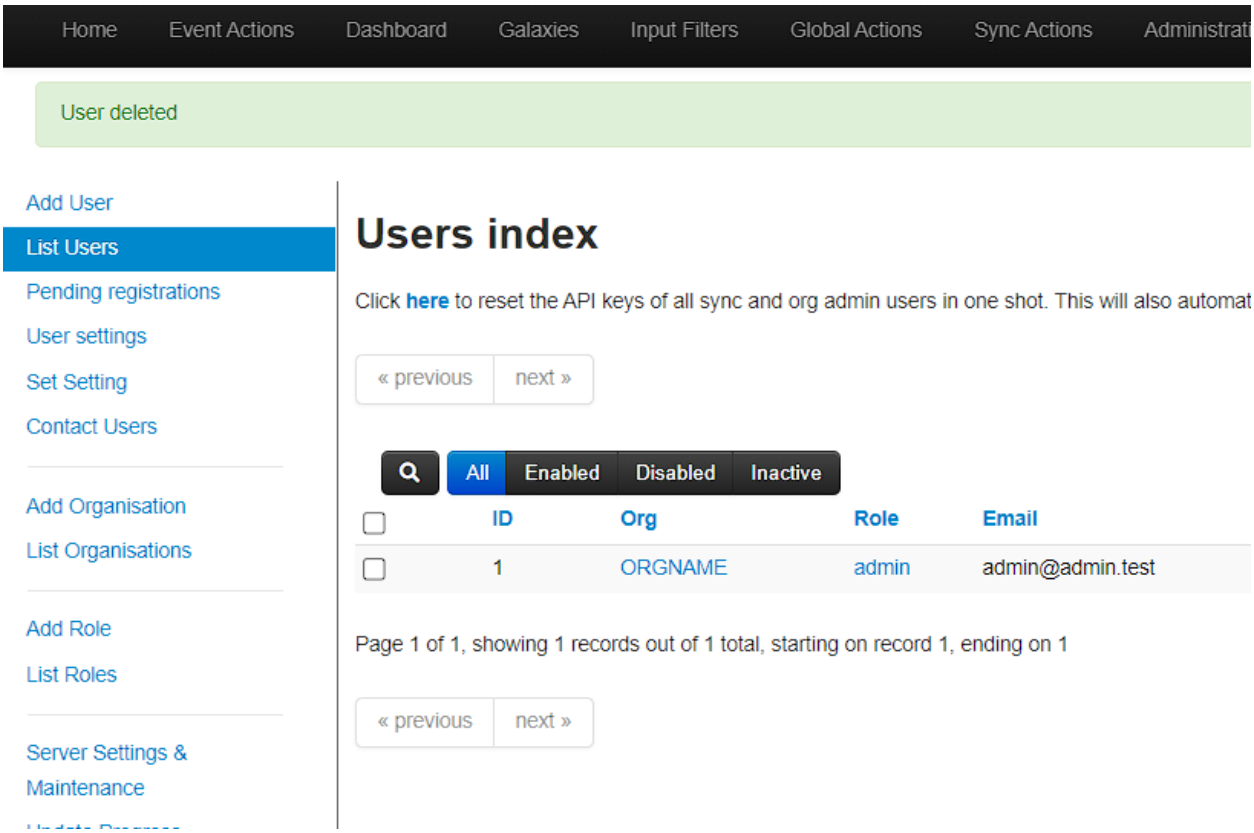


Figure 45 After deleting user hunter

4.1.2. System Testing

System testing involves assessing a fully integrated system to validate its functionality and performance (Testsigma Inc., 2023). It aims to evaluate the system. In this project, ten distinct system tests were conducted, and any associated problems or issues were addressed.

4.1.2.1. Test Plan

Test Cases	Objectives	Results
Case 1	To check if windows and ubuntu agent is working.	Successful
Case 2	To check if logs collection and visualization is working for windows and ubuntu endpoint.	Successful
Case 3	To check if Sysmon log collection is working in windows endpoint.	Successful
Case 4	To check if MISP API is working.	Successful
Case 5	To check if MISP integration with Wazuh is working.	Successful
Case 6	To check if browsing suspicious domain gets detected.	Failed
Case 7	To check if file integrity monitoring is working.	Successful
Case 8	To check if Yara Analysis scan detects suspicious files.	Successful
Case 9	To check if suspicious file gets quarantined.	Successful
Case 10	To check if default plugins in dashboard can be removed.	Failed

Table 13 Test Plans for System Testing

4.1.2.1.1. Test Case 1

System Testing	Objectives
Objective	To check if windows and ubuntu agent is working.
Action	Checked status in Wazuh agent application and in Wazuh dashboard.
Expected Result	Agent should be active and running.
Actual Result	Agents were active and running.
Conclusion	Test successful.

Table 14 System Testing - Test Case 1

Evidence

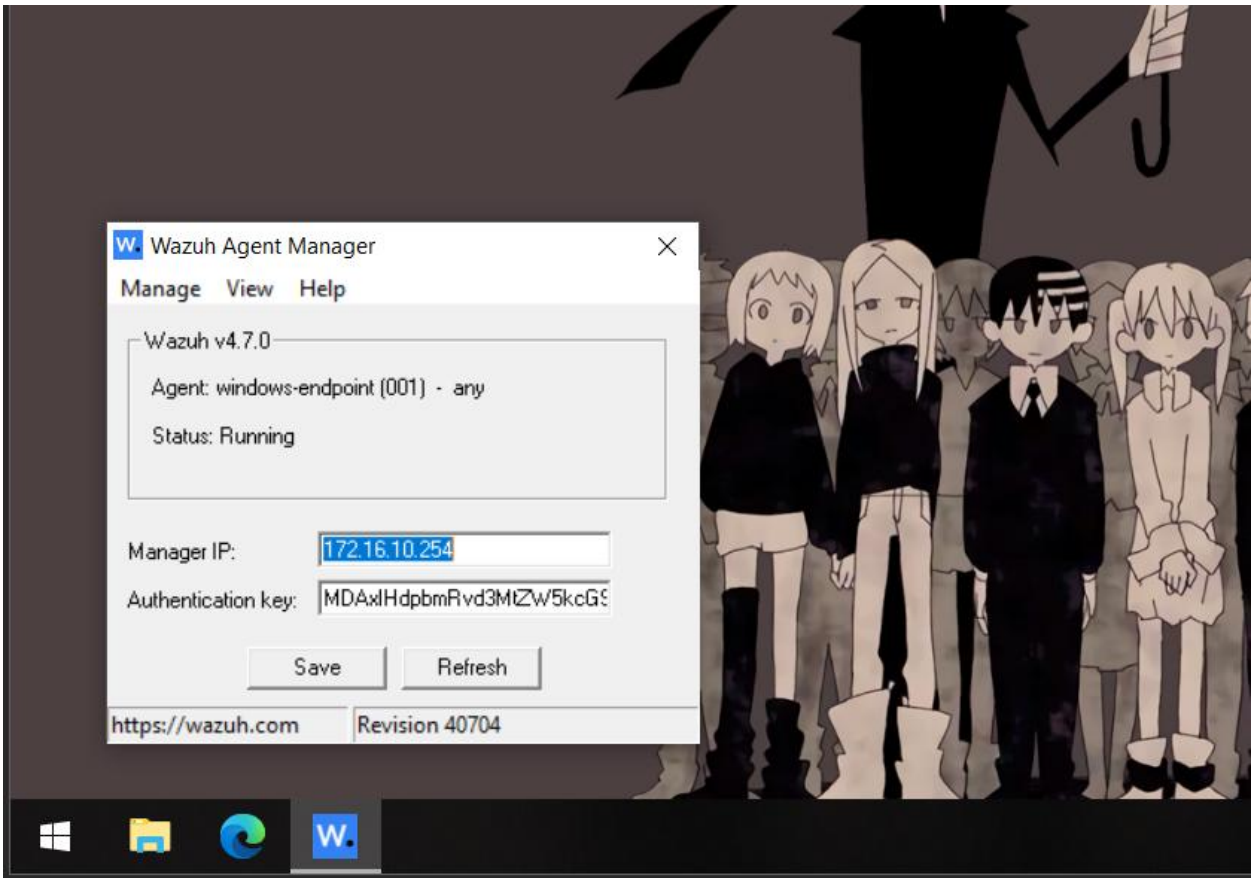


Figure 46 Checking agent status in windows endpoint

```
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ sudo systemctl status wazuh-agent
[sudo] password for ubuntu:
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-04-22 23:26:16 +0545; 1min 35s ago
    Process: 921 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 32 (limit: 2217)
     Memory: 306.8M
        CPU: 25.978s
     CGroup: /system.slice/wazuh-agent.service
             ├─1327 /var/ossec/bin/wazuh-execd
             ├─1435 /var/ossec/bin/wazuh-agentd
             ├─1488 /var/ossec/bin/wazuh-syscheckd
             ├─1588 /var/ossec/bin/wazuh-logcollector
             └─1838 /var/ossec/bin/wazuh-modulesd

Apr 22 23:26:07 ubuntu env[921]: Deleting PID file '/var/ossec/var/run/wazuh-agentd-1490.pid' not used...
Apr 22 23:26:07 ubuntu env[921]: Deleting PID file '/var/ossec/var/run/wazuh-execd-1391.pid' not used...
Apr 22 23:26:09 ubuntu env[921]: Started wazuh-execd...
Apr 22 23:26:10 ubuntu env[921]: Started wazuh-agentd...
Apr 22 23:26:11 ubuntu env[921]: Started wazuh-syscheckd...
Apr 22 23:26:13 ubuntu env[921]: Started wazuh-logcollector...
Apr 22 23:26:13 ubuntu env[1822]: 2024/04/22 23:26:13 wazuh-modulesd: WARNING: The 'hotfixes' option is only avail
Apr 22 23:26:14 ubuntu env[921]: Started wazuh-modulesd...
Apr 22 23:26:16 ubuntu env[921]: Completed.
Apr 22 23:26:16 ubuntu systemd[1]: Started Wazuh agent.
```

Figure 47 Checking agent status in ubuntu endpoint

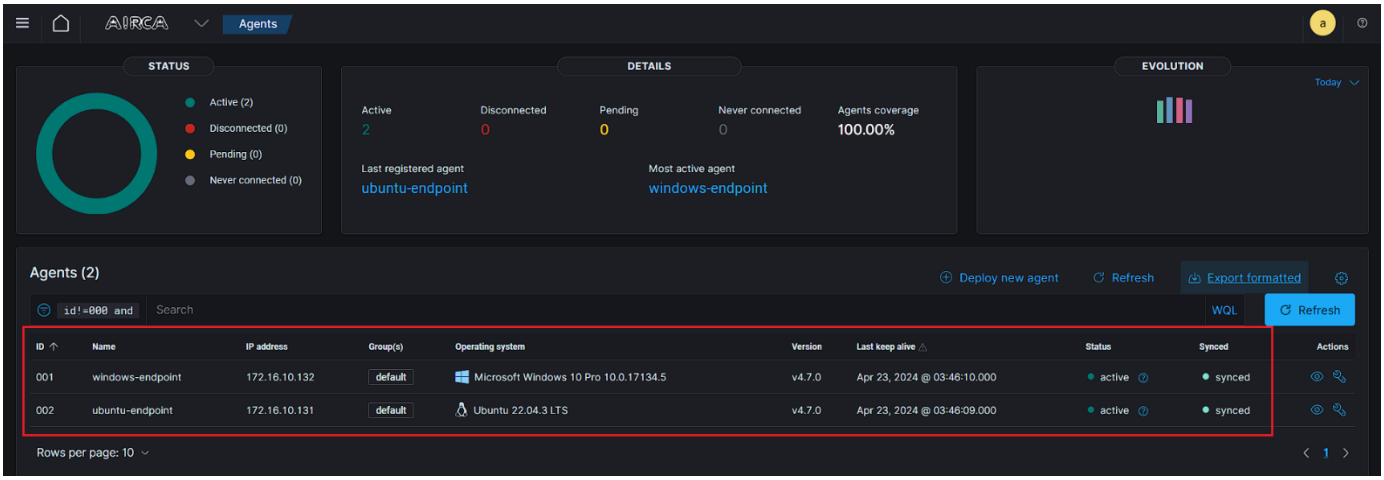


Figure 48 Checking agents' status in Wazuh Dashboard

4.1.2.1.2. Test Case 2

System Testing	Objectives
Objective	To check if logs collection and visualization is working for windows and ubuntu endpoint.
Action	Checking security event logs in Wazuh.
Expected Result	Event logs should be seen.
Actual Result	Even logs were seen.
Conclusion	Test successful.

Table 15 System Testing - Test Case 2

Evidence

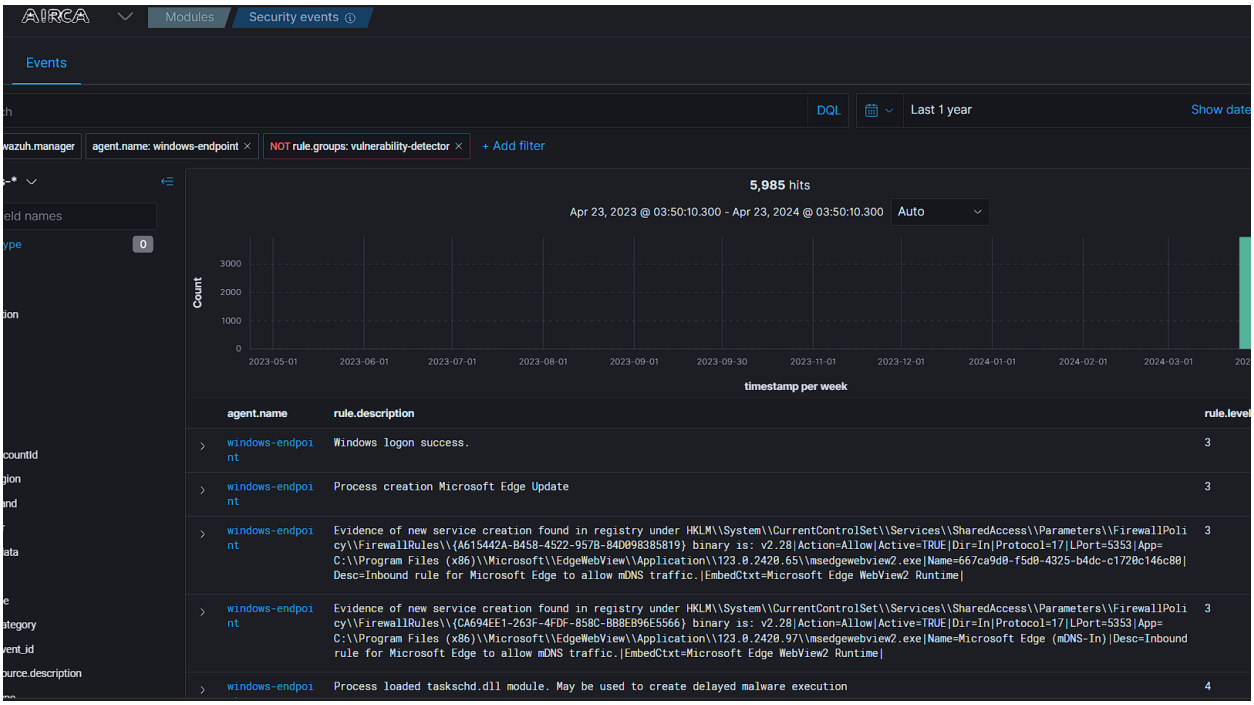


Figure 49 Filtering windows endpoint security logs in Wazuh

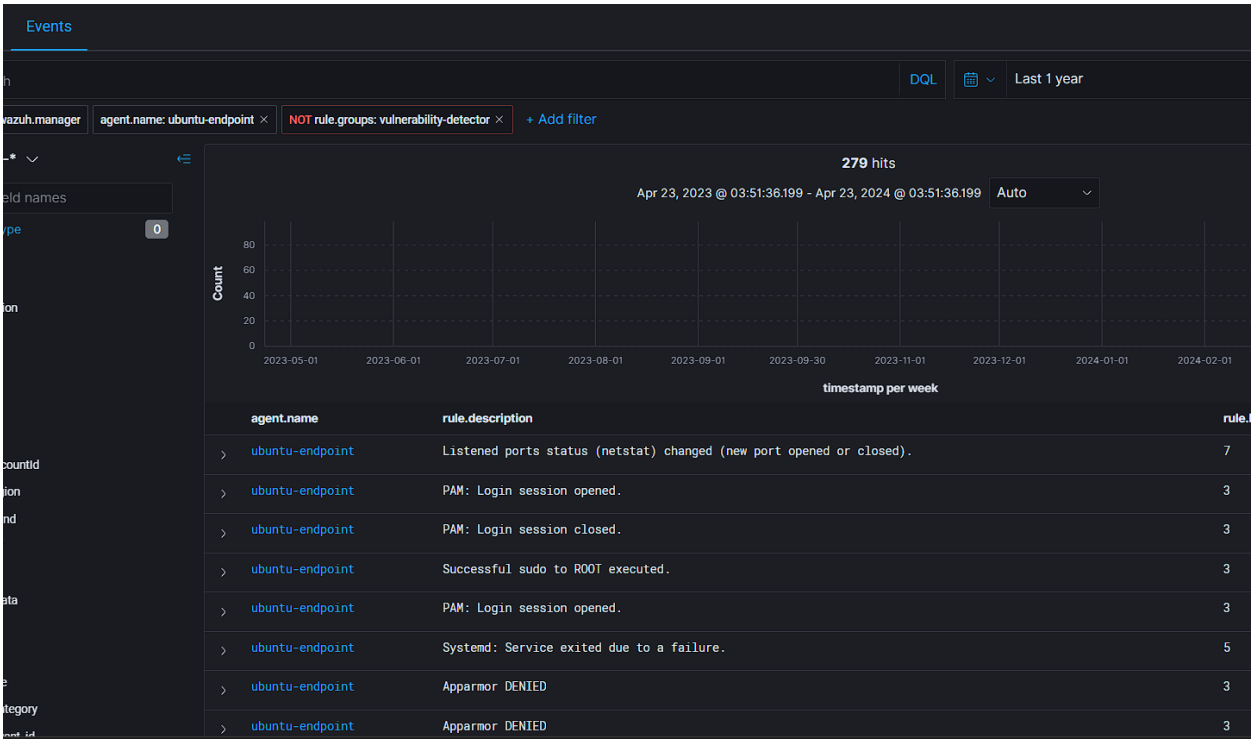


Figure 50 Filtering ubuntu endpoint security logs in Wazuh

4.1.2.1.3. Test Case 3

System Testing	Objectives
Objective	To check if Sysmon log collection is working in windows endpoint.
Action	Filtering only Sysmon logs in Wazuh.
Expected Result	Sysmon logs should be seen.
Actual Result	Sysmon logs were seen.
Conclusion	Test successful.

Table 16 System Testing - Test Case 3

Evidence

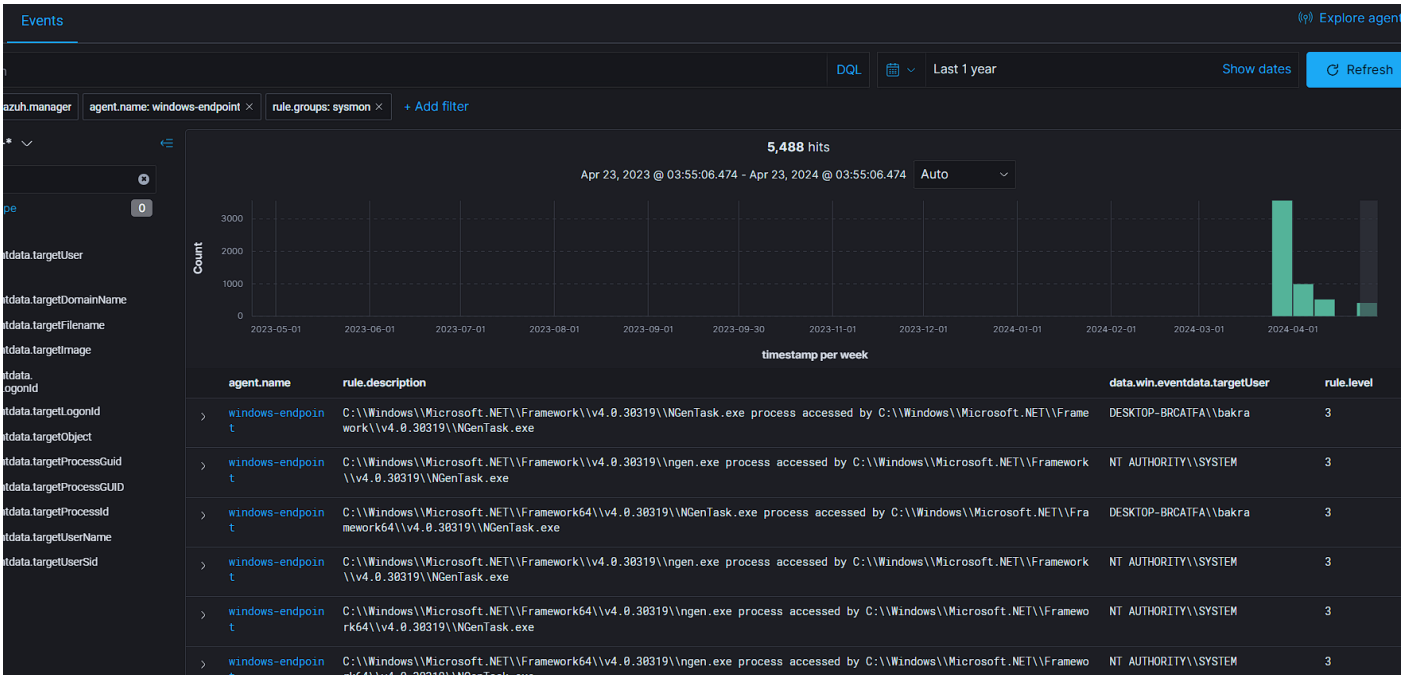


Figure 51 Filtering overall Sysmon event logs

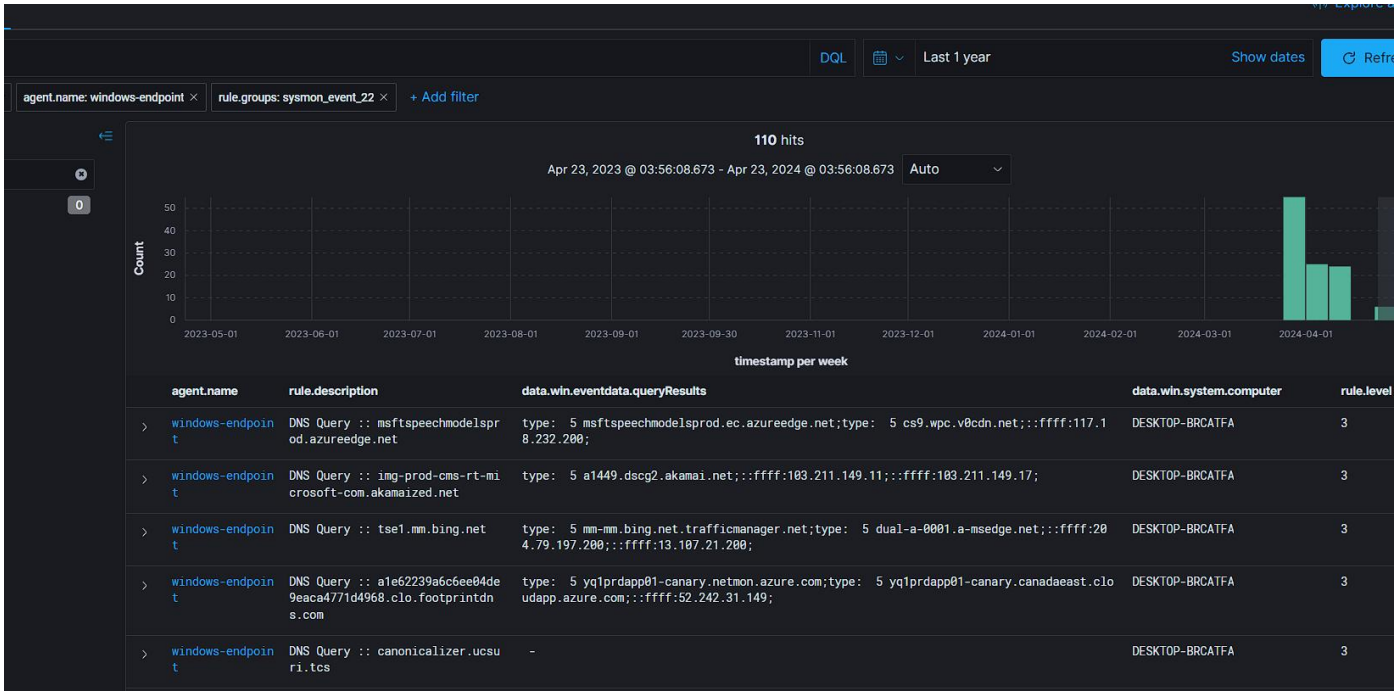


Figure 52 Filtering DNS query through Sysmon events

4.1.2.1.4. Test Case 4

System Testing	Objectives
Objective	To check if MISP API is working.
Action	Ran a custom script file to query IoC to MISP API.
Expected Result	IoC hit result should be fetched.
Actual Result	IoC hit result were fetched.
Conclusion	Test successful.

Table 17 System Testing - Test Case 4

Evidence

Figure 53 Using custom script file to request domain IoC to MISP API

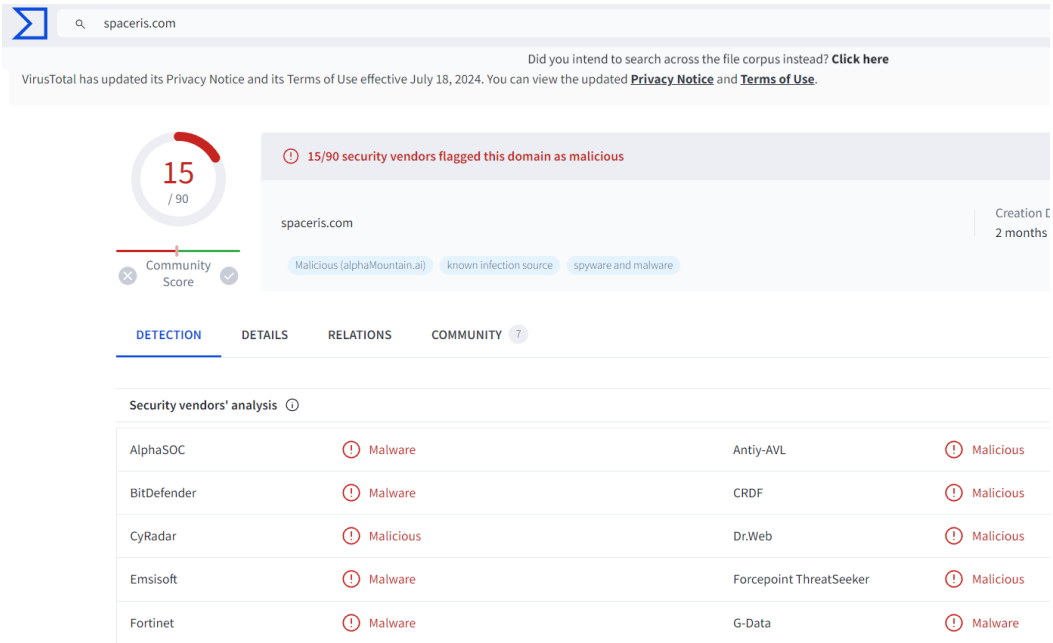


Figure 54 Scanning Domain IoC in VirusTotal

Figure 55 Using custom script file to request file hash IoC to MISP API

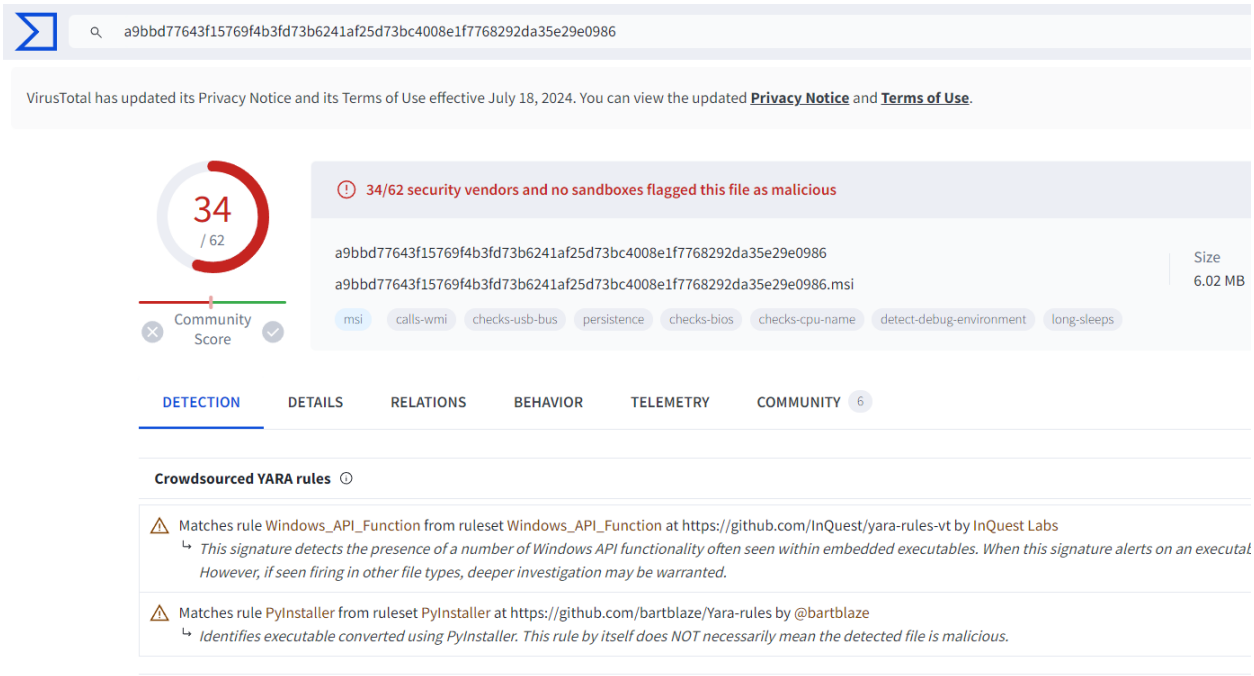


Figure 56 Scanning file hash IoC in VirusTotal

4.1.2.1.5. Test Case 5

System Testing	Objectives
Objective	To check if MISP integration with Wazuh is working.
Action	Pinging a suspicious domain from MISP feed.
Expected Result	IoC match found alert should be generated.
Actual Result	IoC match found alert was generated.
Conclusion	Test successful.

Table 18 System Testing - Test Case 5

Evidence

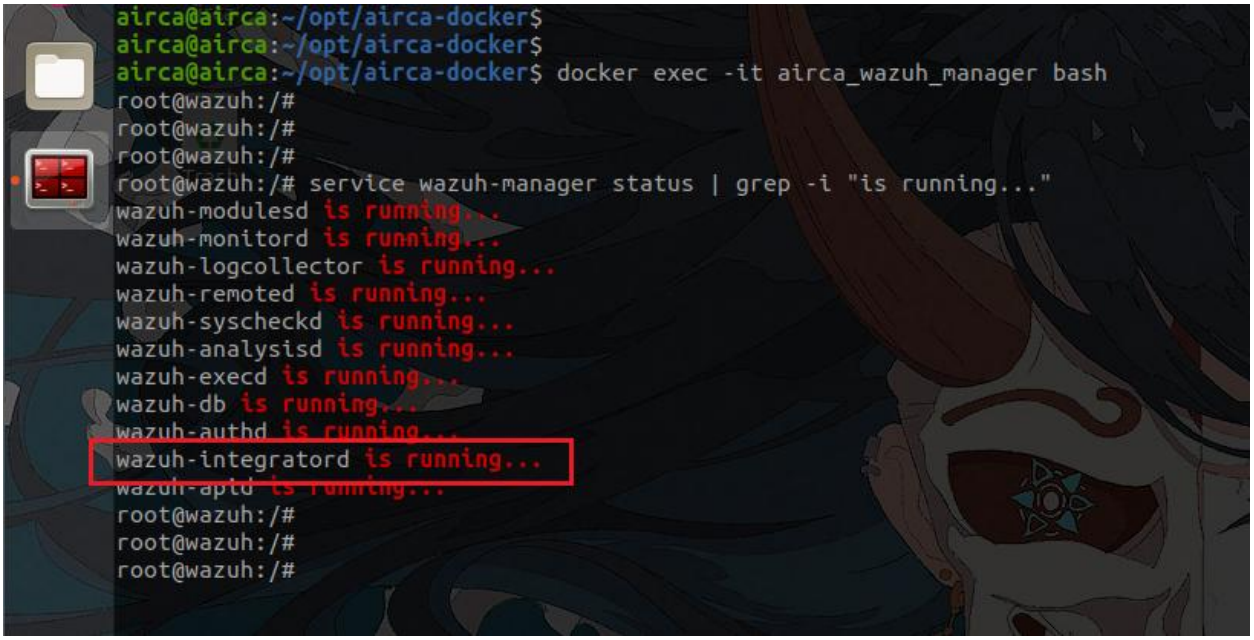


Figure 57 Checking integrator status in Wazuh container

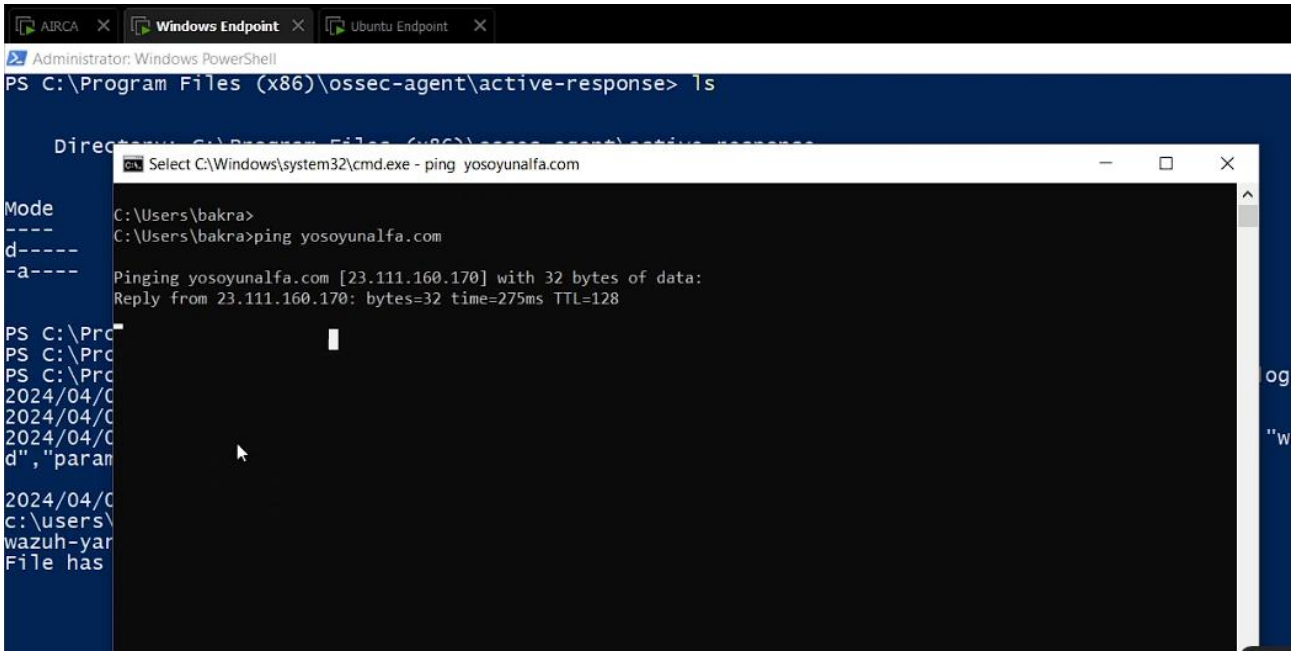


Figure 58 Pinging a suspicious domain from MISP feed

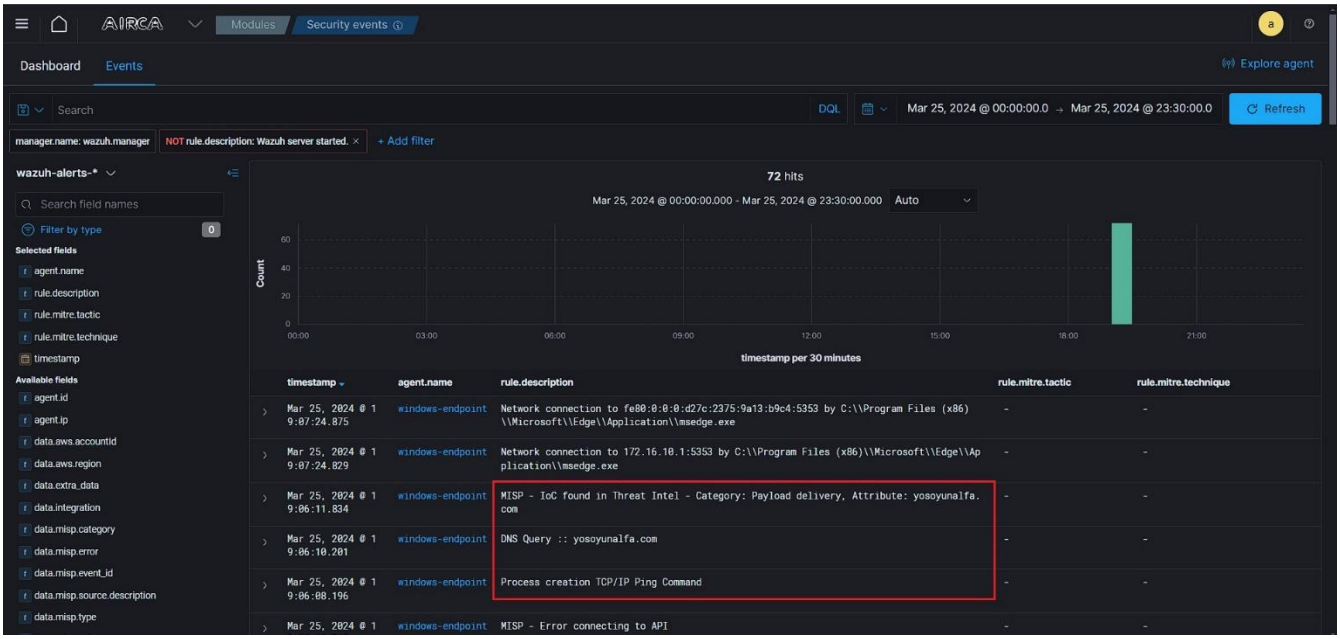


Figure 59 MISP IoC hit found alert in Wazuh

4.1.2.1.6. Test Case 6

System Testing	Objectives
Objective	To check if browsing suspicious domain gets detected.
Action	Browsing suspicious domain from Test Case 5 from MISP feed.
Expected Result	IoC match alert should be generated.
Actual Result	IoC match alert was not generated.
Conclusion	Test Failed.

Table 19 System Testing - Test Case 6

Evidence

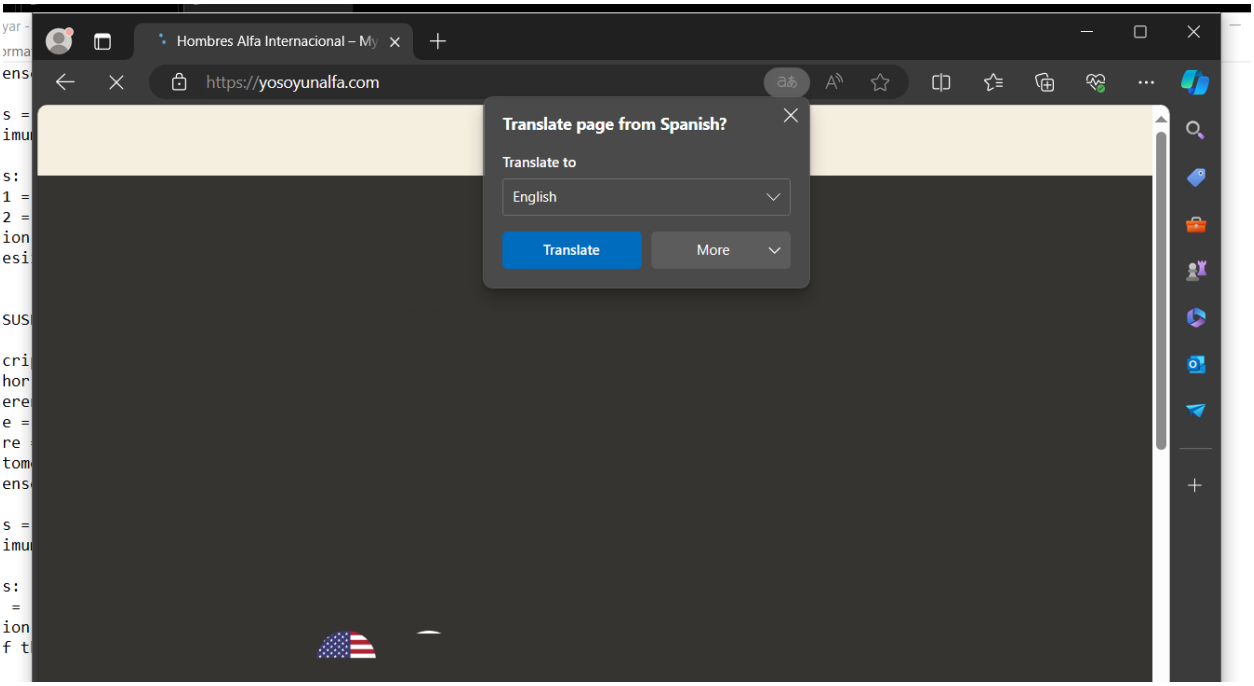


Figure 60 Browsing the suspicious domain in a browser

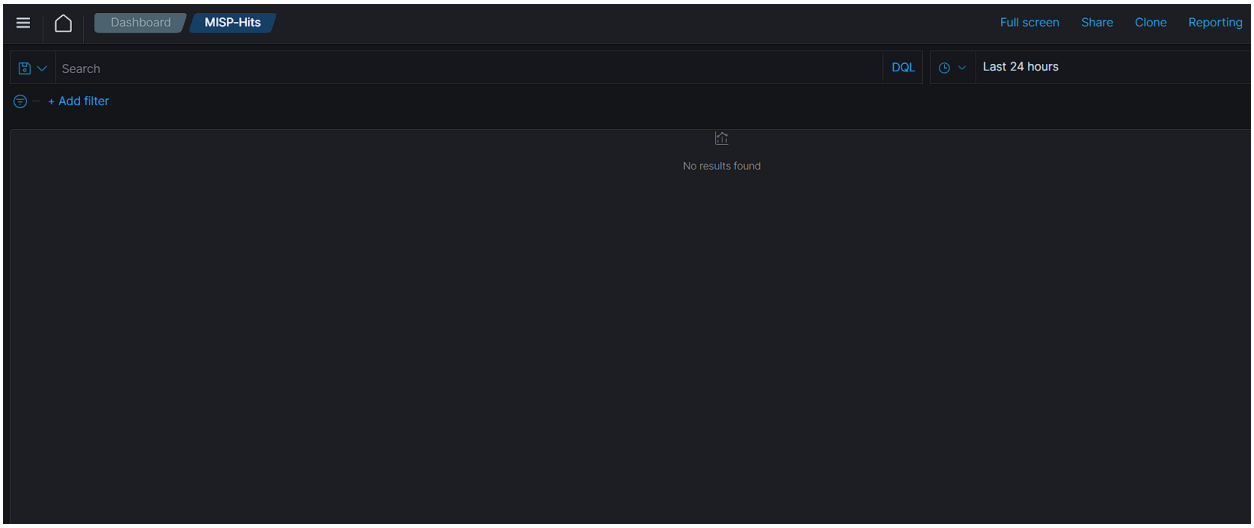


Figure 61 No MISP hits were seen for the suspcious domain after browsing

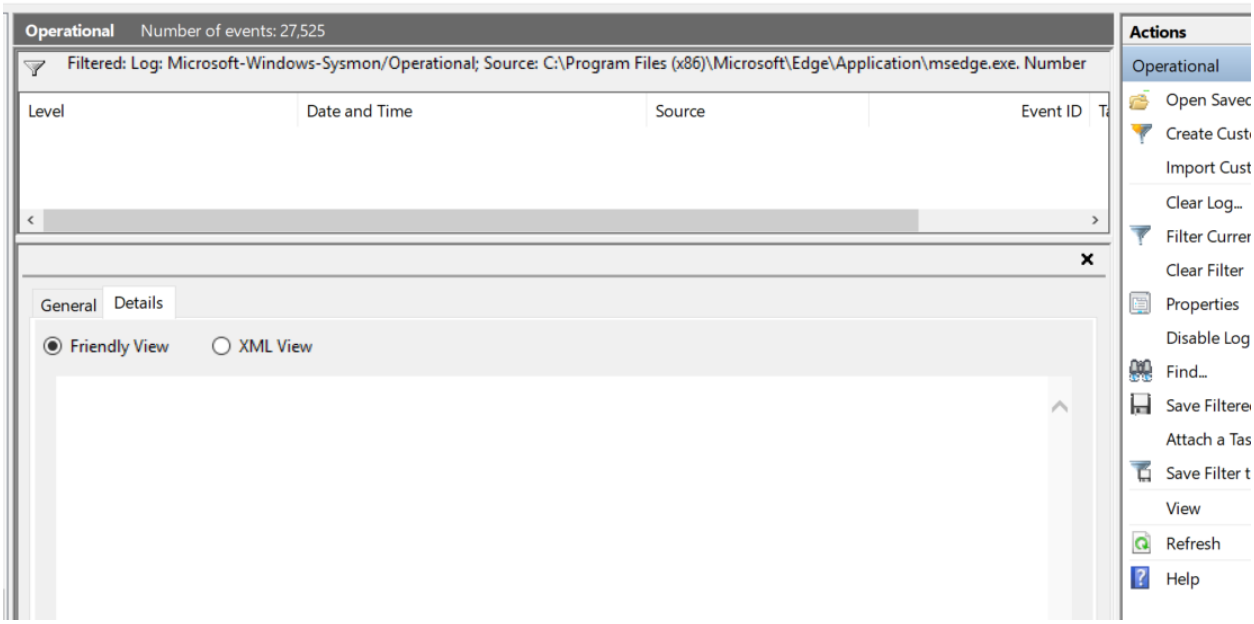


Figure 62 No logs were seen from Microsoft Edge browser process

4.1.2.1.7. Test Case 7

System Testing	Objectives
Objective	To check if file integrity monitoring is working.
Action	Checking logs in FIM dashboard.
Expected Result	Files changes logs in monitored directory should be generated.
Actual Result	Files changes logs in monitored directory were generated.
Conclusion	Test successful.

Table 20 System Testing - Test Case 7

Evidence

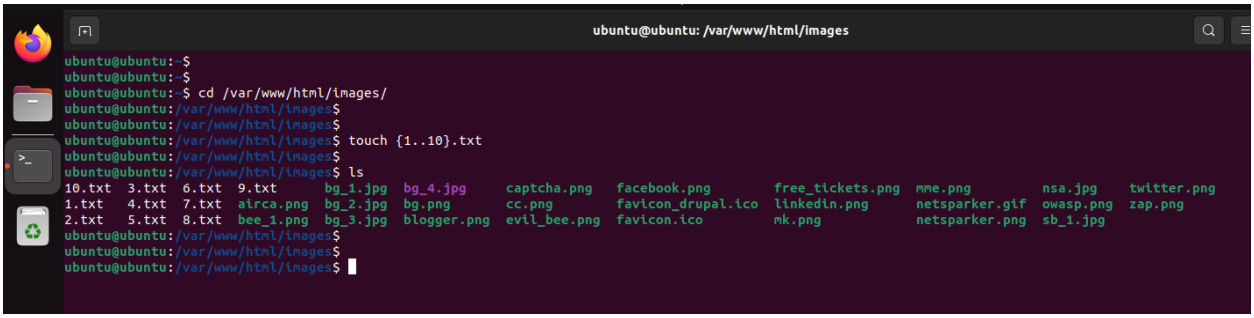
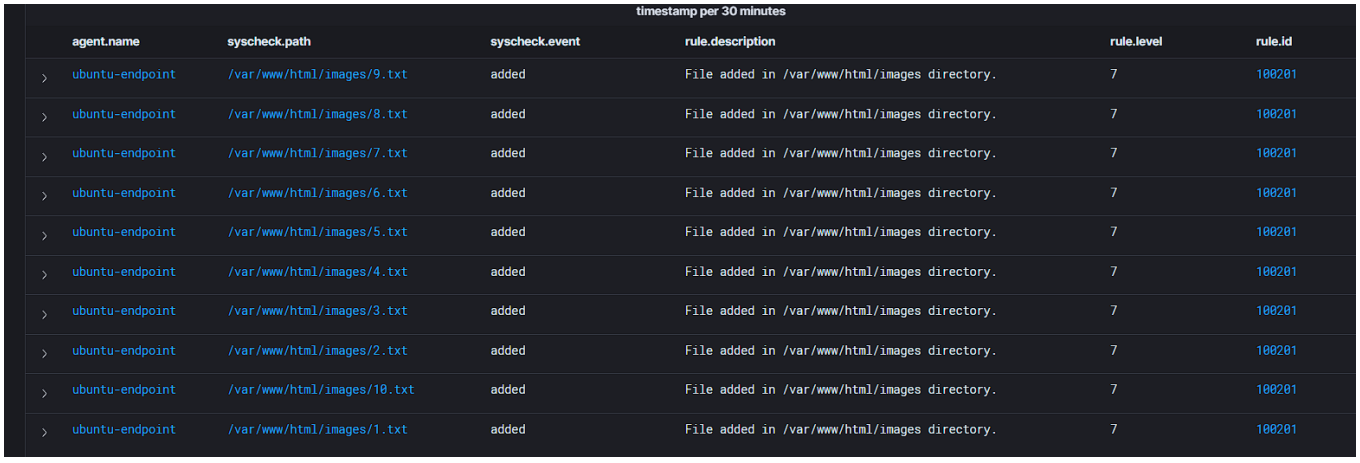


Figure 63 Creating files in ubuntu endpoint's monitored directory



agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
ubuntu-endpoint	/var/www/html/images/9.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/8.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/7.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/6.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/5.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/4.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/3.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/2.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/10.txt	added	File added in /var/www/html/images directory.	7	100201
ubuntu-endpoint	/var/www/html/images/1.txt	added	File added in /var/www/html/images directory.	7	100201

Figure 64 Filtering FIM events for ubuntu endpoint

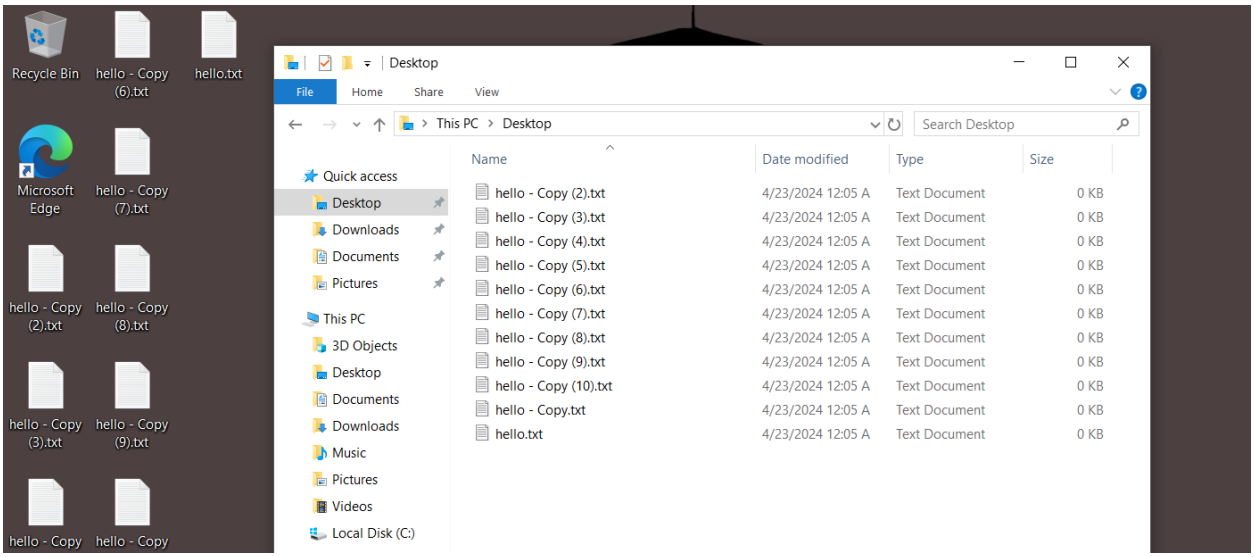


Figure 65 Creating files in windows endpoint's monitored directory

timestamp per 30 minutes

agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
windows-endpoint	c:\users\bakra\desktop\hello - copy (9).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (8).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (7).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (6).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (5).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (4).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (3).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (2).txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy.txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello.txt	added	FileCreated ::	3	666666
windows-endpoint	c:\users\bakra\desktop\hello - copy (10).txt	added	FileCreated ::	3	666666

Figure 66 Filtering FIM events for windows endpoint

4.1.2.1.8. Test Case 8

System Testing	Objectives
Objective	To check if Yara Analysis scan detects suspicious files.
Action	Uploaded Eicar and Wannacry sample file in monitored directory.
Expected Result	Yara Scan Positive Rule hit alert should be generated.
Actual Result	Yara Scan Positive Rule hit alert was generated.
Conclusion	Test successful.

Table 21 System Testing - Test Case 8

Evidence

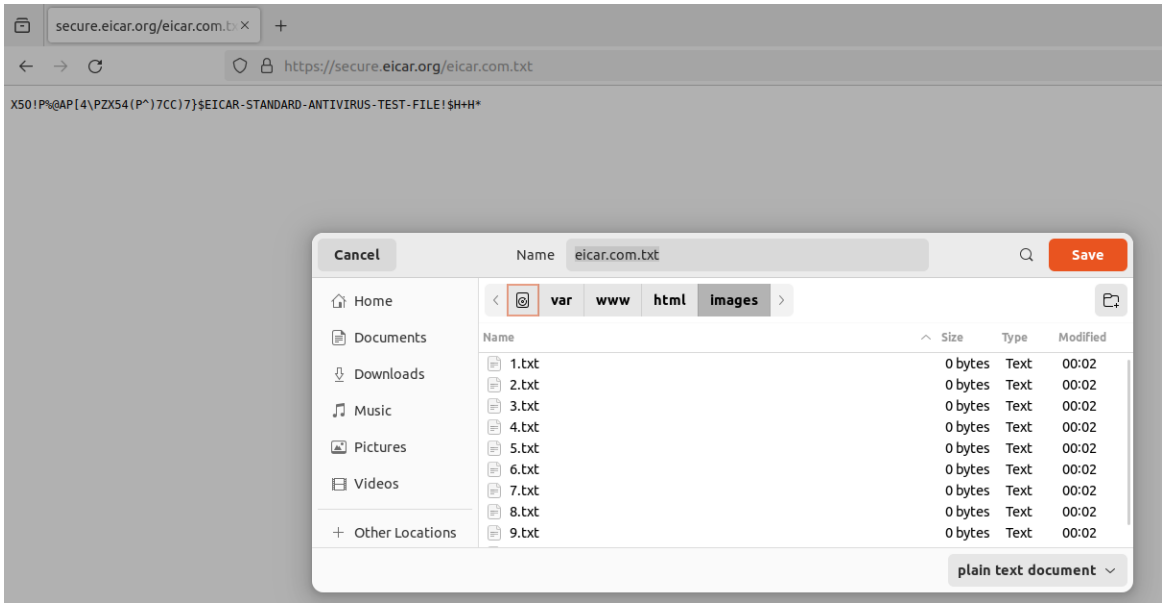


Figure 67 Downloading Eicar file to ubuntu endpoint's monitored directory

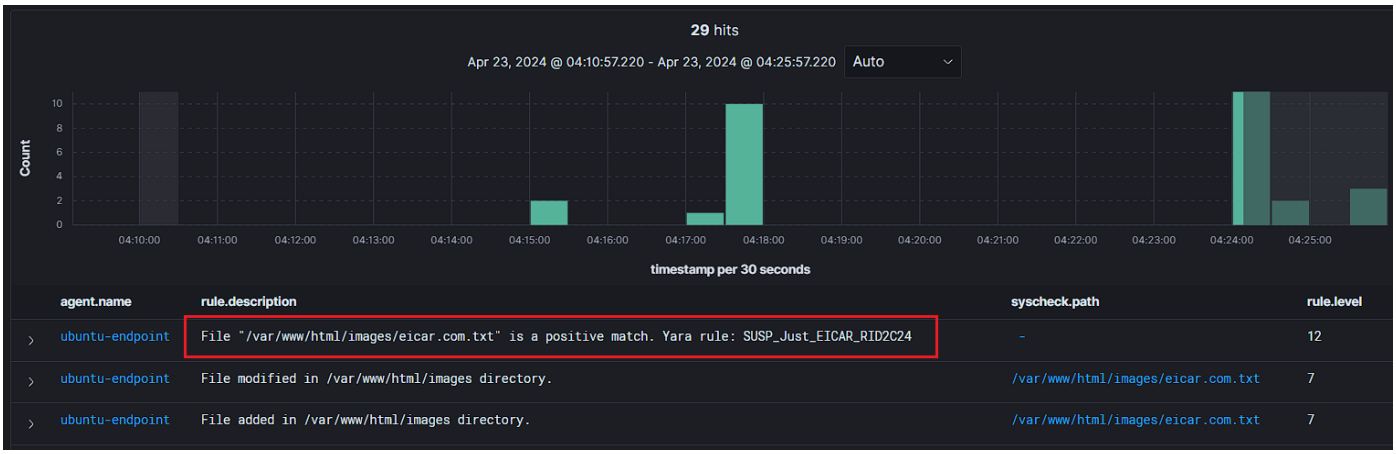


Figure 68 Yara analysis positive hit alert for Eicar file

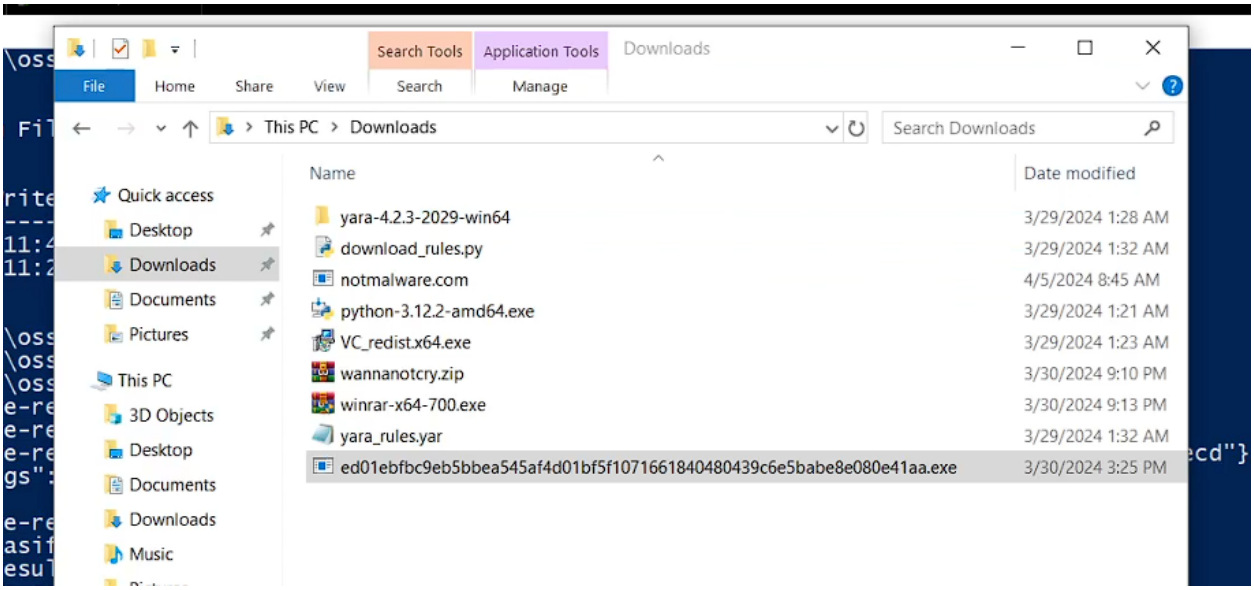


Figure 69 Adding WannaCry sample in window endpoint's monitored directory

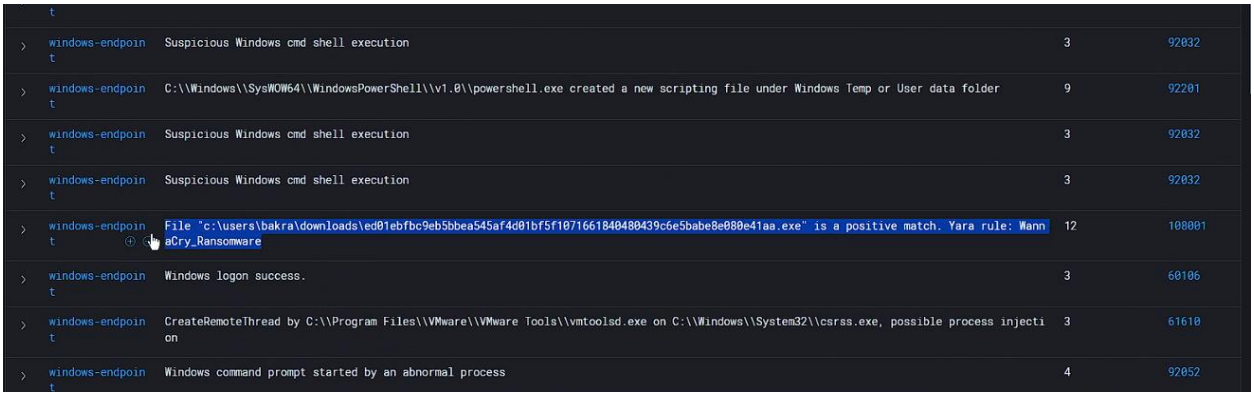


Figure 70 Yara analysis positive hit alert for WannaCry sample

4.1.2.1.9. Test Case 9

System Testing	Objectives
Objective	To check if suspicious file gets quarantined.
Action	Uploaded WannaCry sample in monitored directory.
Expected Result	WannaCry sample should get quarantined.
Actual Result	WannaCry sample was quarantined.
Conclusion	Test successful.

Table 22 System Testing - Test Case 9

Evidence

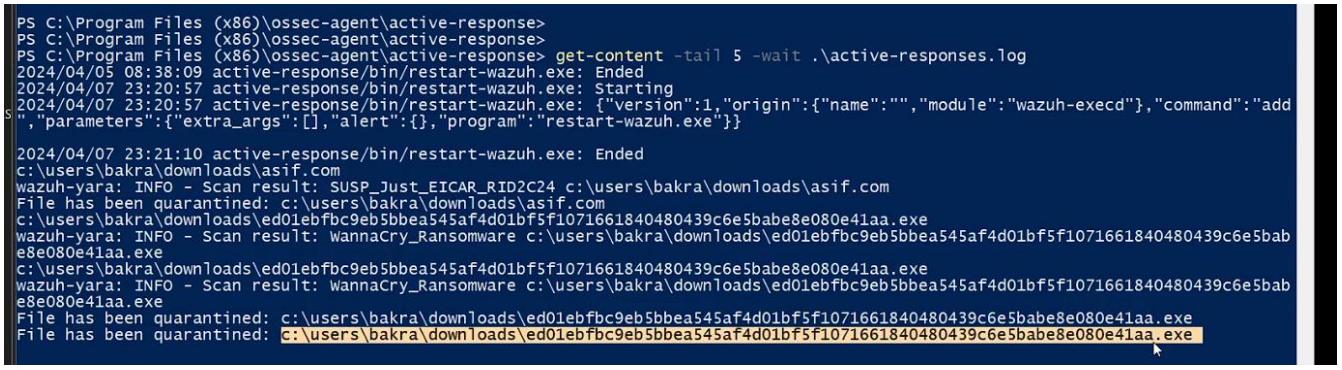


Figure 71 Active Response log of WannaCry sample getting quarantined

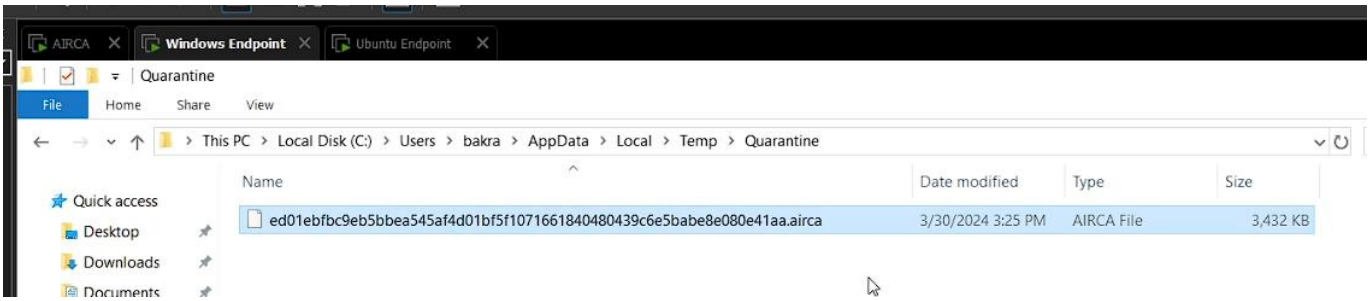


Figure 72 Verifying that WannaCry sample was quarantined in temp folder

4.1.2.1.10. Test Case 10

System Testing	Objectives
Objective	To check if default plugins in Wazuh dashboard can be removed.
Action	Uninstalled plugins from docker container.
Expected Result	Default dashboard should be changed.
Actual Result	Dashboard could not be loaded.
Conclusion	Test failed.

Table 23 System Testing - Test Case 10

Evidence

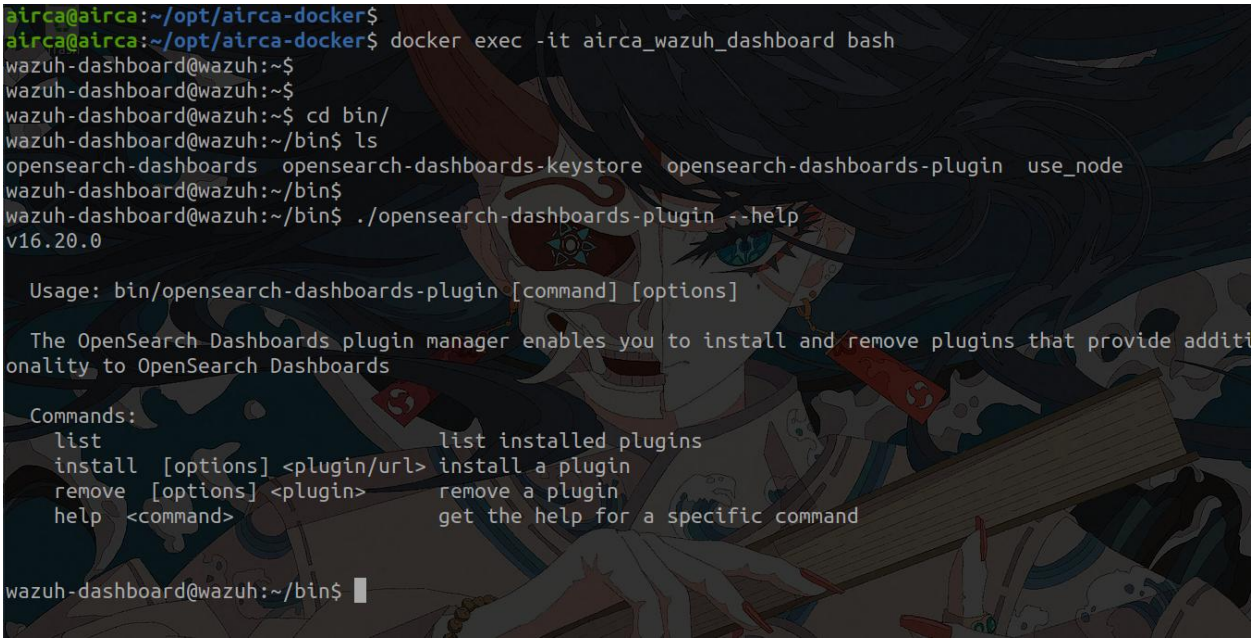


Figure 73 Listing manual of OpenSearch Dashboards Plugins binary

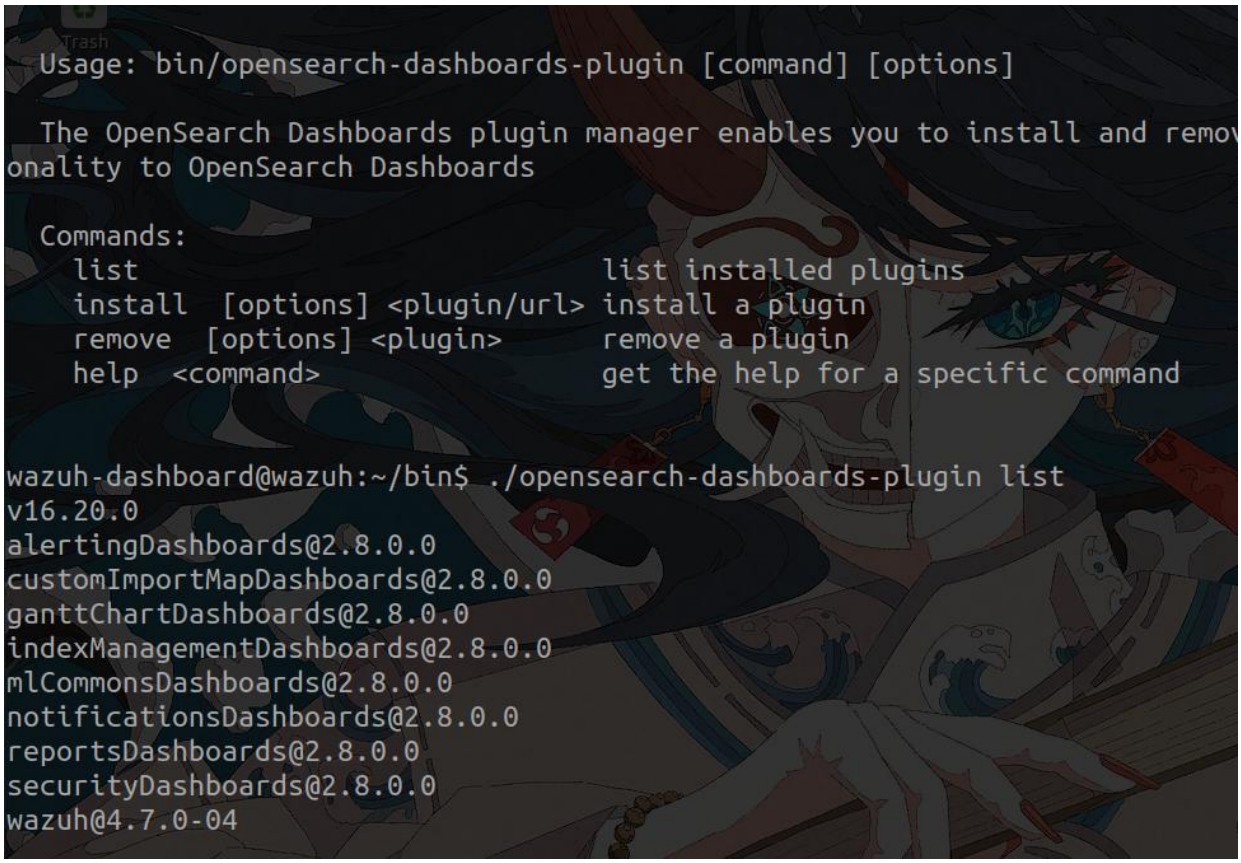


Figure 74 Listing installed plugins in OpenSearch

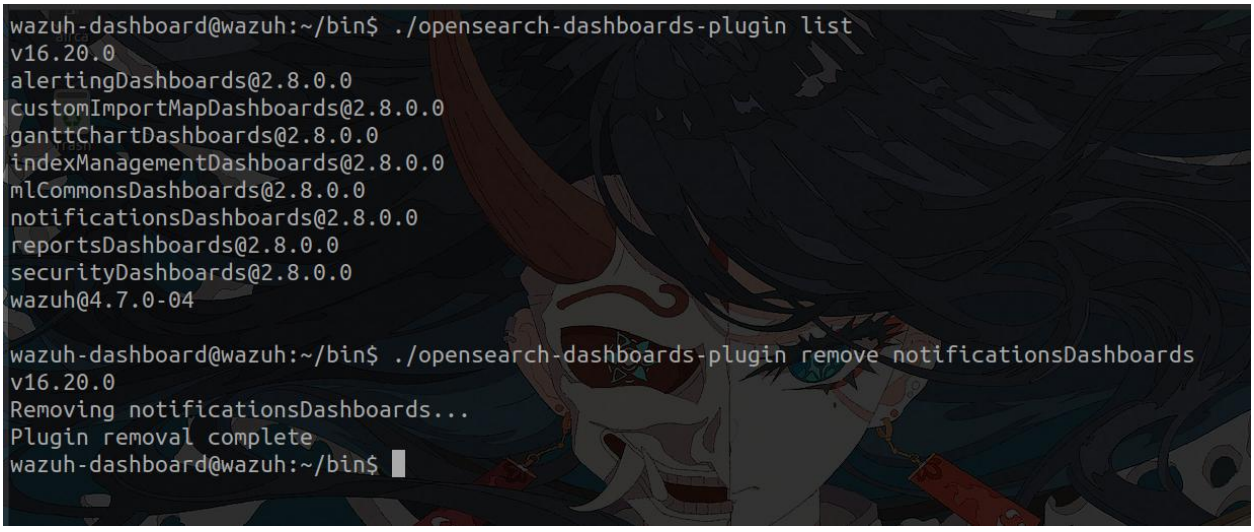


Figure 75 Removed notification dashboards plugin

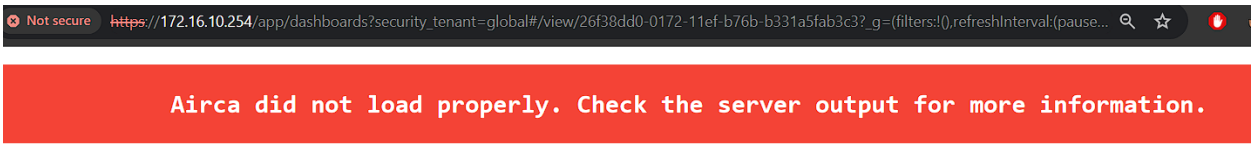


Figure 76 Dashboard did not load properly

4.1.3. Security Testing

Security testing is a form of software testing aimed at detecting and addressing vulnerabilities, risks, and threats within a software application (HackerOne, 2024). Its primary focus is on uncovering all possible flaws and weaknesses in the system. A security test was conducted on this system, and it yielded successful results.

4.1.3.1. Test Plan

Test Cases	Objectives	Results
Case 1	To identify security vulnerability in the windows endpoint.	Successful
Case 2	To identify security vulnerability in the ubuntu endpoint.	Failed

Table 24 Test plans for security testing

4.1.3.1.1. Test Case 1

Security Testing	Objectives
Objective	To identify security vulnerability in the windows endpoint.
Action	Enabling vulnerability detection in Wazuh.
Expected Result	Vulnerability identification alerts should be generated.
Actual Result	Vulnerability identification alerts were generated.
Conclusion	Test successful.

Table 25 Security Testing - Test Case 1

Evidence

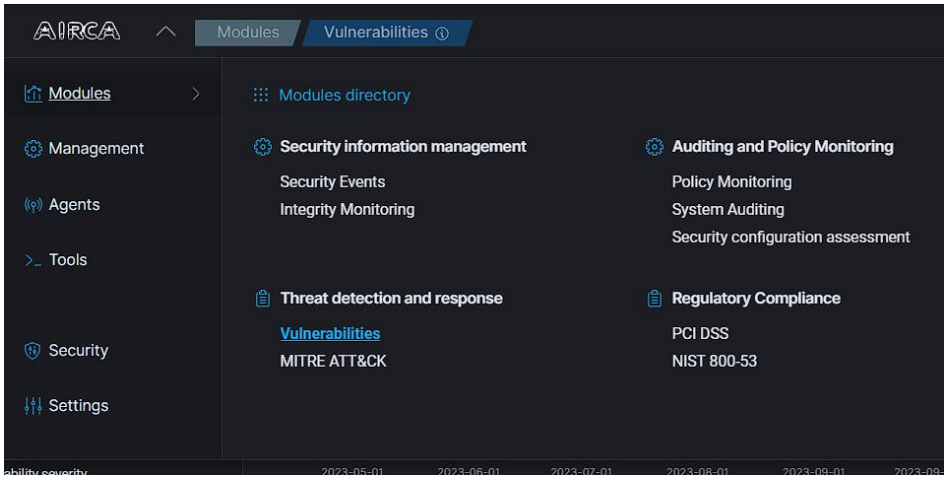


Figure 77 Vulnerabilities module in Wazuh

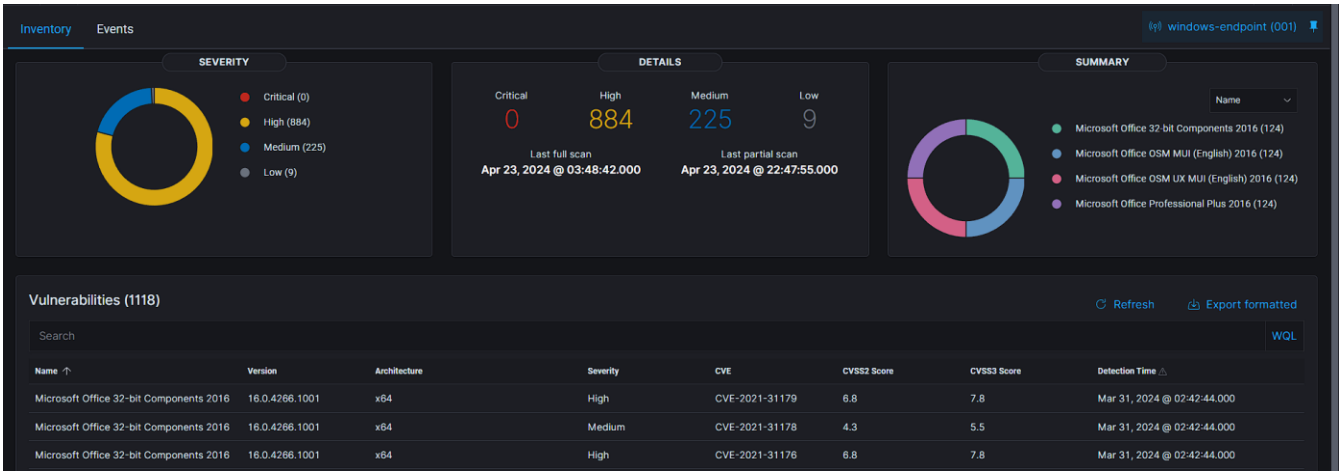


Figure 78 Vulnerabilities alerts for each application in windows endpoint

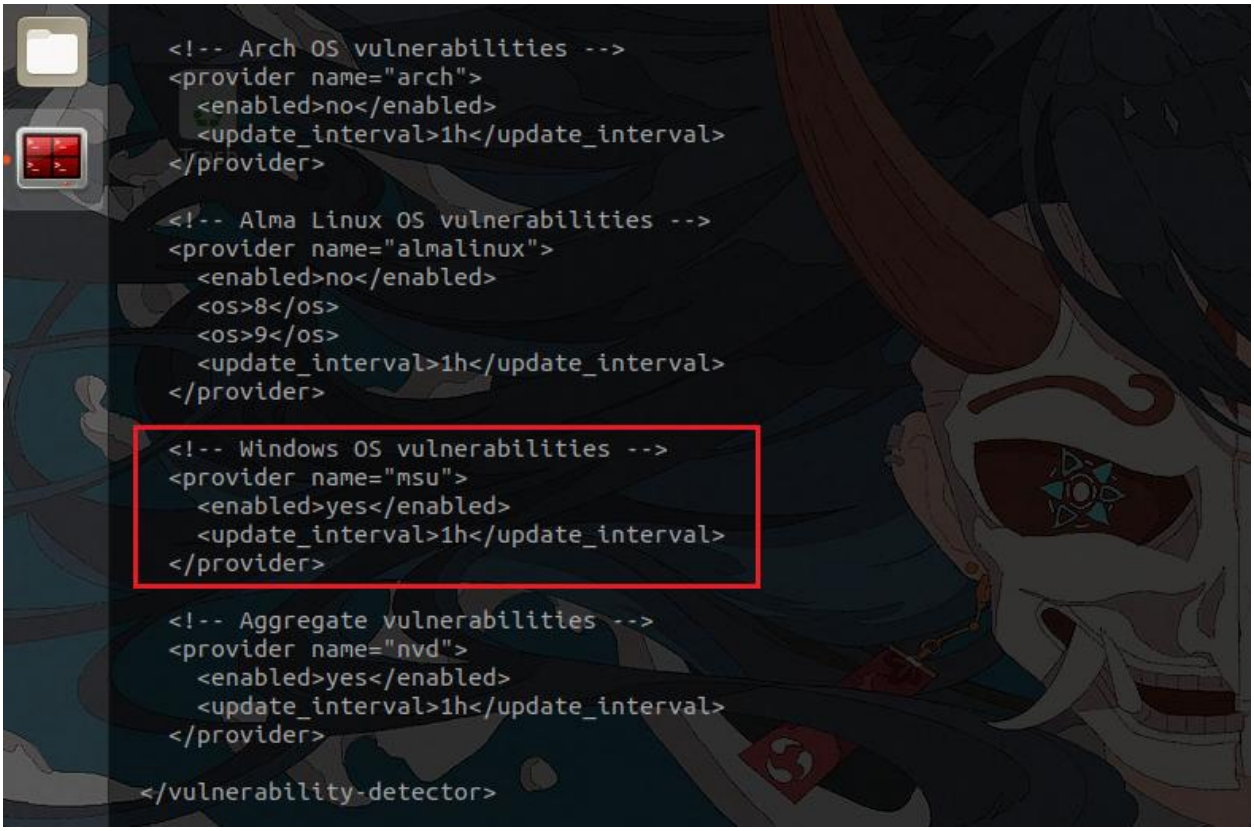


Figure 79 Enabled vulnerability detector for windows before checking events

4.1.3.1.2. Test Case 2

Security Testing	Objectives
Objective	To identify security vulnerability in the ubuntu endpoint.
Action	Enabling vulnerability detection in Wazuh.
Expected Result	Vulnerability identification alerts should be generated.
Actual Result	Vulnerability identification alerts were not generated.
Conclusion	Test failed.

Table 26 Security Testing - Test Case 2

Evidence

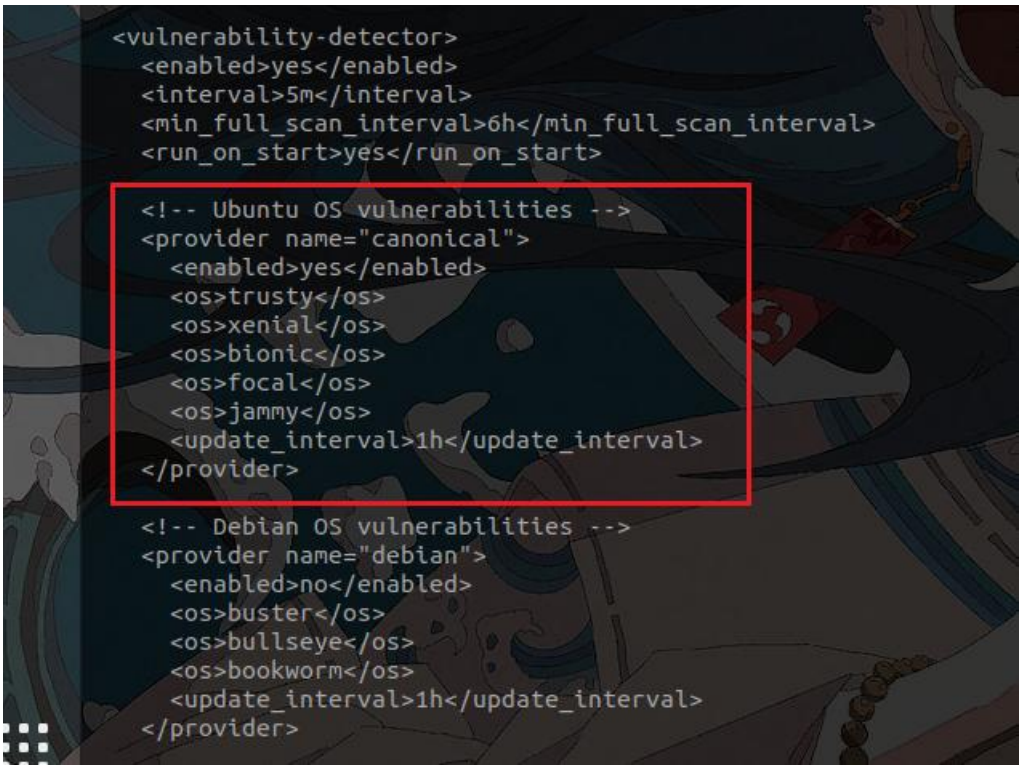


Figure 80 Enabled vulnerability detector for ubuntu before checking events

Figure 81 Restarting windows endpoint agent

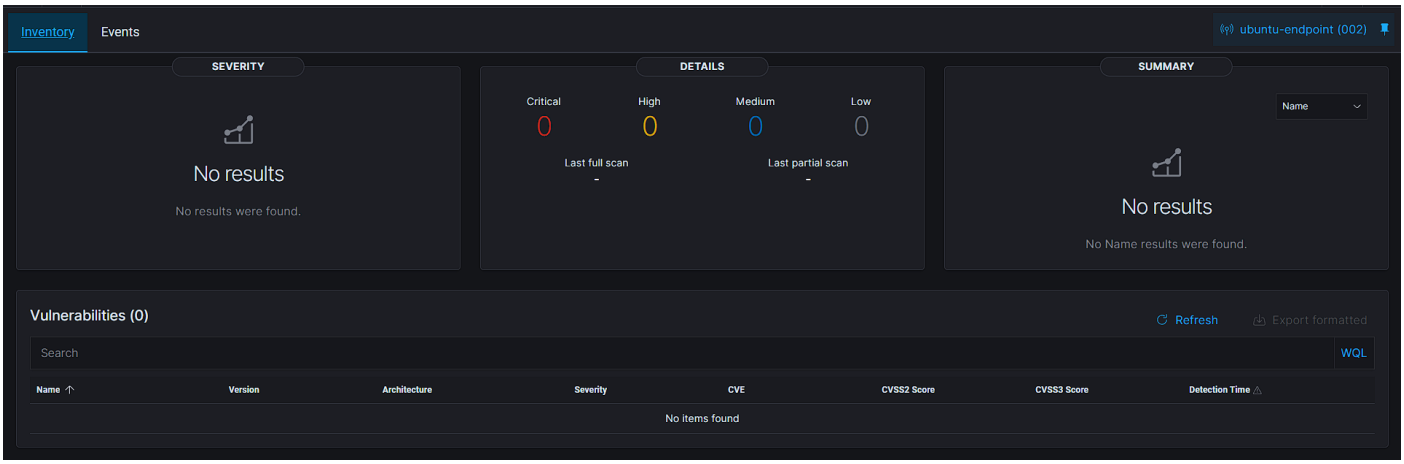


Figure 82 Vulnerabilities alerts in ubuntu endpoint

4.2. Critical Analysis

After the overall analysis, development and testing of the system was completed, it was made clear that the project indeed helps in mitigating a lot of cyber threats in this digital world with its core feature of protect against them with low cost for deploying in production environment.

While developing the system, the official documentation of the open-source projects that was used and implemented were found to be valuable for the development of the project.

The project has made significant improvements with the multiple iterations of development in implementing core components such as Wazuh and MISP integration and developing detection rules and active response mechanisms, but there are areas for potential enhancement and improvements for it grow as a product. The inability to fully customize dashboards and implement the feedback from survey analysis, due to time constraints highlights the importance of managing project scope and resource allocation effectively.

Moving forward, ongoing refinement of development processes, and exploration of strategies to address time constraints will be essential for further optimizing the solution and ensuring its long-term viability in mitigating cyber threats effectively.

Chapter V: Project Risk, Threats, and Contingency Plans

5.1. Project Risks and Threats

The potential risks and threats associated with this project include the following:

i. Possible Hardware Failures
ii. Security Tool Integration Issues
iii. Dependant on Malicious file signatures matching
iv. Threat Intelligence Feed Updates

5.2. Contingency Plans

All the risk and treats mentioned above can be minimized to some extent by following certain contingency plans:

i. Create timely snapshots/backup of the server as a precautionary measure.
ii. Conduct thorough testing and validation during the integration phase to identify and resolve compatibility issues with security tools.
iii. Develop pattern-based analysis rules to detect malicious files.
iv. Implement a threat intelligence update mechanism for evolving threats and adjust the system's response accordingly.

Chapter VI: Conclusion

6.1. Summary

In today's digital world, the rise of cyber threats demands a proactive defence. This project provides an Automated Incident Response for Cyber Anomalies (AIRCA) system, designed to swiftly detect, and respond to emerging threats. Using the dynamic system development model (DSDM), the approach prioritized adaptability, collaboration, and iterative development, ensuring a resilient and user-centric system.

Recognizing potential risks like hardware failures and query limitations, the project incorporated contingency plans to navigate these challenges. AIRCA aimed to strengthen cybersecurity measures with expected outcomes including improved response efficiency and enhanced threat detection. From hardware recommendations to essential software components, each requirement is carefully considered to create a safer and more efficient operation. In essence, this project is not just about technology, it's a proactive step towards building a secure digital future where cyber threats are met with swift and effective responses.

6.2. Advantages

AIRCA is composed of different features and carries numerous advantages for users and some of the major advantages of this project are as follows :

- Makes use of containerization which makes it lightweight and ready to deploy in any environment.
- Collects, correlates and visualizes the logs from endpoints.
- Integrates threat intelligence platform and assists in detecting suspicious domain connections.
- Uses Yara's pattern-based matching instead of traditional signature-based matching for detecting suspicious files.
- Uses vulnerability detector which helps in detecting vulnerabilities in endpoints.
- Assists in quarantining suspicious files in real time.

6.3. Limitations

Some of the limitations of the project are as follows :

- Amount of IOCs in MISP may not be sufficient.
- Yara rules may not be sufficient.
- Pattern based file matching takes more time than signature-based file matching which may result in slow response to the suspicious files.
- Customization of the whole UI needs to be done by modifying the source code.
- Configurations in agents to setup the Yara scan and active response to suspicious files needs to be done manually.
- Integrations with enterprise level applications may not be possible.

6.4. Future Works

For now, AIRCA has been completed such that the aim and objectives of the project has been fulfilled. However, there is vast space for improvement and enhancement in the project where some of the possible future works to improve the project have been mentioned below.

- Multi-tenant can be used to monitor multiple devices across regions.
- Virus Total integration can be used to analyse the files as it can improve detection performance.
- Monitoring suspicious windows registry key modification behaviour properly can be implemented to accurately detect anomaly.
- Automated scheduled reporting can be implemented to check the overall statistics of different events.
- Automated onboarding and offboarding of users can be implemented.
- Automated ticket generation or notifications through mail or messages can be implemented to alert users.
- Integration with incident response platform can be done to maintain structural cases and history of severe alerts.

Chapter VII: Legal, Social and Ethical Issues

7.1. Legal Issues

The project stands on a solid legal foundation. There are no instances of piracy or unauthorized use of software, hardware, or any other resources in its development. Every aspect, from coding to testing and implementation, adheres meticulously to the legal framework of the country. This project upholds the IT regulations outlined in Nepal's digital Security Laws, including the "Information Technology Bill, 2075" and "Privacy Act 2075," ensuring robust digital security measures. Throughout the research and documentation phases, no illicit sites or resources were consulted.

7.2. Social Issues

The project contributes positively to society because the system's development and implementation are guided by a commitment to inclusivity and accessibility, aiming to provide cybersecurity solutions that benefit all users regardless of their socioeconomic status or technological literacy. By prioritizing user-friendly interfaces, AIRCA strives to bridge the digital divide and empower individuals to protect themselves online. The project's focus on automated incident response can alleviate the burden on overstretched cybersecurity teams, potentially addressing workforce shortages in the field and enhancing overall cybersecurity resilience within communities. Through these efforts, AIRCA seeks to not only improve cybersecurity but also promote social equity and digital inclusion.

7.3. Ethical Issues

The project is targeted to any organizations who need to protect their digital assets from different cyber threat and anomalies. In the development and deployment of AIRCA, ethical integrity was a fundamental principle guiding every stage of the project. The system is designed to prioritize cybersecurity measures while simultaneously respecting the privacy and rights of individuals. Through transparent communication and a commitment to ethical data handling practices, AIRCA ensures that data collection and usage are conducted responsibly and with full respect for individuals' autonomy and privacy. By proactively addressing potential ethical concerns, the project aimed to maintain trust and confidence among end users, fostering a positive impact on cybersecurity practices while upholding ethical standards throughout.

Chapter VIII : References and Bibliography

8.1. References and Bibliography

Abdullahi Sani, A. F. S. R. J. I. G., 2013. A Review on Software Development Security Engineering using Dynamic System Method (DSDM). *International Journal of Computer Applications,* pp. 33-44.

Aiman Khan Nazir, I. Z. M. A., 2017. The Impact of Agile Methodology (DSDM) on. *Circulation in Computer Science: International Conference on Engineering, Computing & Information Technology (ICECIT 2017),* pp. 1-6.

Amazon Web Services, Inc. , 2024. *What is Unit Testing? - Unit Testing Explained - AWS.* [Online]
Available at: https://aws.amazon.com/what-is/unit-testing
[Accessed 10 April 2024].

BlackBerry, 2023. *What Is Automated Incident Response?.* [Online]
Available at: https://www.blackberry.com/us/en/solutions/endpoint-security/managed-security-services/incident-response/automated-incident-response
[Accessed 26 December 2023].

Boehm, B. a. H. W., 2001. The Spiral Model as a Tool for Evolutionary Acquisition. *CrossTalk.*

Boehm, B. W., 1988. Computer. *TRW Defense Systems Group,* pp. 61-72.

CrowdStrike, 2024. *2023 Global Threat Report,* Austin, Texas: CrowdStrike.

CyberArts Bilişim A.Ş, 2023. *OpenEDR | CyberArts.* [Online]
Available at: https://cyberartspro.com/en/openedr/
[Accessed 12 April 2024].

HackerOne, 2024. *What Is Security Testing?.* [Online]
Available at: https://www.hackerone.com/knowledge-center/what-security-testing
[Accessed 10 April 2024].

IBM, 2015. *What is SOAR?.* [Online]
Available at: https://www.ibm.com/topics/security-orchestration-automation-response
[Accessed 25 November 2023].

IBM, 2023. *What is incident response?.* [Online]
Available at: https://www.ibm.com/topics/incident-response
[Accessed 26 December 2023].

Microsoft, 2024. *Analyze Azure network security group flow logs - Graylog.* [Online]
Available at: https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-analyze-nsg-flow-logs-graylog
[Accessed 10 April 2024].

MISP, 2023. *MISP (core software) - Open Source Threat Intelligence and Sharing Platform.* [Online]
Available at: https://github.com/MISP/MISP
[Accessed 26 December 2023].

Palo Alto Networks, 2020. *What Is SOAR? - Palo Alto Networks.* [Online]
Available at: https://www.paloaltonetworks.com/cyberpedia/what-is-soar
[Accessed 10 April 2024].

SANS, 2023. *Glossary of Cyber Security Terms.* [Online]
Available at: https://www.sans.org/security-resources/glossary-of-terms/
[Accessed 25 November 2023].

Skopik, D. F., 2023. *Anomaly Detection & Cyber Threat Intelligence - AIT Austrian Institute Of Technology.* [Online]
Available at: https://www.ait.ac.at/en/research-topics/cyber-security/anomaly-detection-cyber-threat-intelligence
[Accessed 10 April 2024].

SonicWall, Inc, 2024. *2024 SonicWall Cyber Threat Report,* s.l.: SonicWall, Inc.

Splunk Inc., 2024. *Splunk Enterprise | Splunk.* [Online]
Available at: https://www.splunk.com/en_us/products/splunk-enterprise.html
[Accessed 10 April 2024].

Testsigma Inc., 2023. *What is System Testing - A Comprehensive Guide.* [Online]
Available at: https://testsigma.medium.com/what-is-system-testing-a-comprehensive-guide-e2f41857954e
[Accessed 7 April 2024].

Wazuh Inc., 2024. *User Manual.* [Online]
Available at: https://documentation.wazuh.com/current/user-manual/index.html
[Accessed 10 January 2024].

Wazuh, 2023. *Wazuh - The Open Source Security Platform. Unified XDR and SIEM protection for endpoints and cloud workloads..* [Online]
Available at: https://github.com/wazuh/wazuh
[Accessed 26 December 2023].

Chapter IX: Appendix

9.1. Appendix A: Definitions

9.1.1. Defining DSDM

The dynamic system development methodology (DSDM) represents an agile software development approach characterized by its iterative and incremental nature, emphasizing swift delivery and sustained user involvement throughout the project (Abdullahi Sani, 2013). DSDM facilitates the dynamic development of systems, accommodating both object-oriented and functional design approaches. Particularly suitable for projects with evolving or unfixed requirements, DSDM permits revisiting earlier phases of the software development life cycle.

DSDM consists of five phases:

i. **Feasibility Study:** Assess the technical and business feasibility of the project. Identify scope, constraints, risks, and potential benefits.

ii. **Business Study:** Understand the business processes and user needs. Gather and refine requirements, possibly creating a prototype to visualize the solution.

iii. **Functional Model Iteration:** Develop and refine core functionalities in an iterative manner and create a working prototype for user review and testing in this phase.

iv. **Design and Build Iteration:** Incrementally design the system architecture and build the software. Each iteration adds features and refines existing ones based on feedback.

v. **Implementation:** Implement the system based on design specifications. This phase involves coding, testing, and integrating components.

DSDM, the dynamic system development model also shown in *Figure 10*, follows key principles for effective project management. It prioritizes active user involvement, grants decision-making power to teams, and emphasizes regular product deliveries for ongoing feedback. Acceptance is based on business needs, and development is gradual with reversible changes (Abdullahi Sani, 2013). The model maintains high-quality standards, integrates testing at every stage, and promotes collaborative teamwork. Overall, DSDM aims for user-focused and successful project outcomes.

9.2. Appendix B: Pre-Survey

9.2.1. Pre-Survey Questions

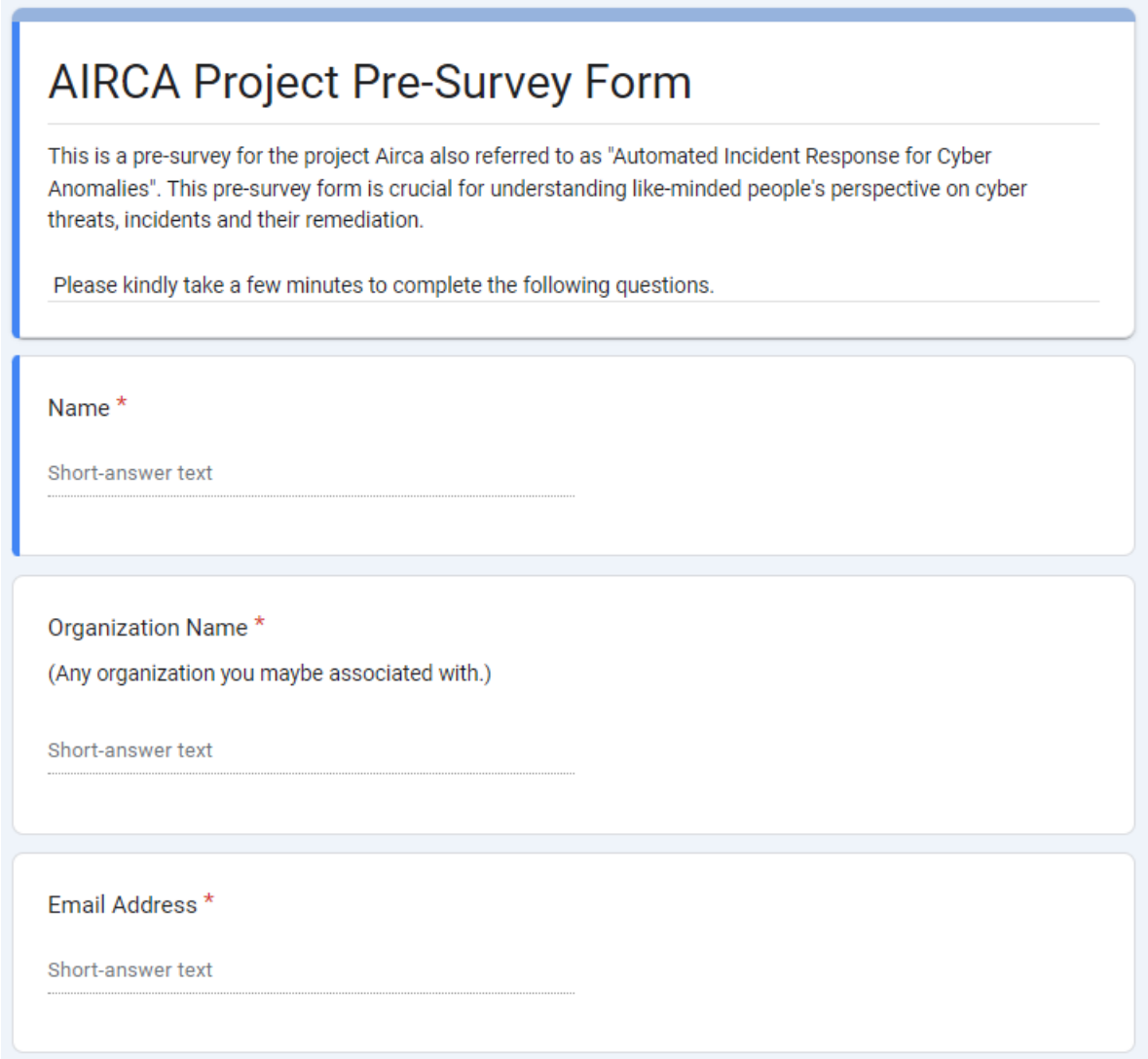


Figure 83 Pre-Survey Form : Personal Details

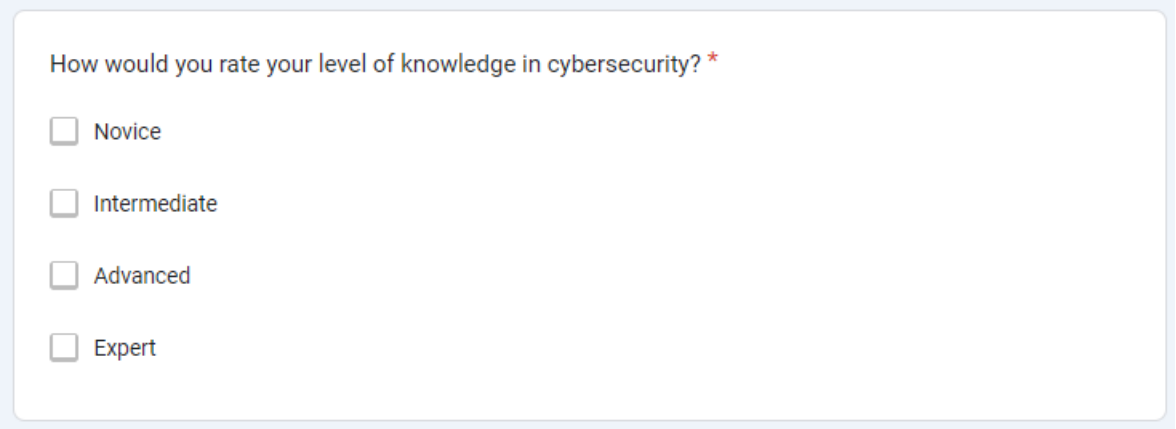


Figure 84 Pre-Survey : Question 1

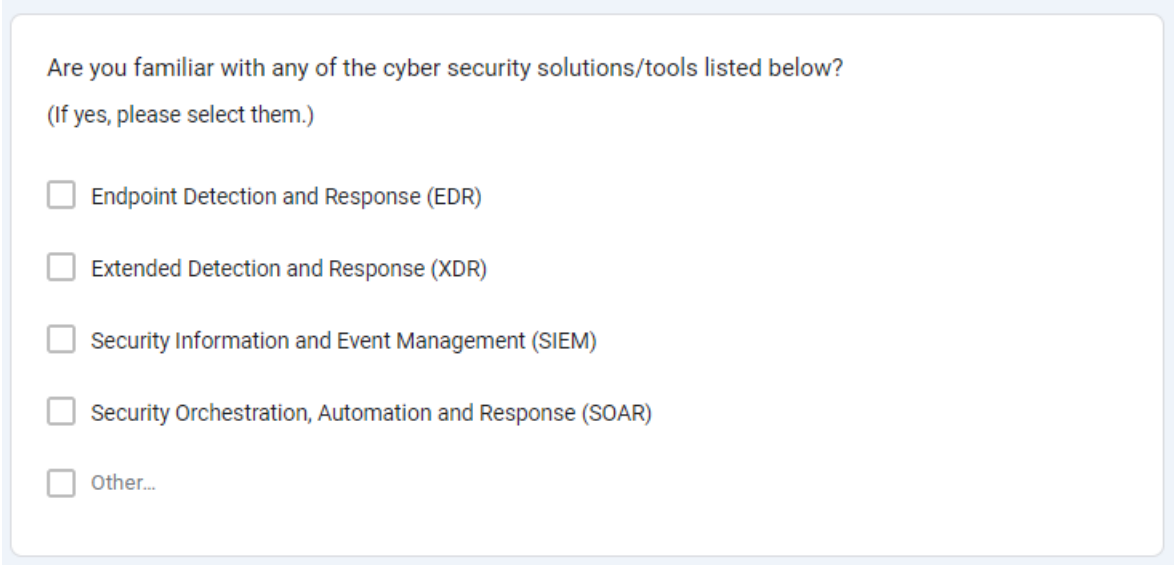


Figure 85 Pre-Survey : Question 2

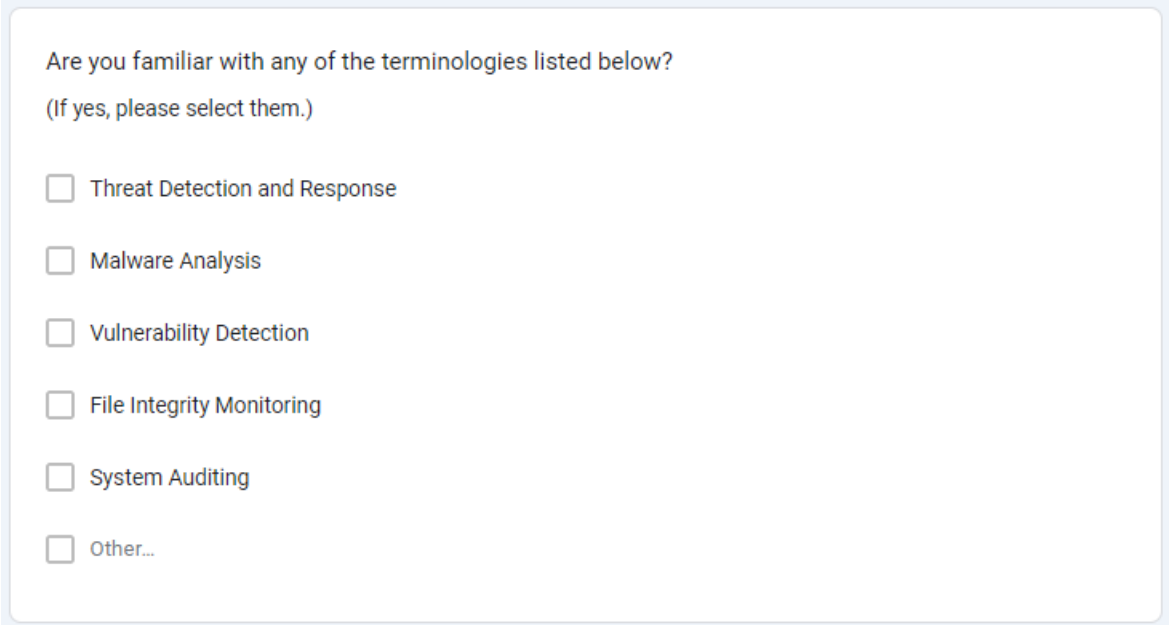


Figure 86 Pre-Survey : Question 3

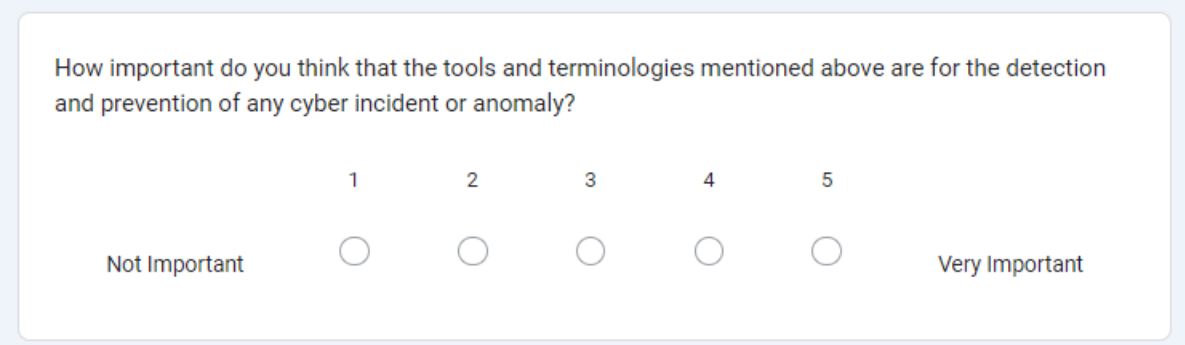


Figure 87 Pre-Survey : Question 4

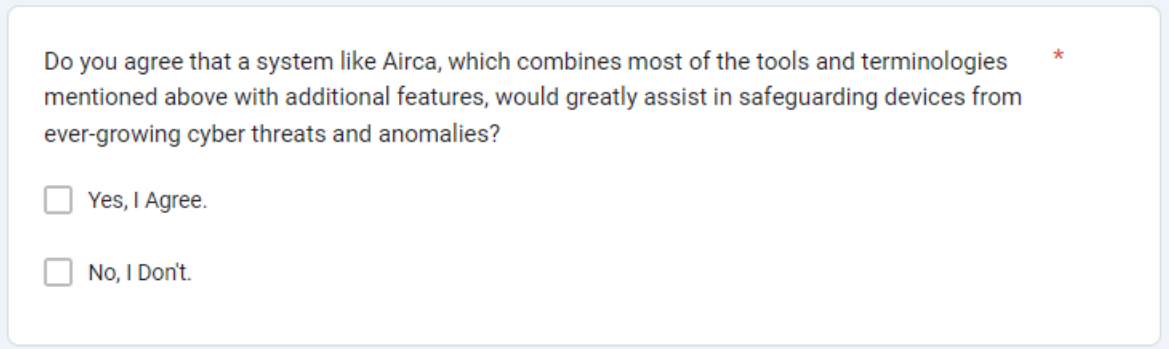


Figure 88 Pre-Survey : Question 5

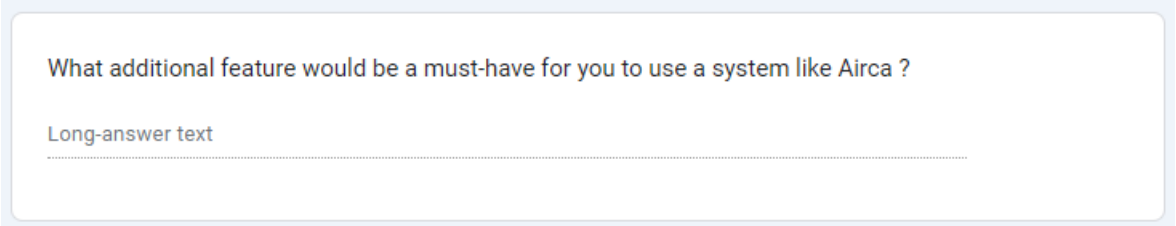


Figure 89 Pre-Survey : Question 6

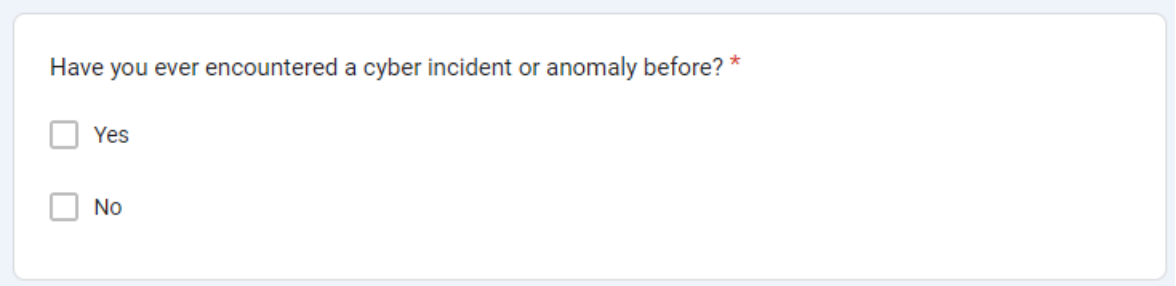


Figure 90 Pre-Survey : Question 7

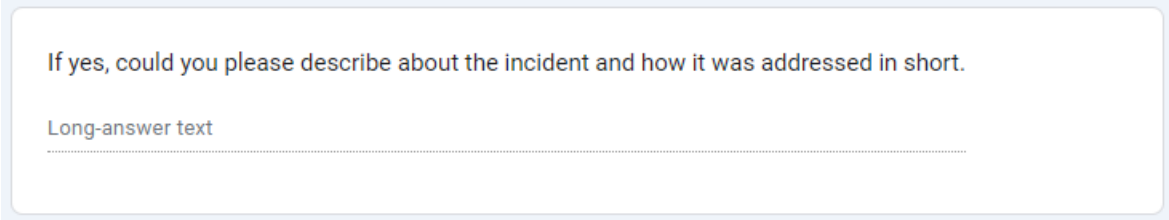


Figure 91 Pre-Survey : Question 8

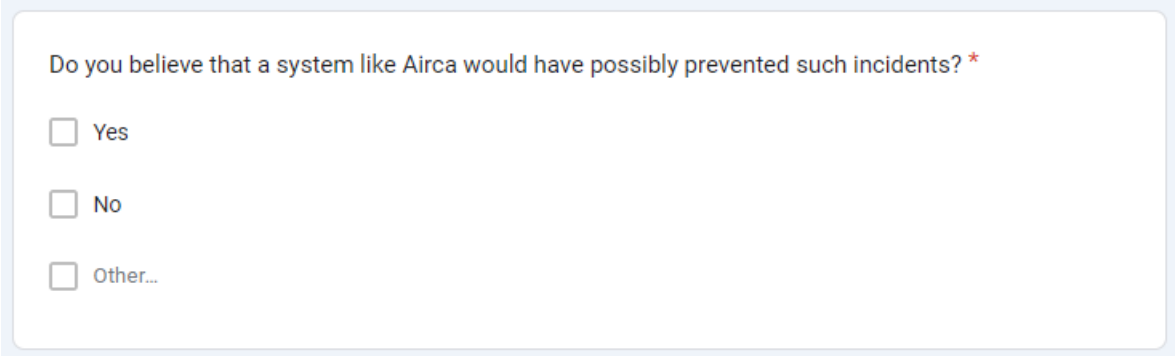


Figure 92 Pre-Survey : Question 9

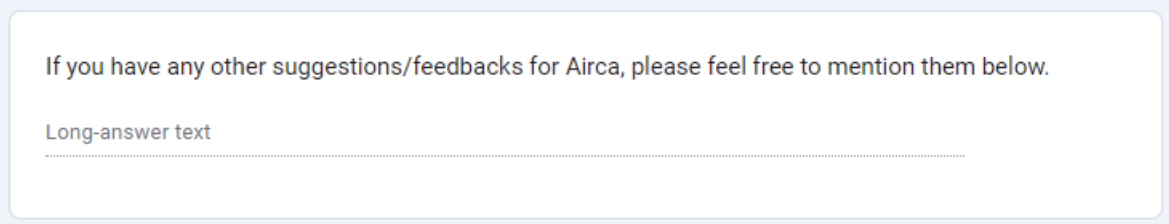


Figure 93 Pre-Survey : Question 10

9.2.2. Pre-Survey Sample

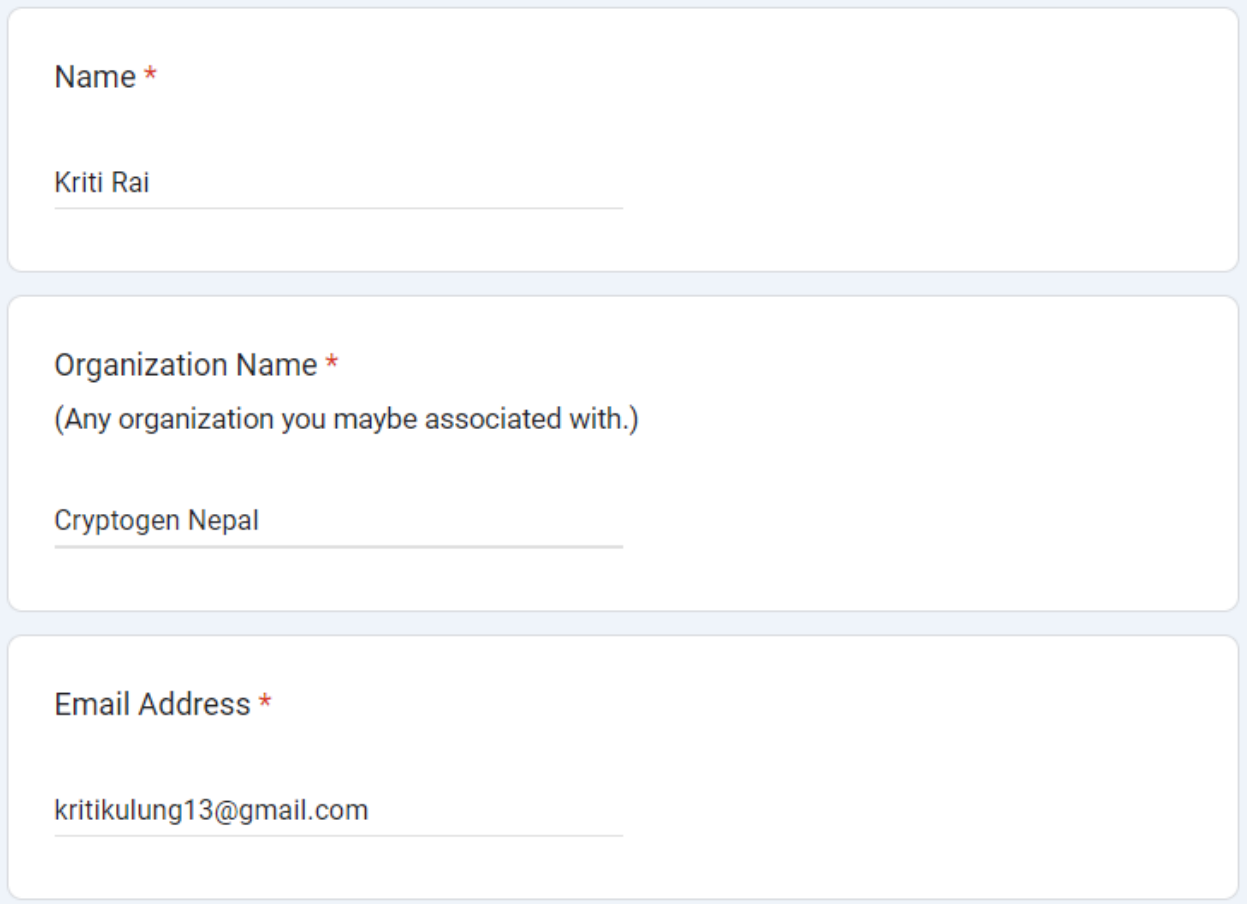


Figure 94 Pre-Survey Sample Feedback: Personal Details

How would you rate your level of knowledge in cybersecurity? *

- [ ] Novice
- [x] Intermediate
- [ ] Advanced
- [ ] Expert

Are you familiar with any of the cyber security solutions/tools listed below?
(If yes, please select them.)

- [x] Endpoint Detection and Response (EDR)
- [x] Extended Detection and Response (XDR)
- [x] Security Information and Event Management (SIEM)
- [x] Security Orchestration, Automation and Response (SOAR)
- [ ] Other:

Are you familiar with any of the terminologies listed below?
(If yes, please select them.)

- [x] Threat Detection and Response
- [x] Malware Analysis
- [x] Vulnerability Detection
- [x] File Integrity Monitoring
- [x] System Auditing
- [ ] Other:

Figure 95 Pre-Survey Sample Feedback: Part 1

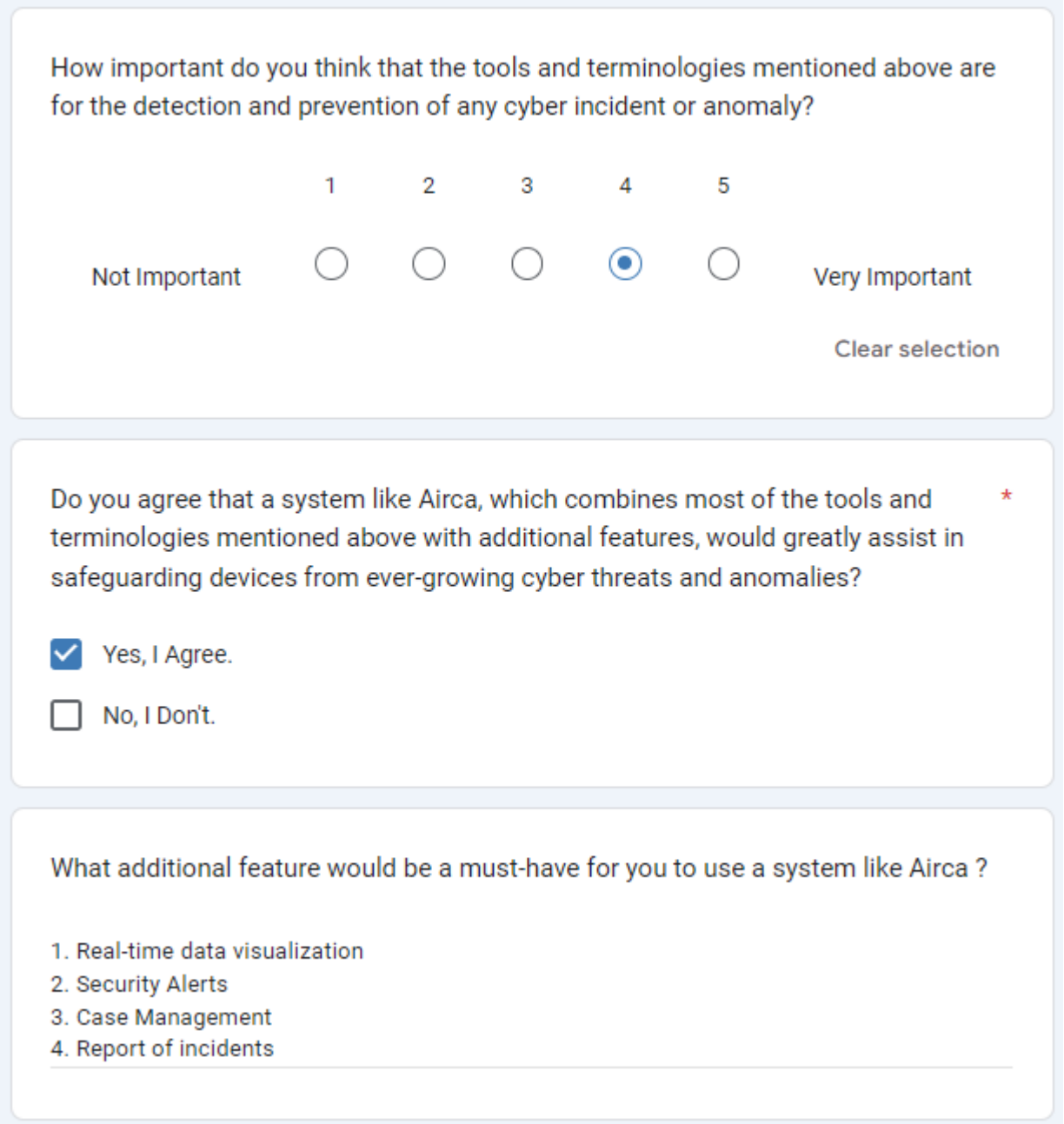


Figure 96 Pre-Survey Sample Feedback: Part 2

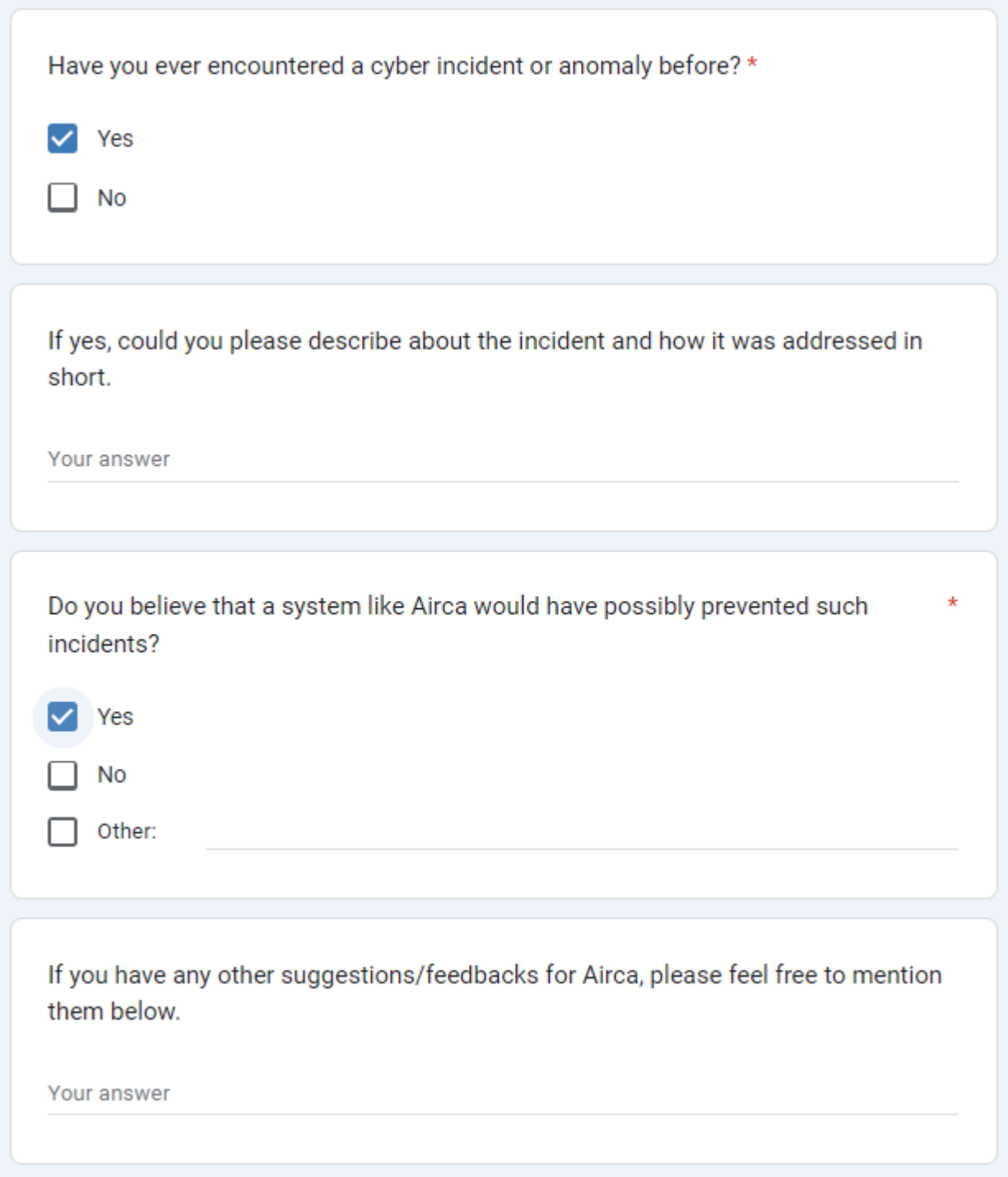


Figure 97 Pre-Survey Sample Feedback: Part 3

9.2.3. Pre-Survey Responses

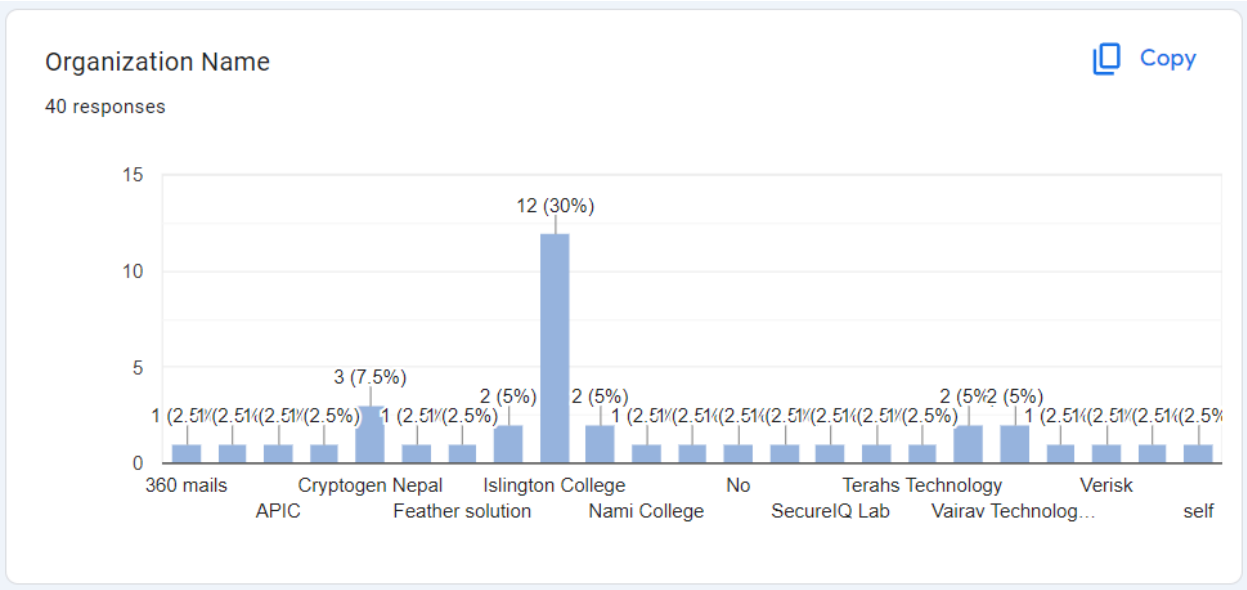


Figure 98 Pre-Survey Response : Organizations

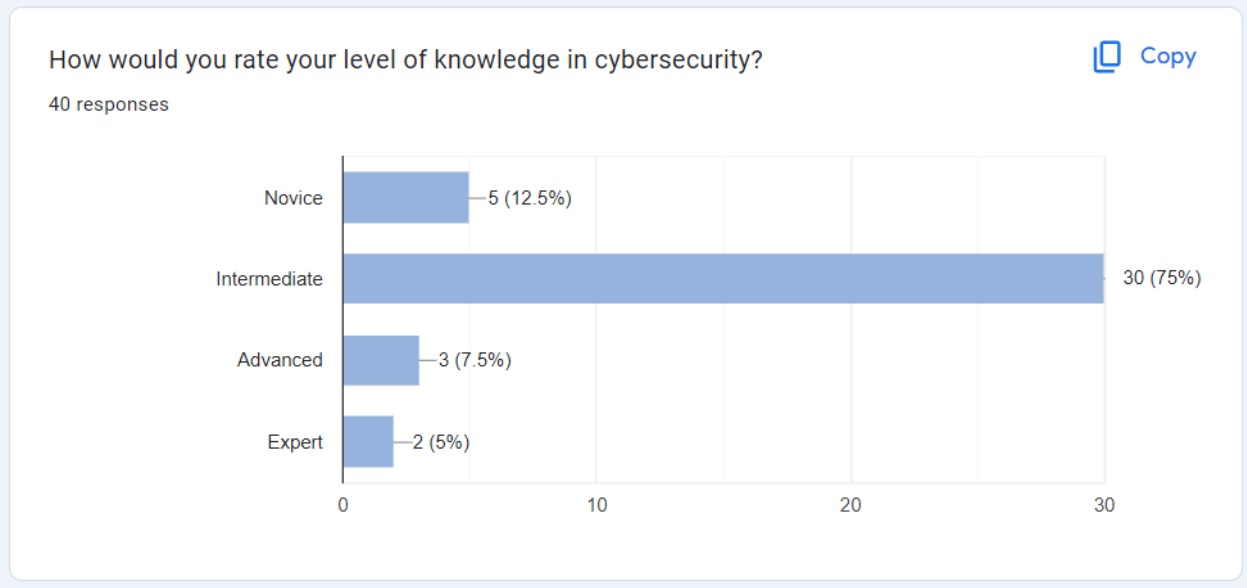


Figure 99 Pre-Survey Response : Question 1

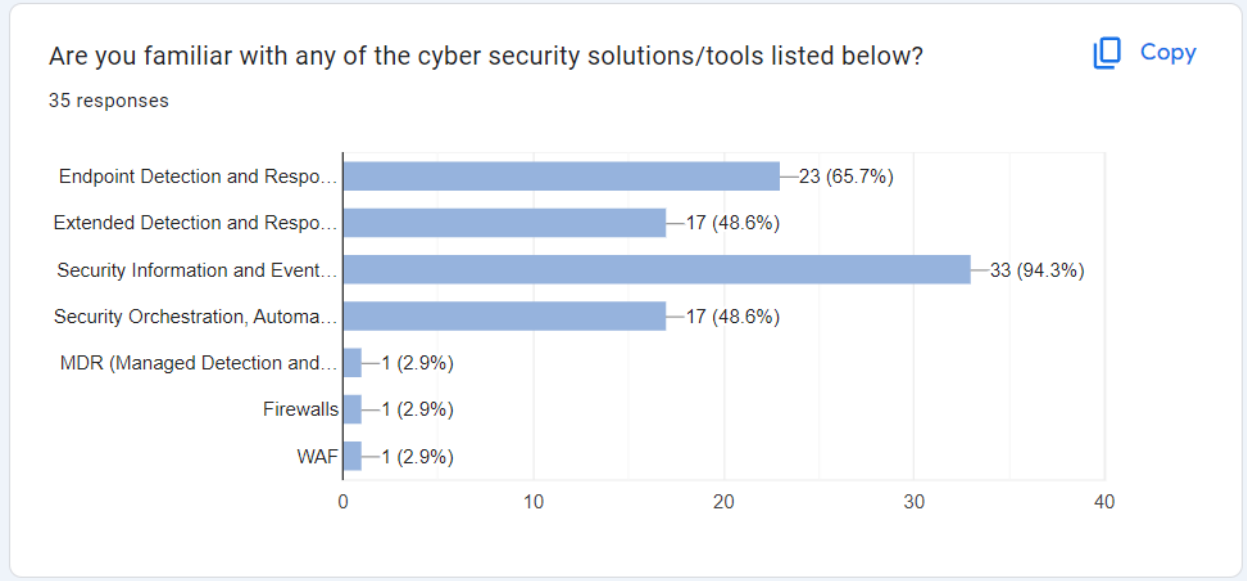


Figure 100 Pre-Survey Response : Question 2

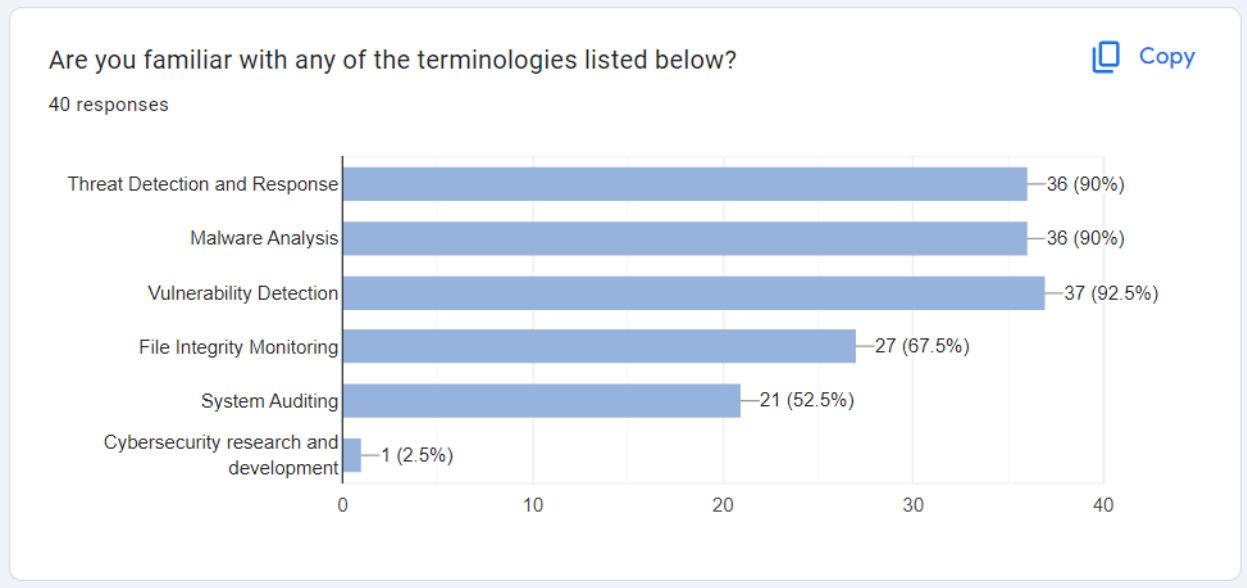


Figure 101 Pre-Survey Response : Question 3

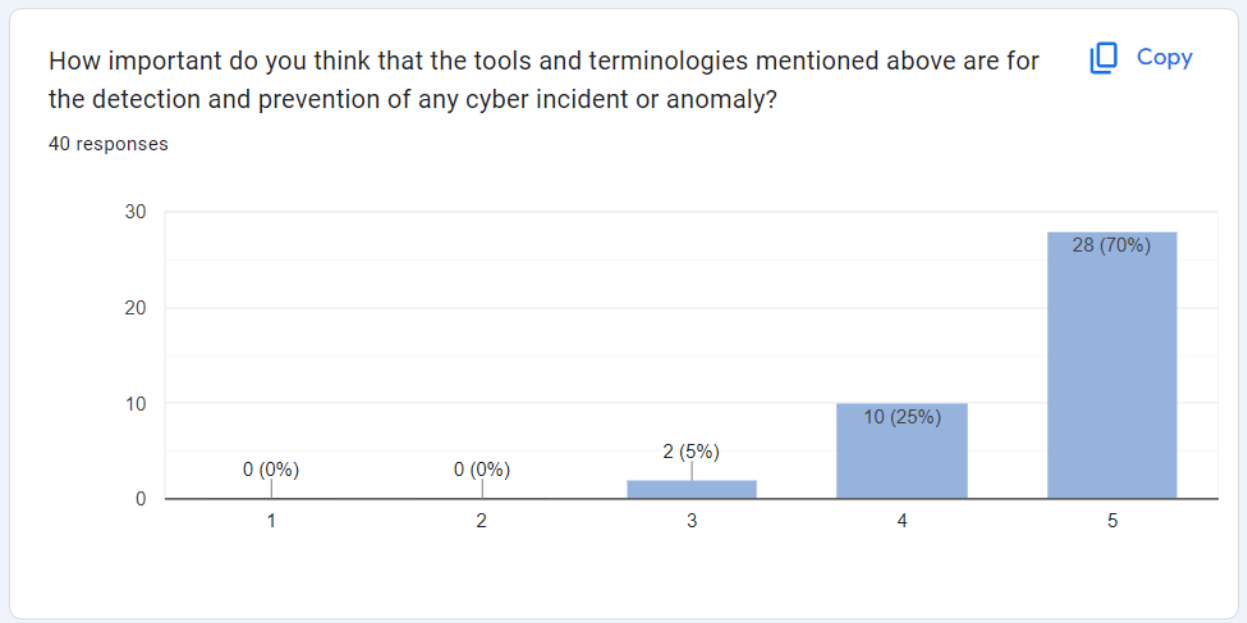


Figure 102 Pre-Survey Response : Question 4

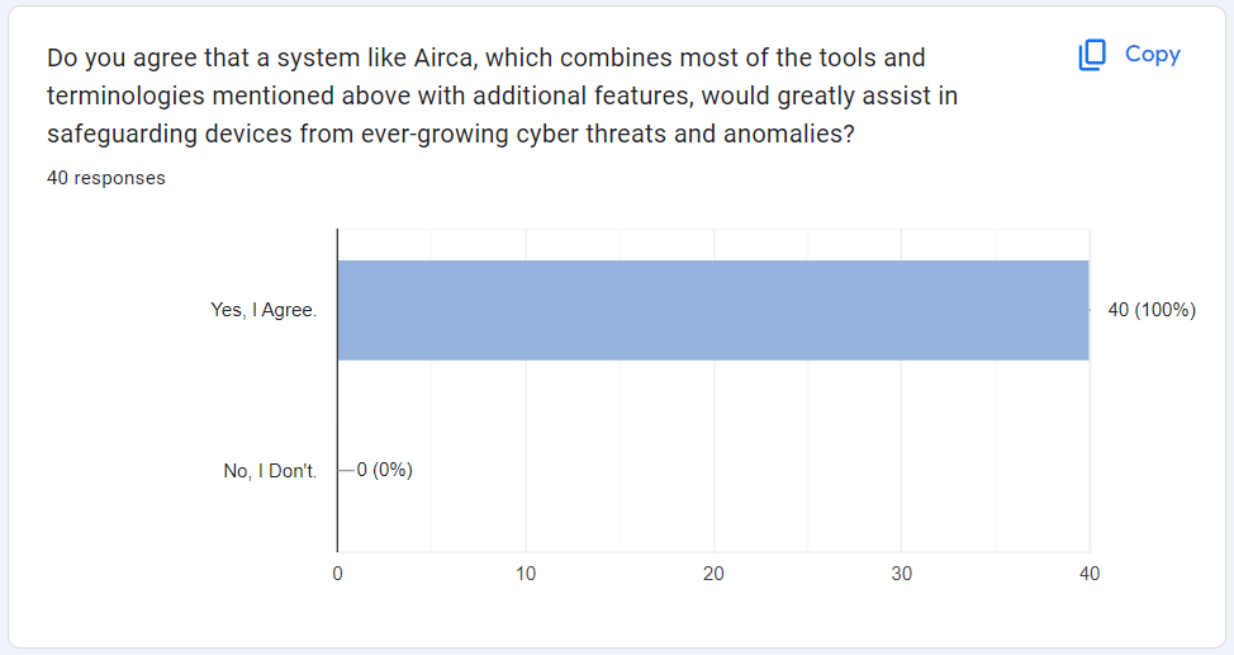


Figure 103 Pre-Survey Response : Question 5

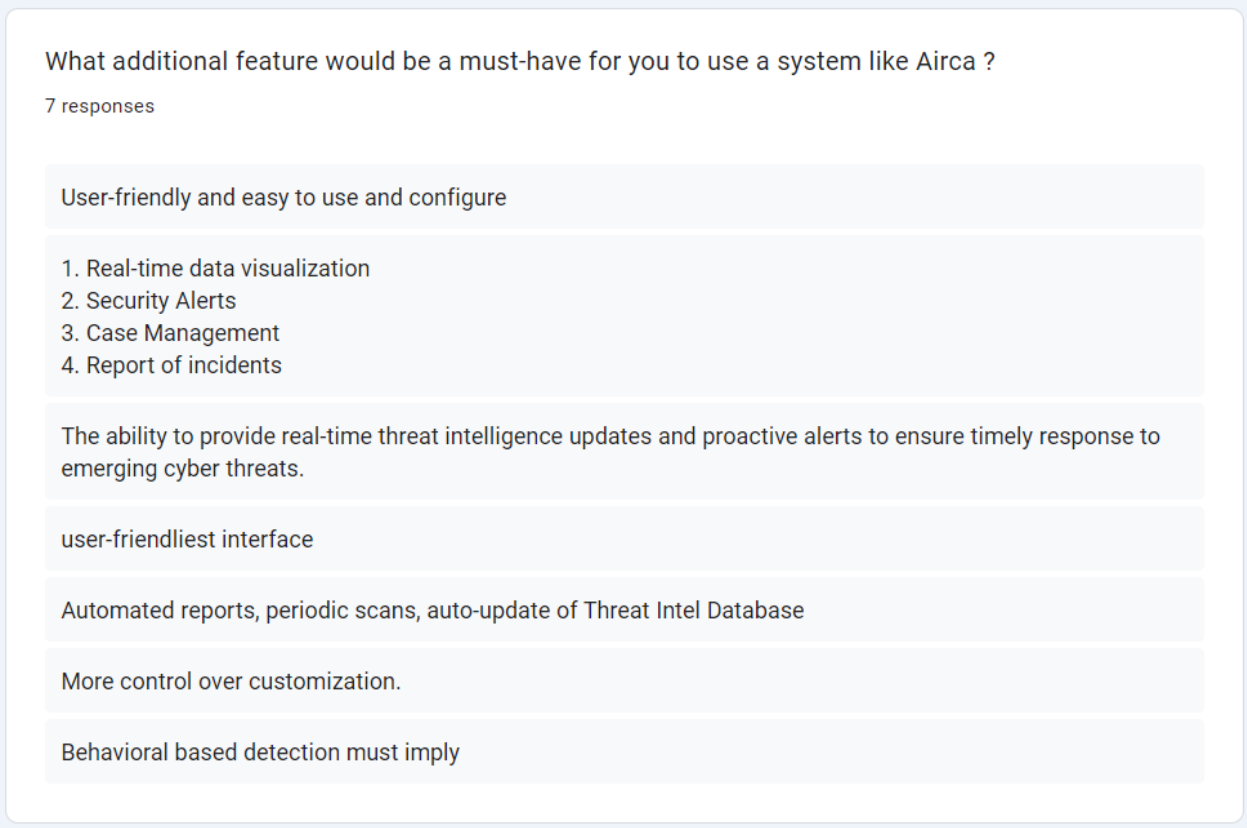


Figure 104 Pre-Survey Response : Question 6

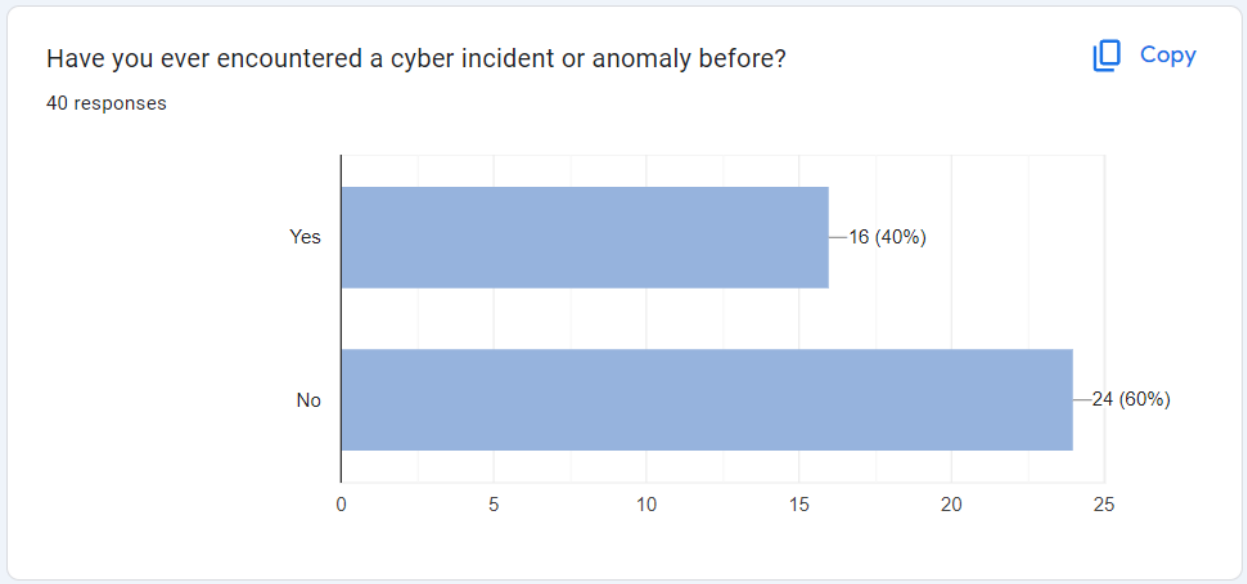


Figure 105 Pre-Survey Response : Question 7

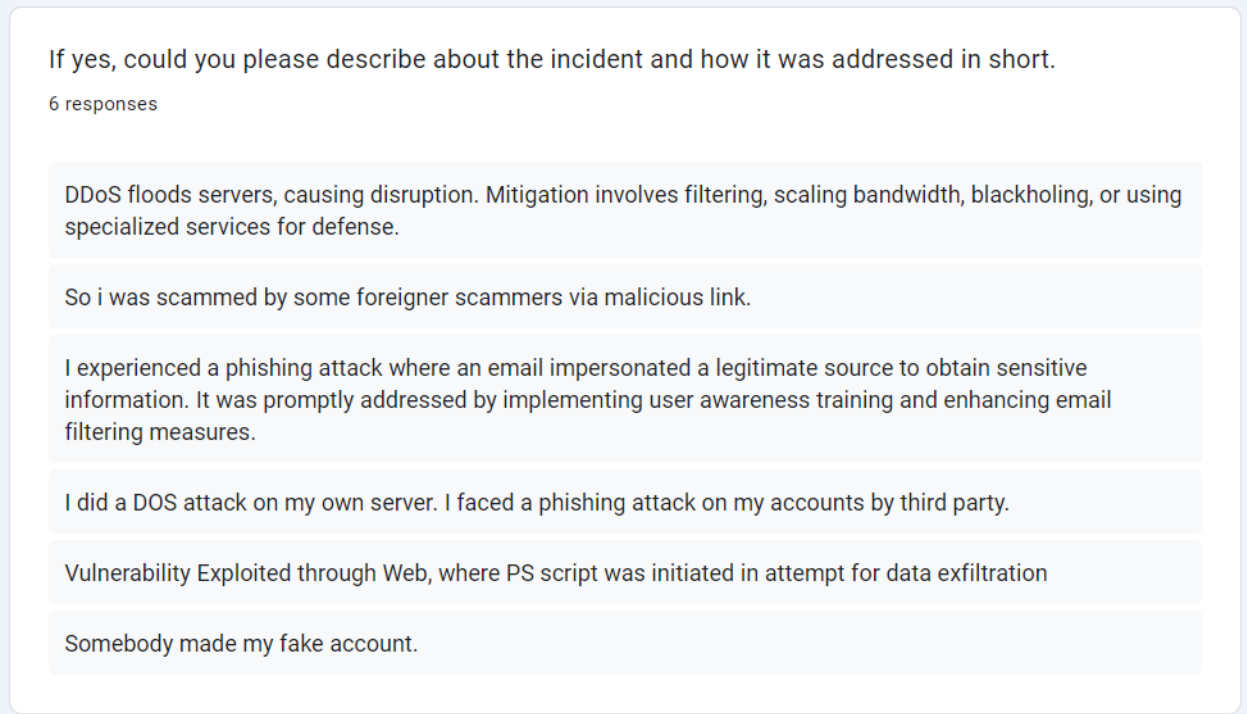


Figure 106 Pre-Survey Response : Question 8

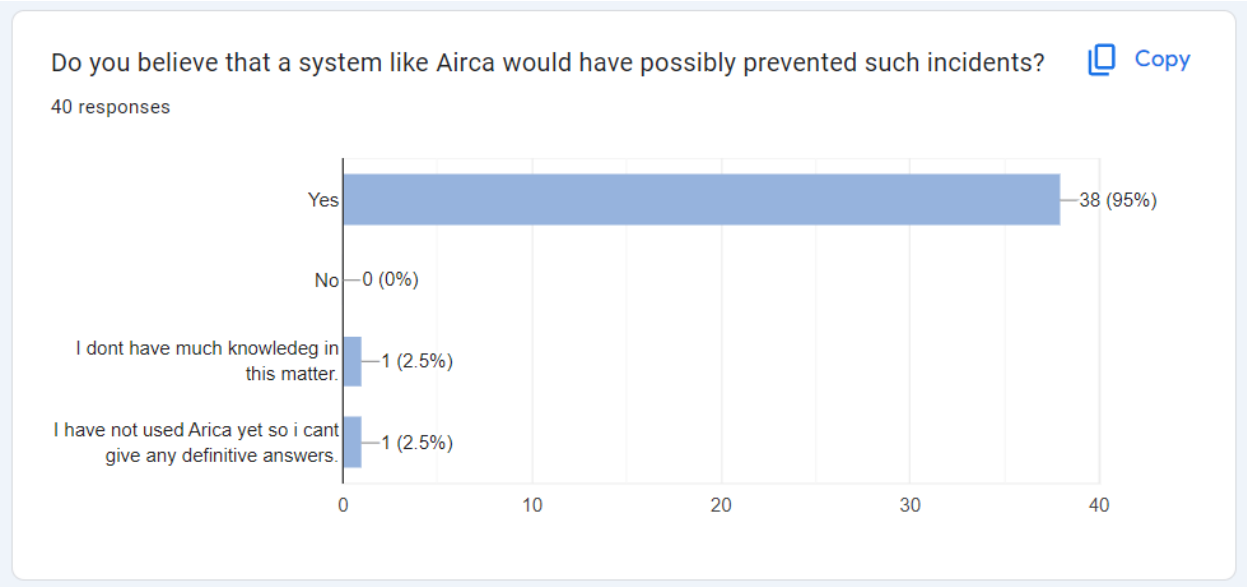


Figure 107 Pre-Survey Response : Question 9

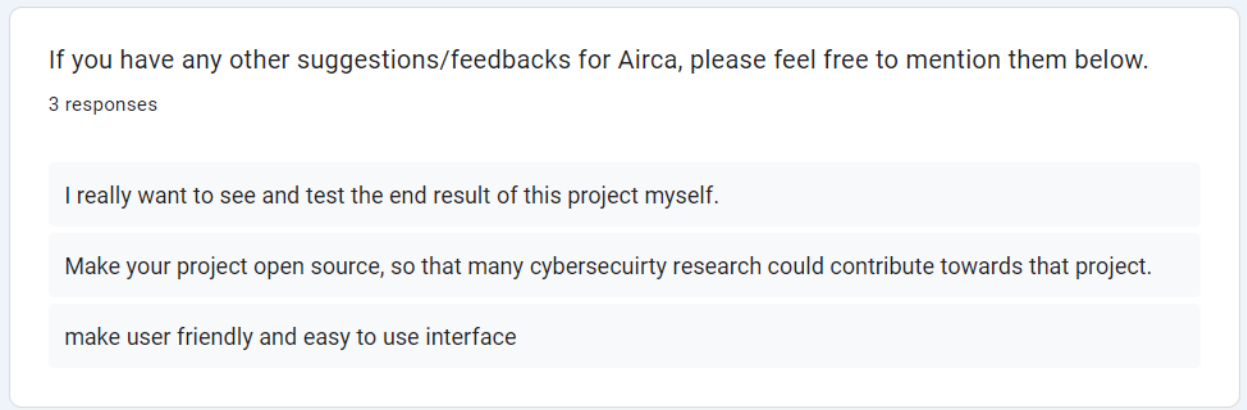


Figure 108 Pre-Survey Response : Question 10

9.3. Appendix C: Post-Survey

9.3.1. Post-Survey Questions

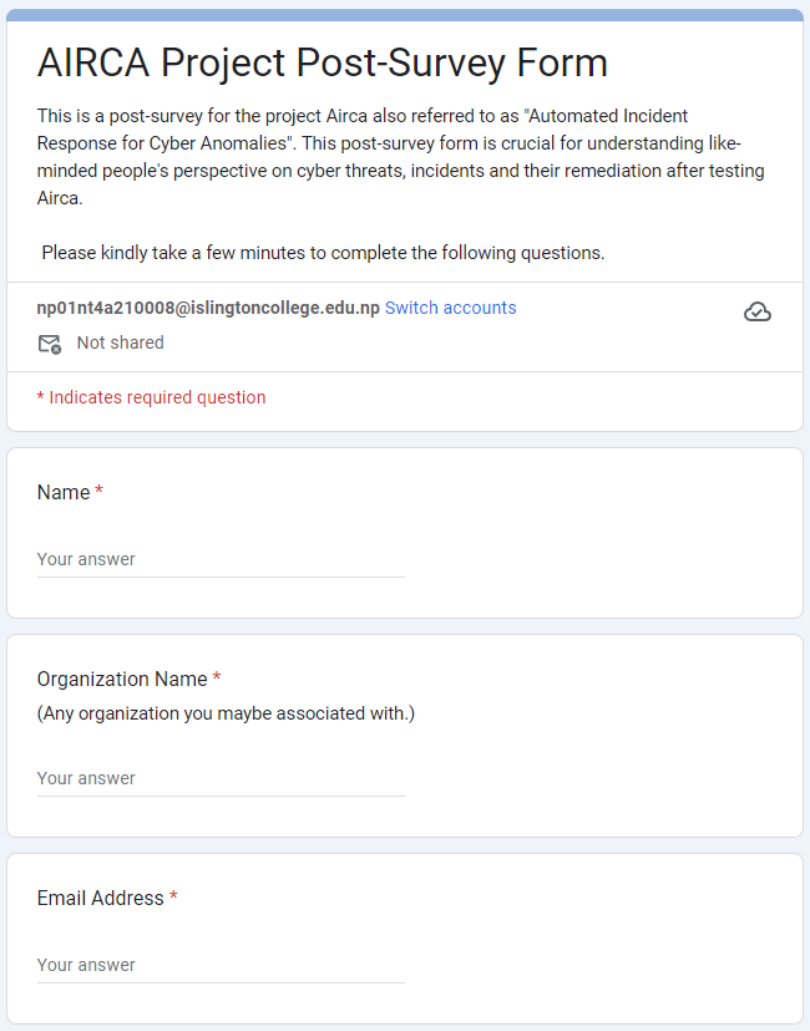


Figure 109 Post-Survey Form: Personal Details

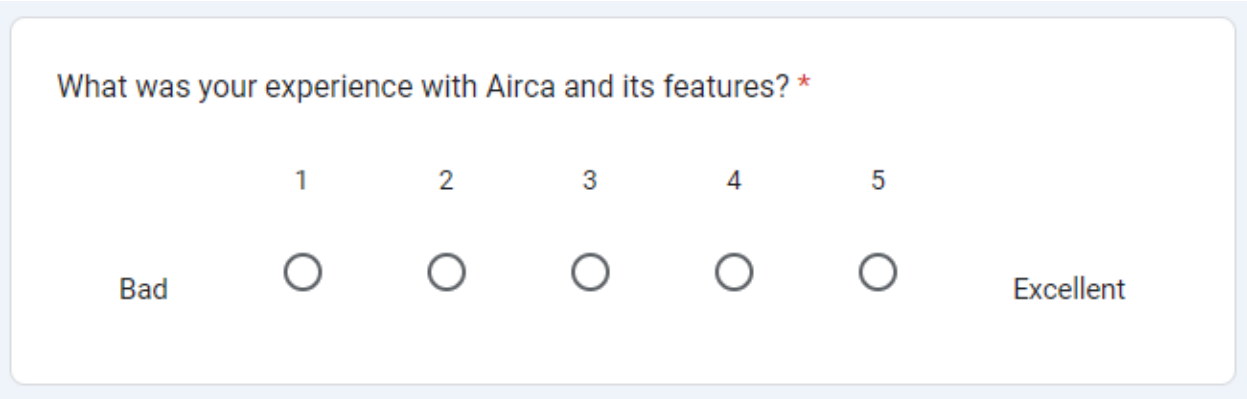


Figure 110 Post-Survey: Question 1

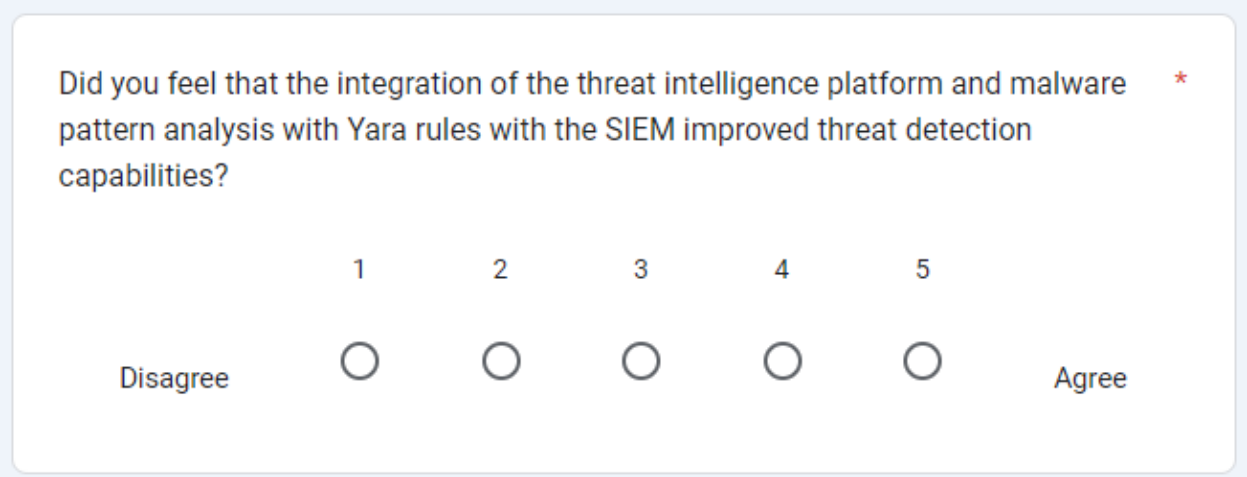


Figure 111 Post-Survey: Question 2

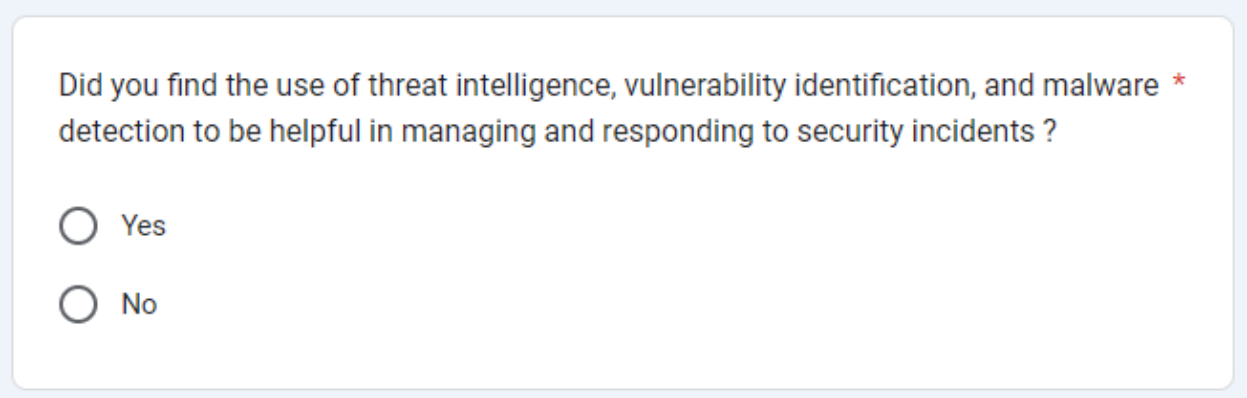


Figure 112 Post-Survey: Question 3

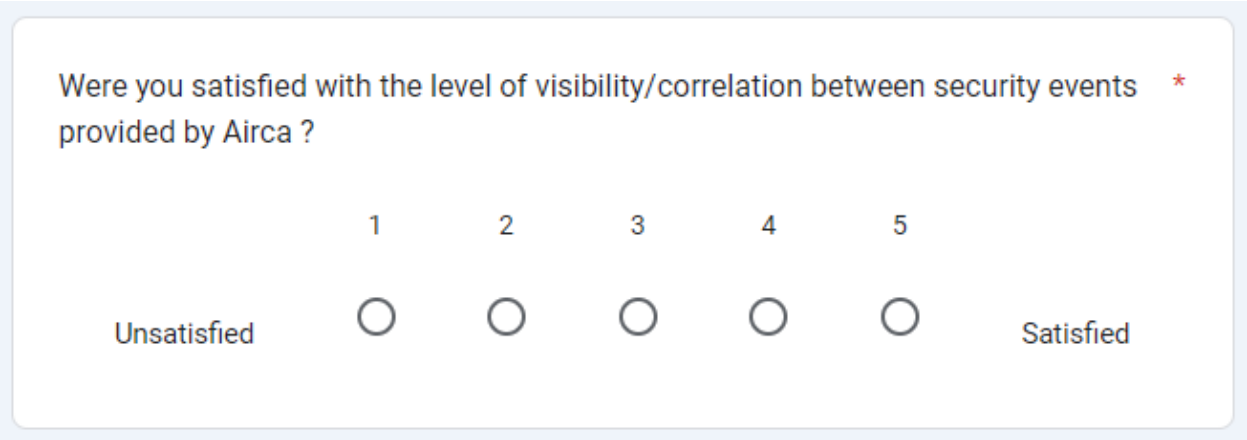


Figure 113 Post-Survey: Question 4

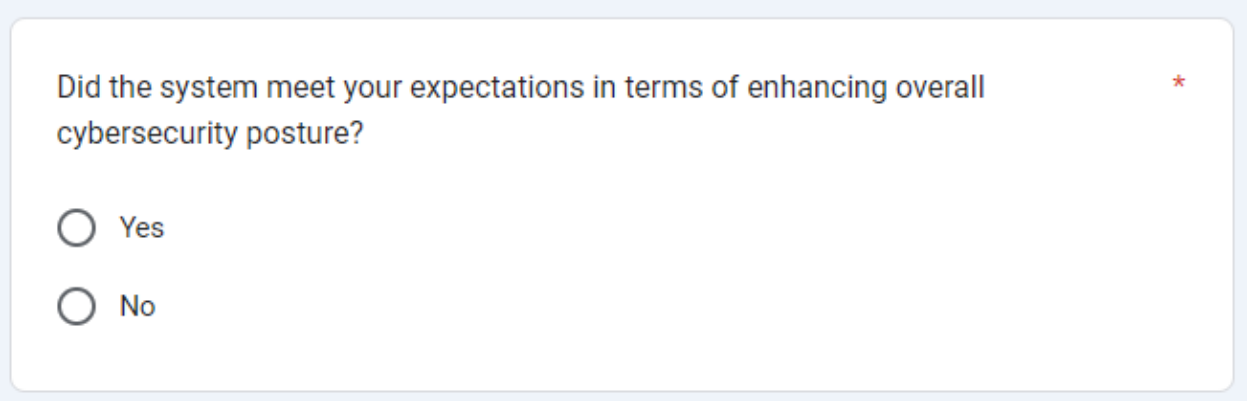


Figure 114 Post-Survey: Question 5

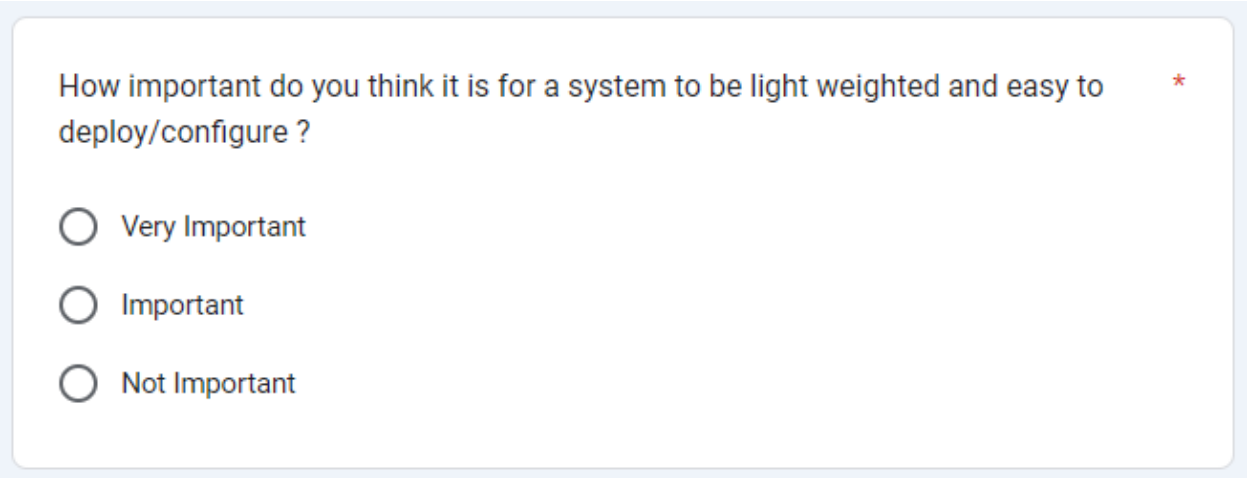


Figure 115 Post-Survey: Question 6

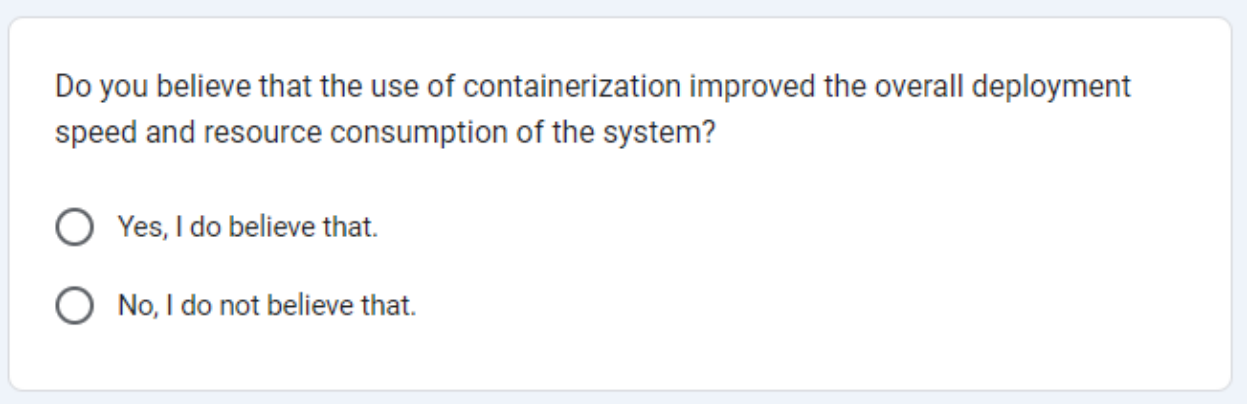


Figure 116 Post-Survey: Question 7

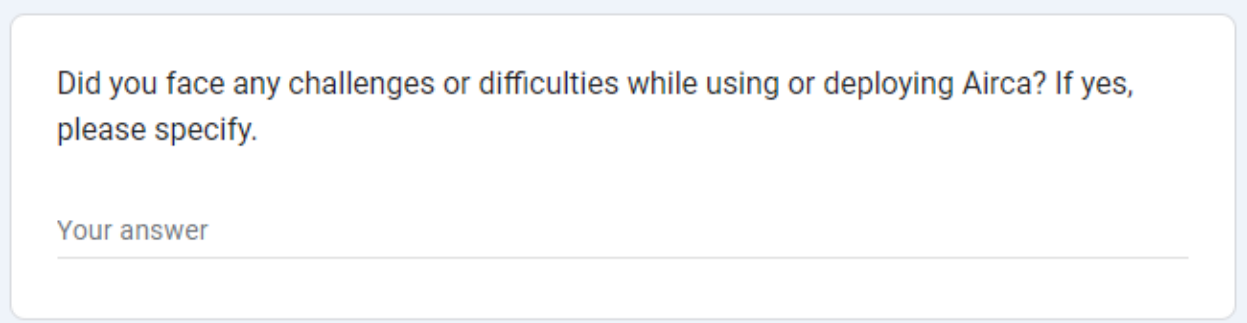


Figure 117 Post-Survey: Question 8

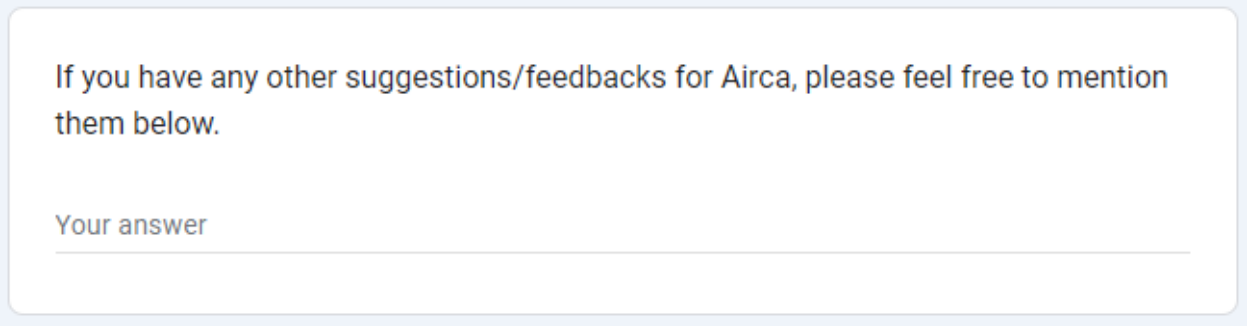


Figure 118 Post-Survey: Question 9

9.3.2. Post-Survey Sample

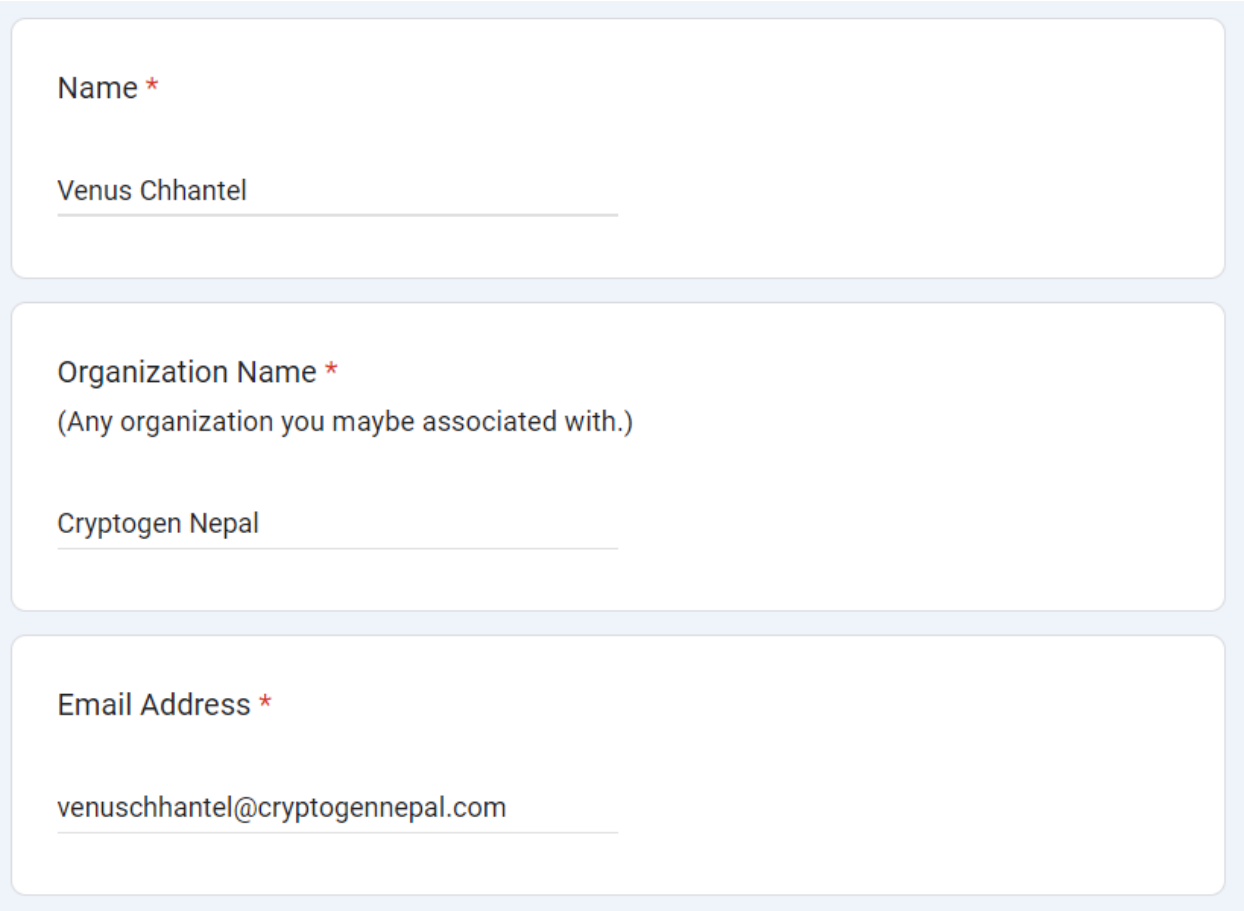


Figure 119 Post-Survey Sample Feedback: Personal Details

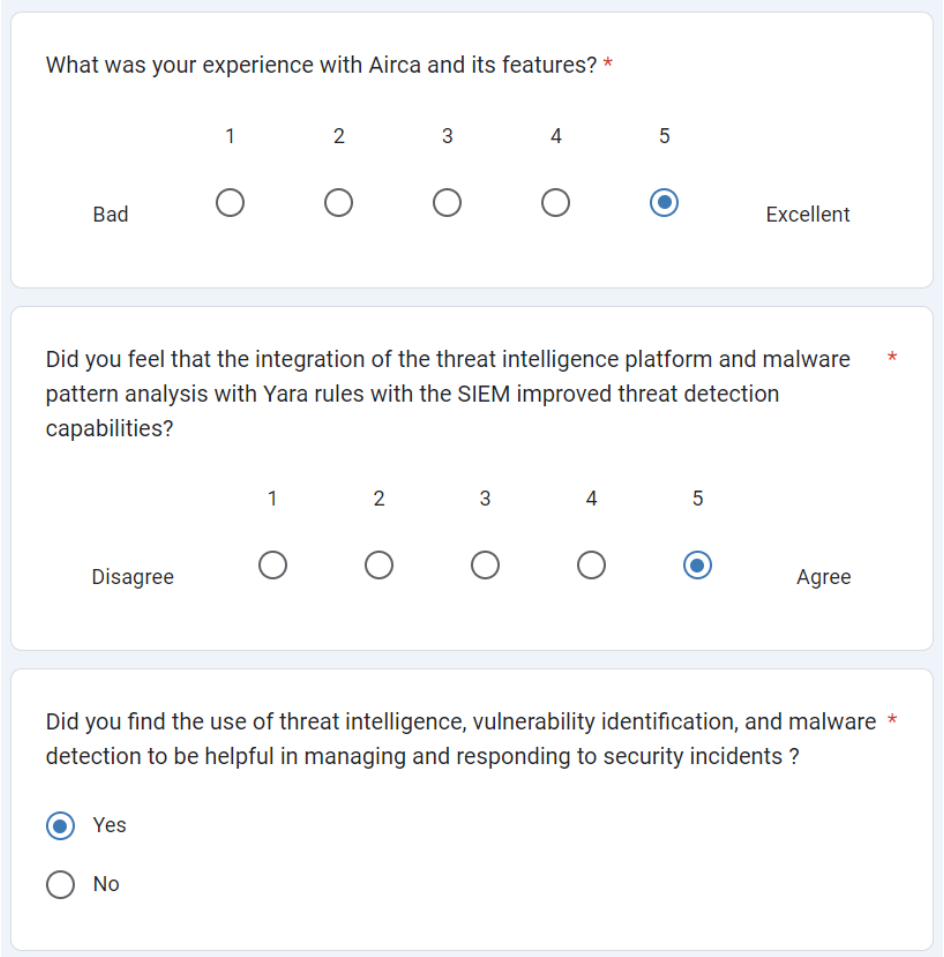


Figure 120 Post-Survey Sample Feedback: Part 1

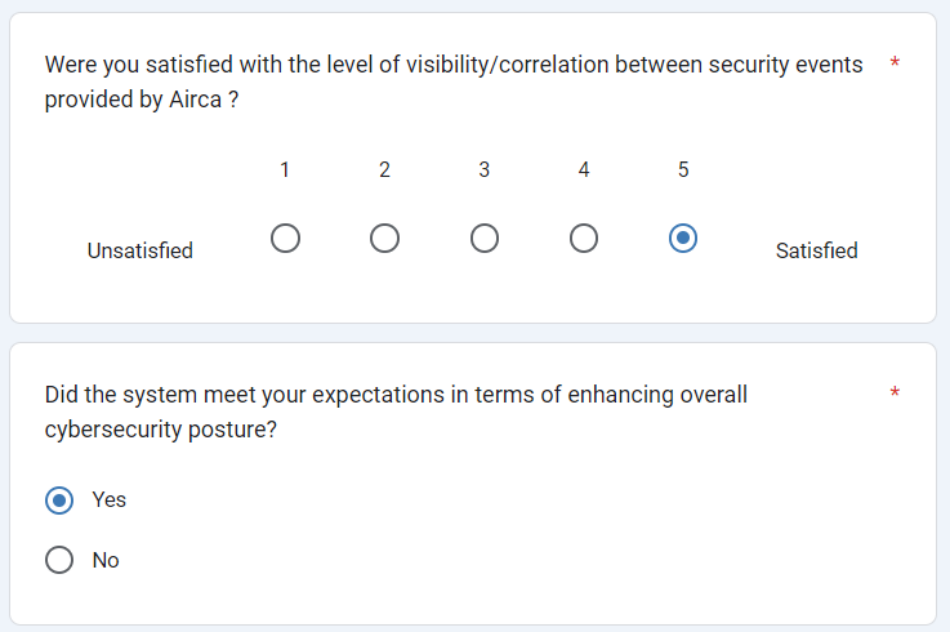


Figure 121 Post-Survey Sample Feedback: Part 2

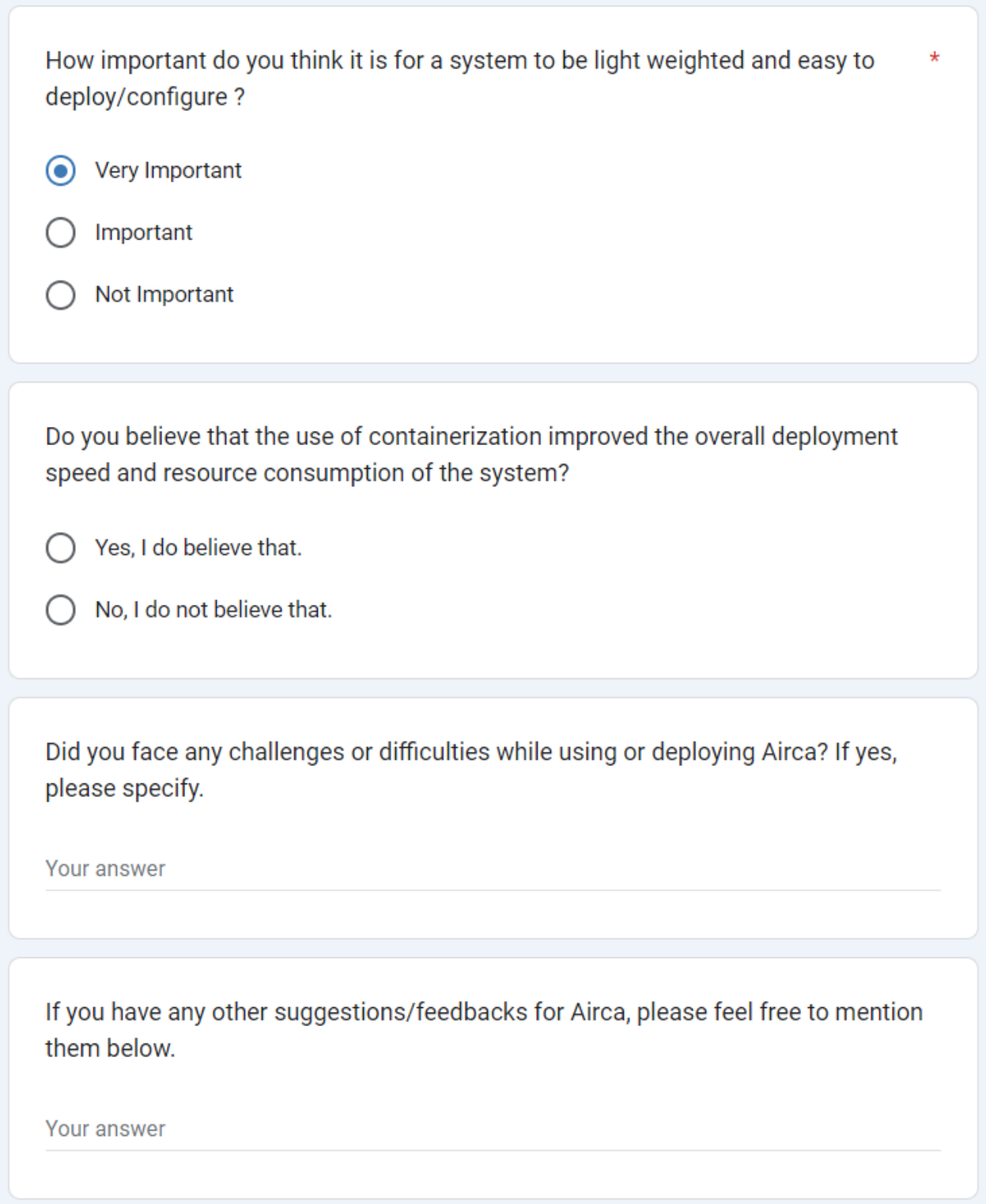


Figure 122 Post-Survey Sample Feedback: Part 3

9.3.3. Post-Survey Responses

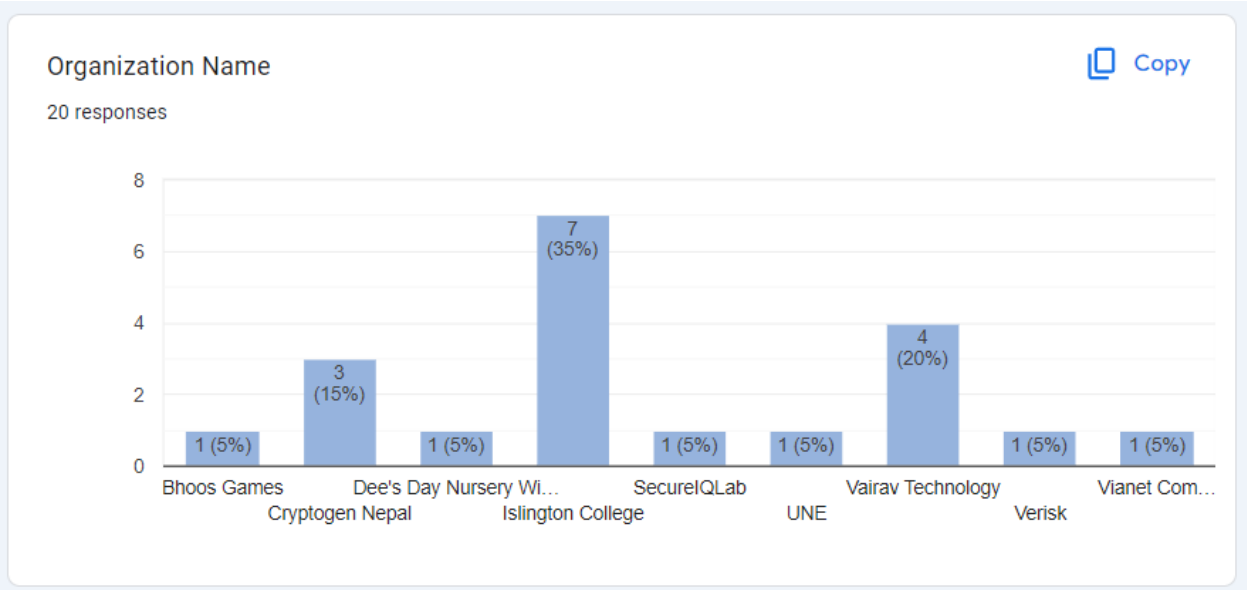


Figure 123 Post-Survey Response: Organizations

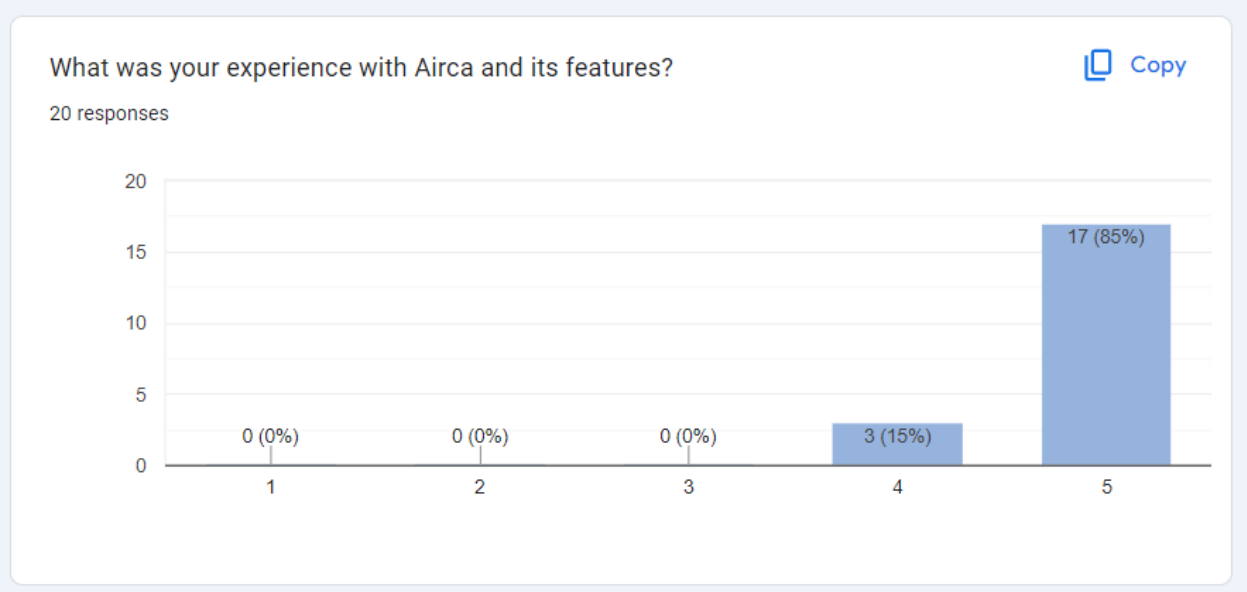


Figure 124 Post-Survey Response: Question 1

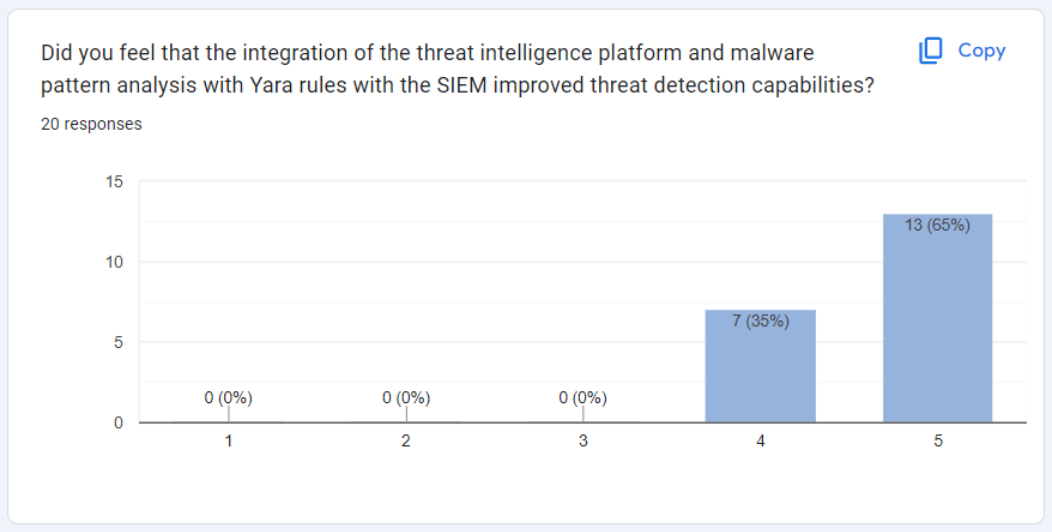


Figure 125 Post-Survey Response: Question 2

Figure 126 Post-Survey Response: Question 3

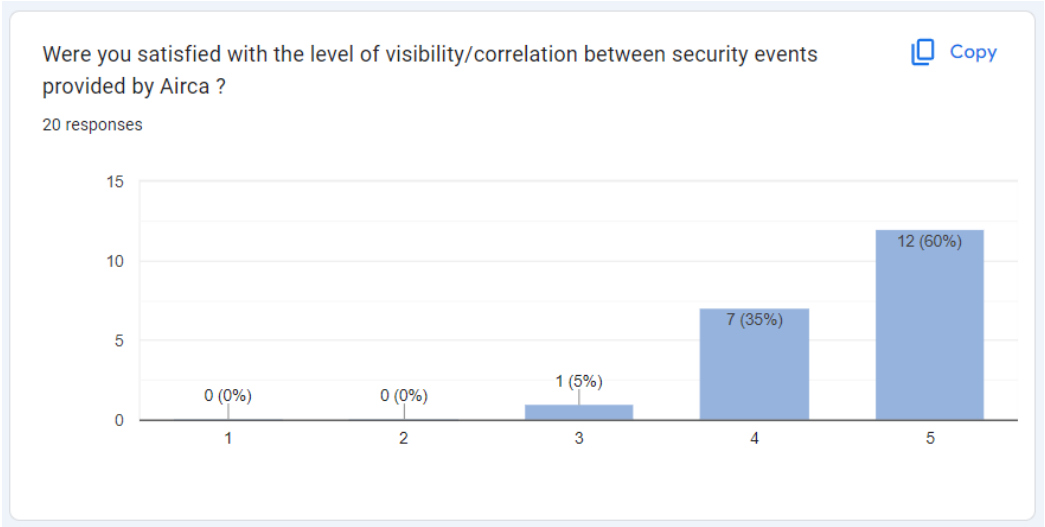


Figure 127 Post-Survey Response: Question 4

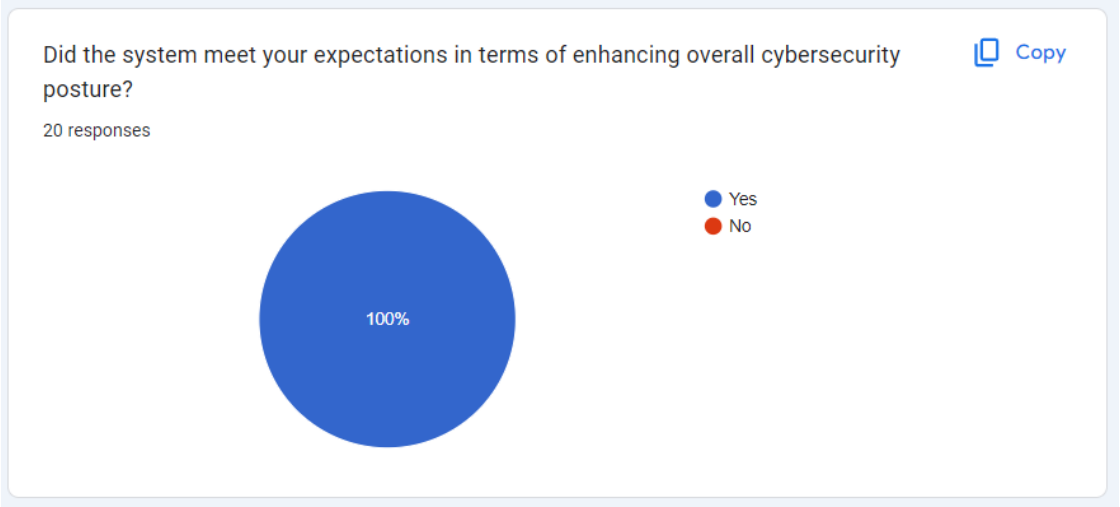


Figure 128 Post-Survey Response: Question 5

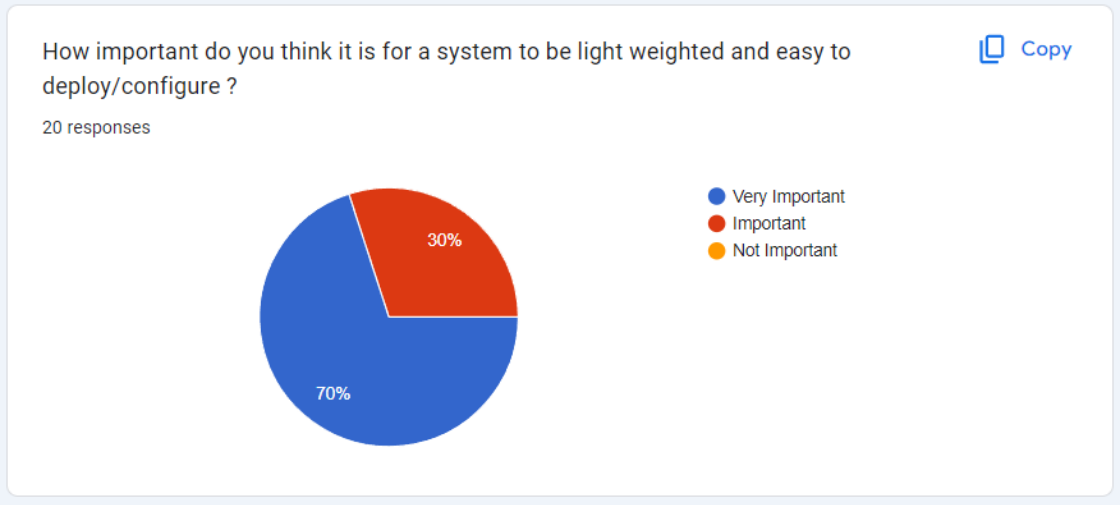


Figure 129 Post-Survey Response: Question 6

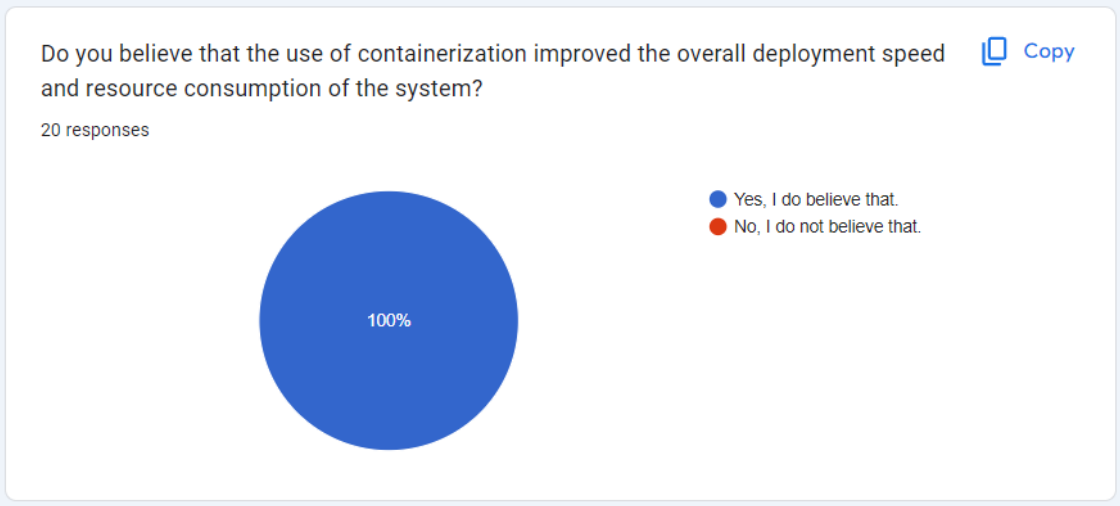


Figure 130 Post-Survey Response: Question 7

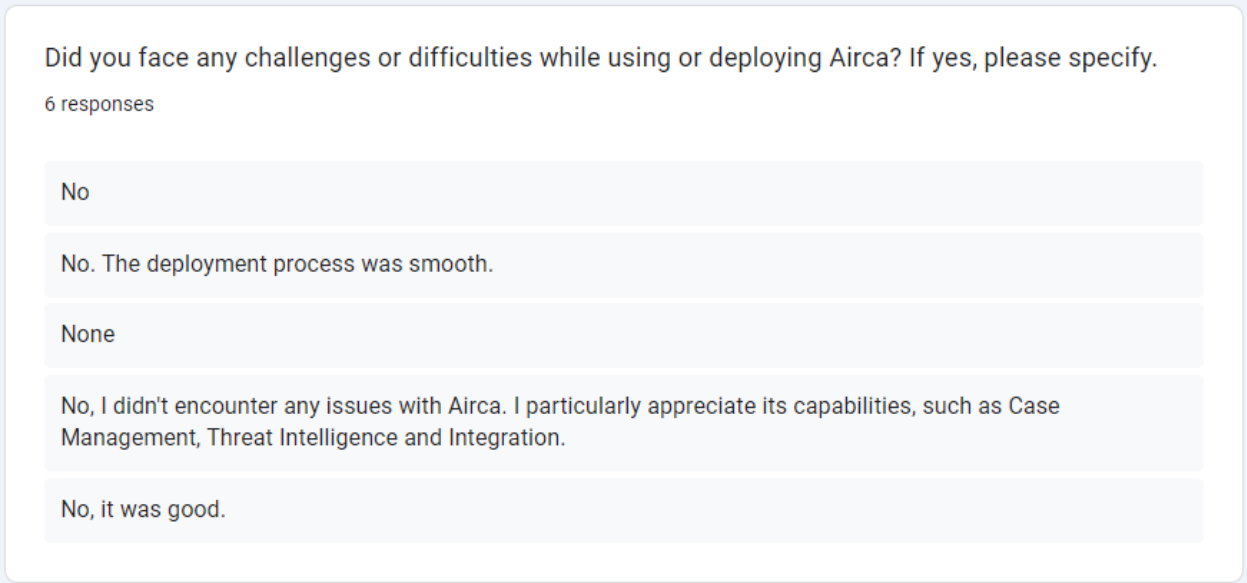


Figure 131 Post-Survey Response: Question 8

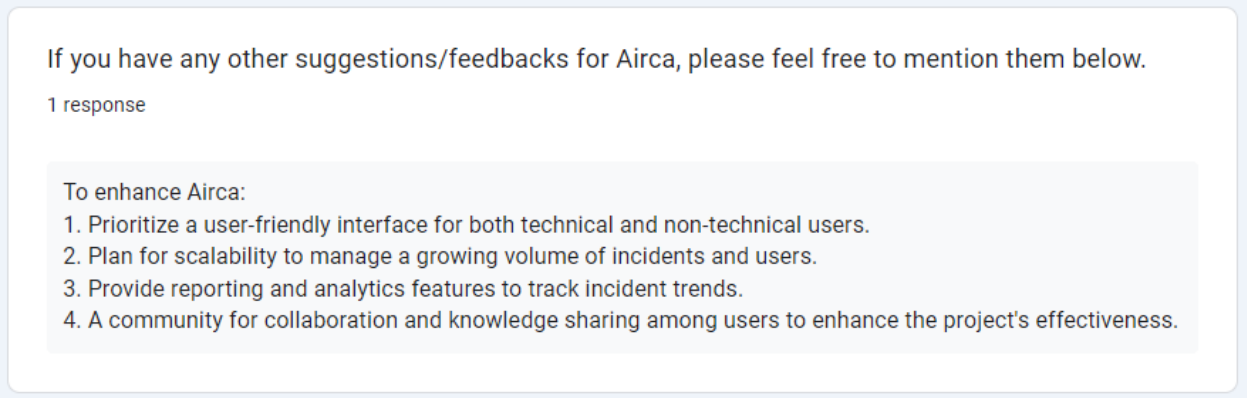


Figure 132 Post-Survey Response: Question 9

9.4. Appendix D: System Development Phase Evidence

9.4.1. First Iteration - Selection of required tools and system resources

9.4.1.1. Machines Resource Information

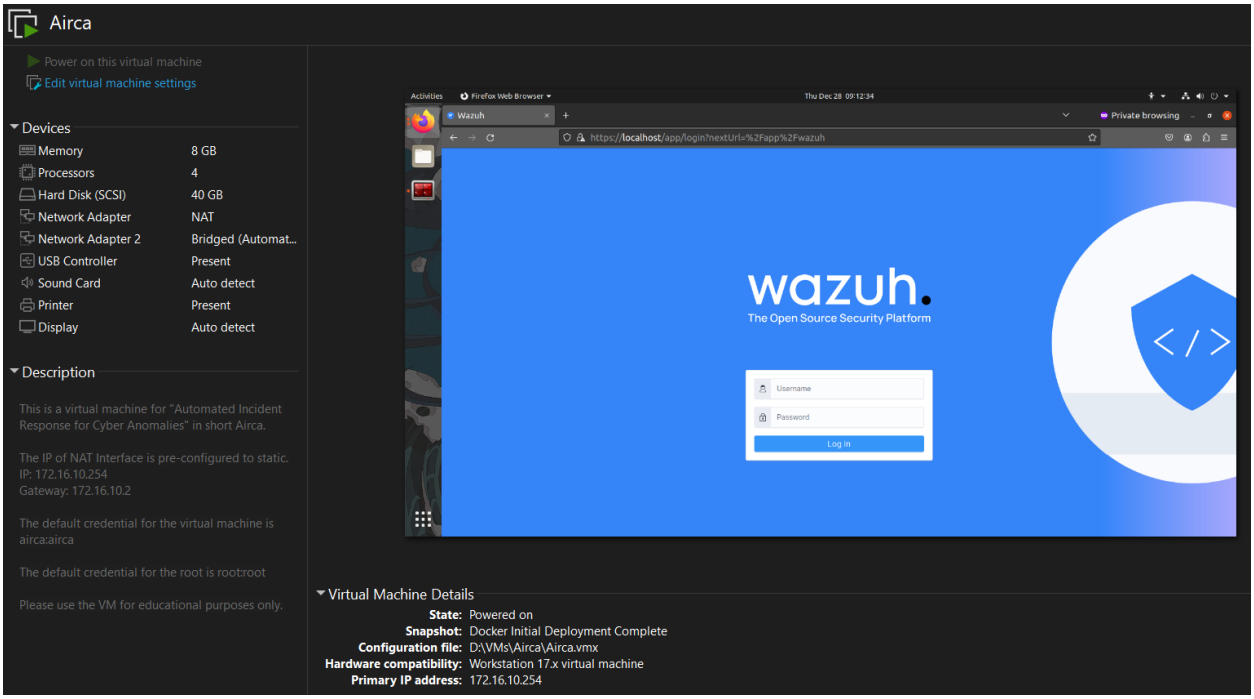


Figure 133 AIRCA's Machine Information

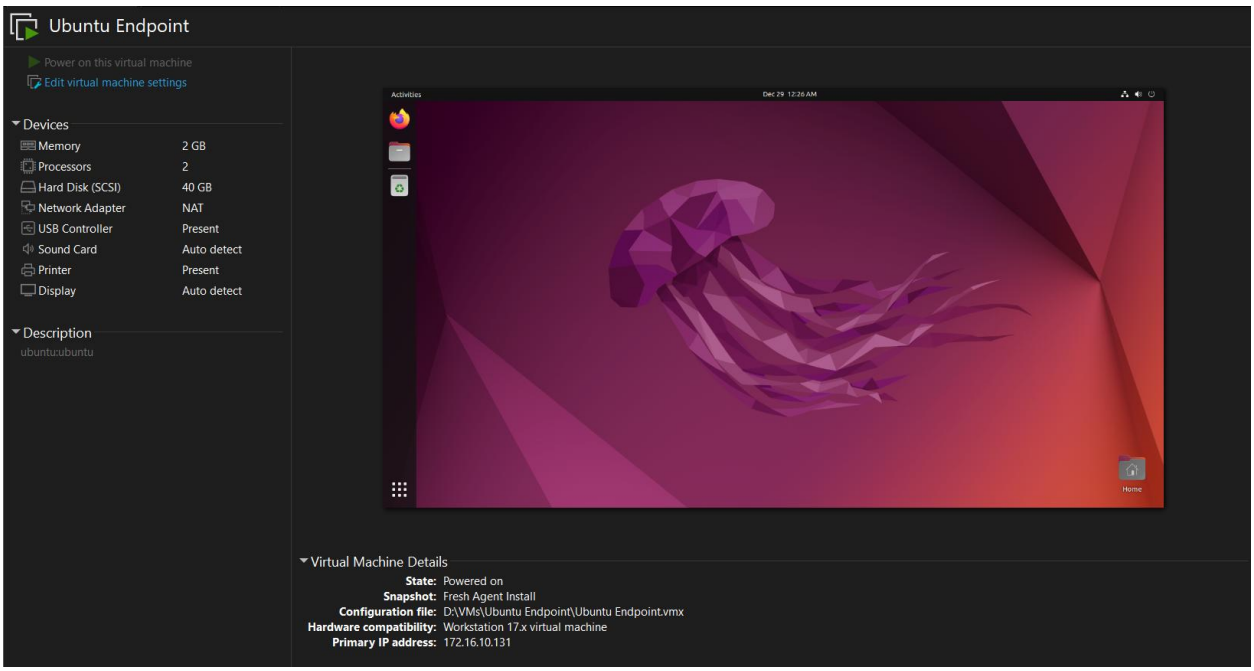


Figure 134 Ubuntu Endpoint's Information

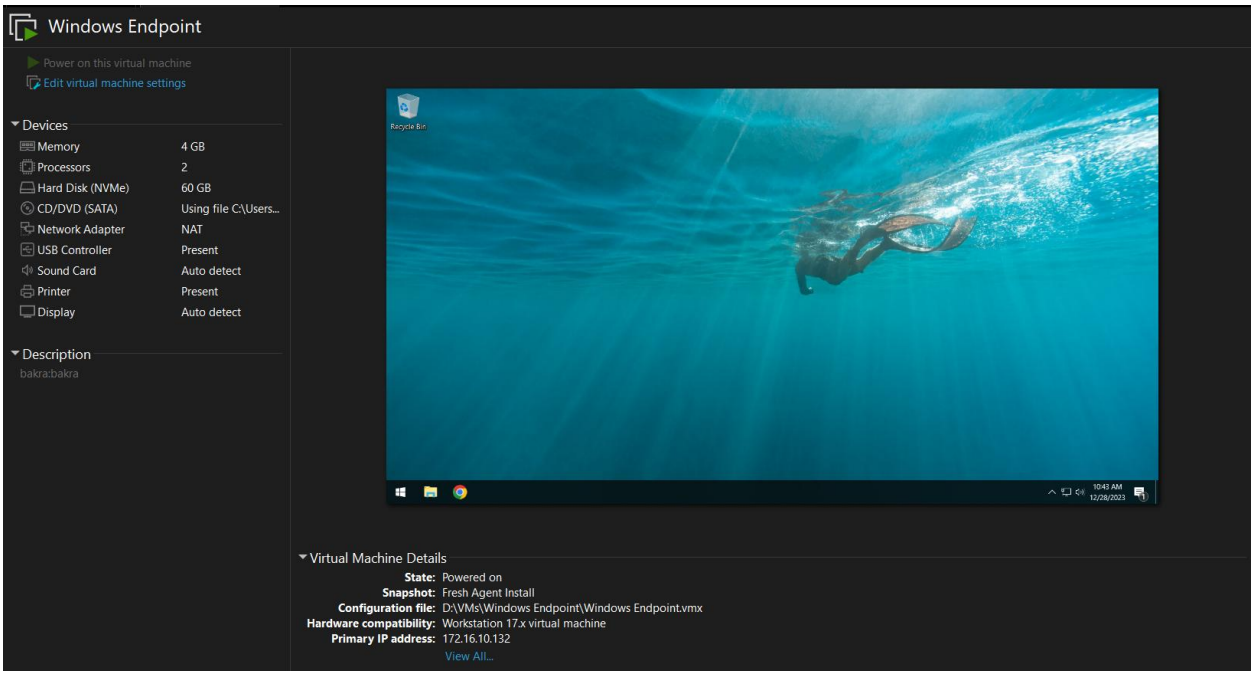


Figure 135 Windows Endpoint's Information

9.4.2. Second Iteration - Deployment of Wazuh and MISP

9.4.2.1. Wazuh Dashboard Overview

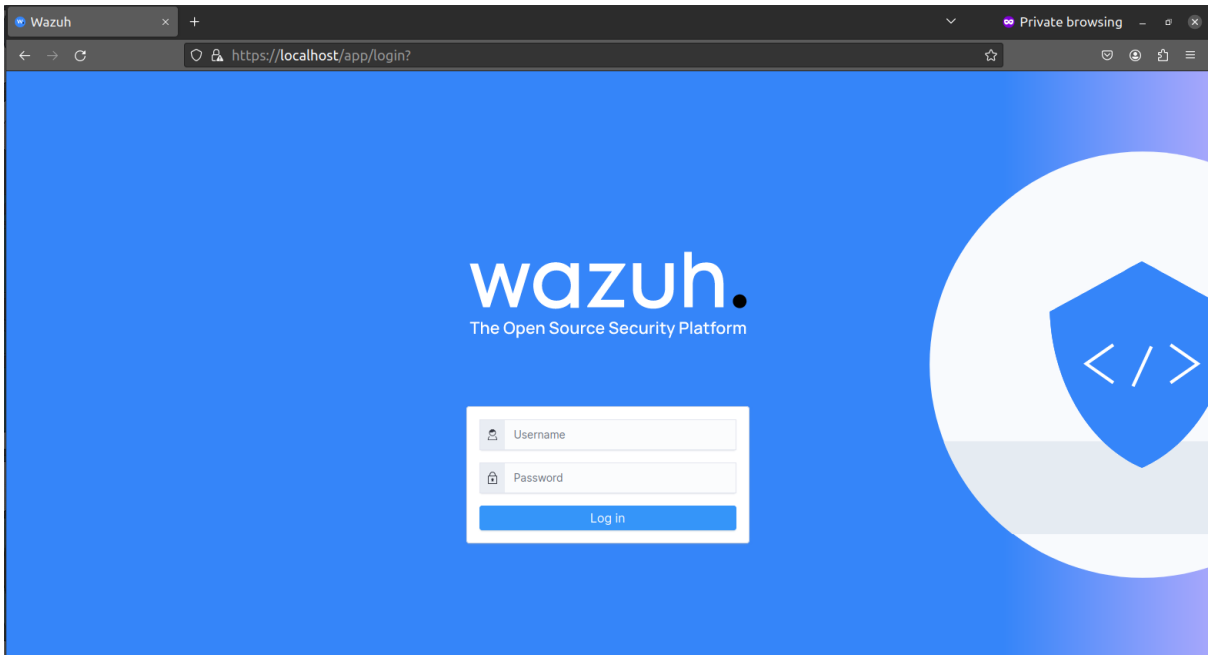


Figure 136 Navigating Wazuh Login Page

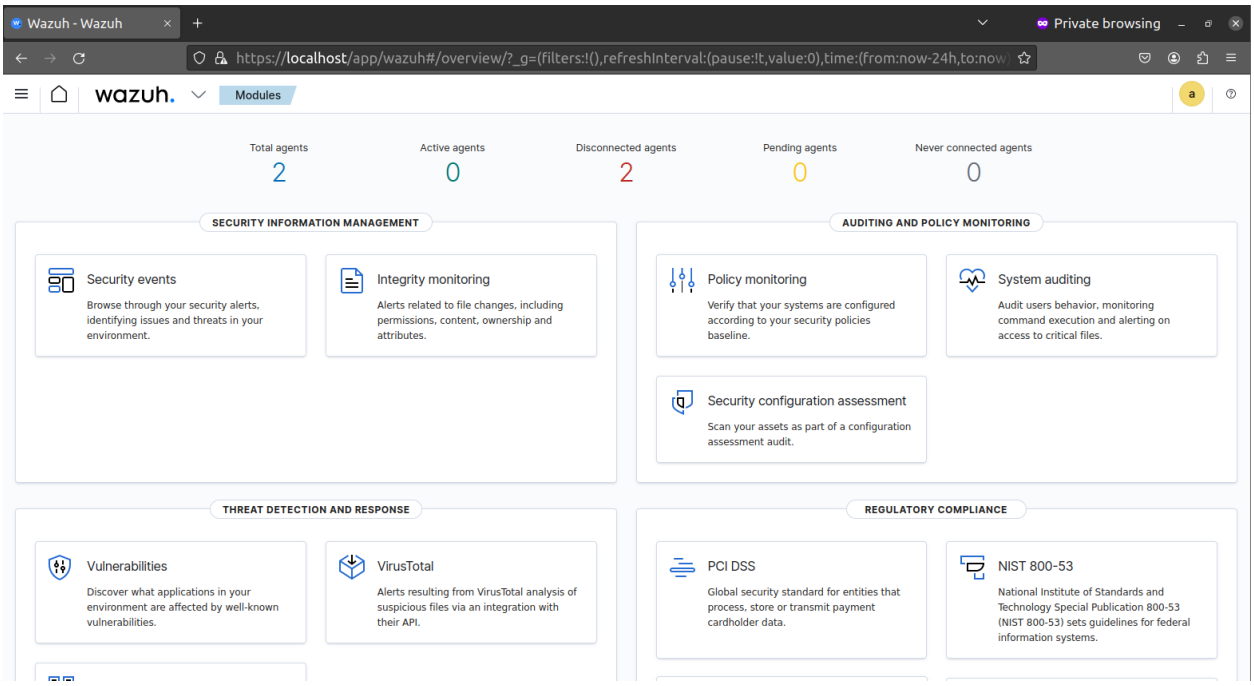


Figure 137 Wazuh Modules Overview

9.4.2.2. MISP Dashboard Overview

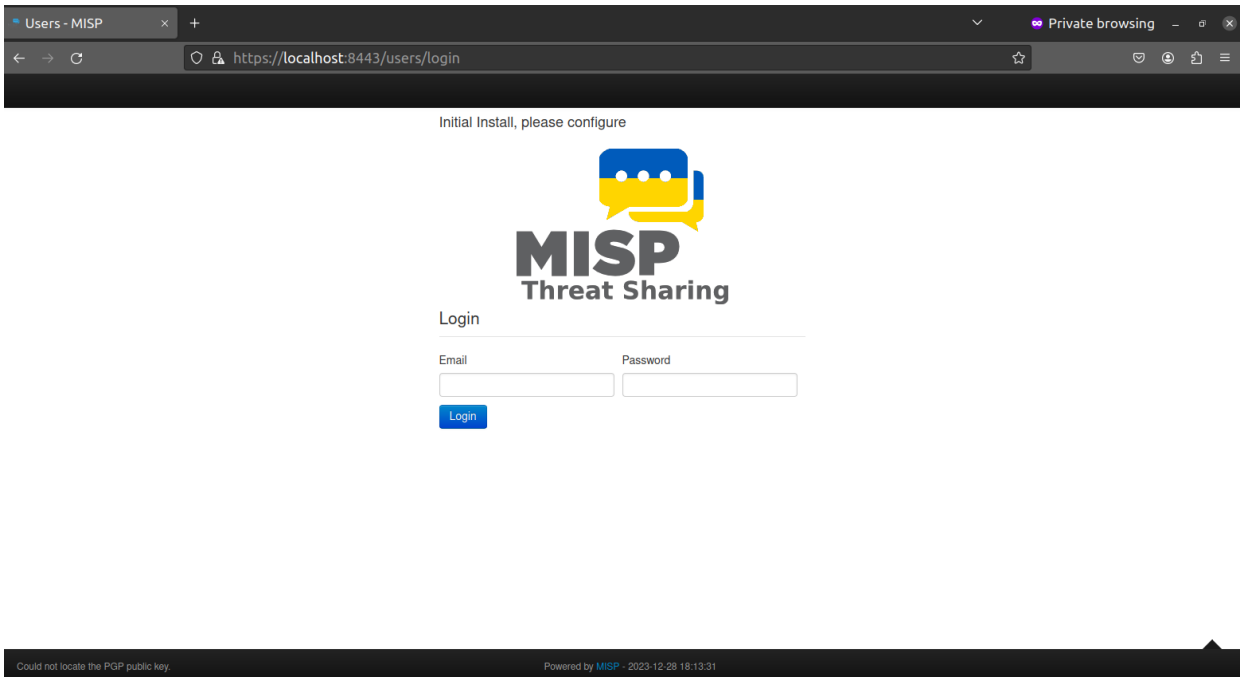


Figure 138 Navigating MISP Login Page

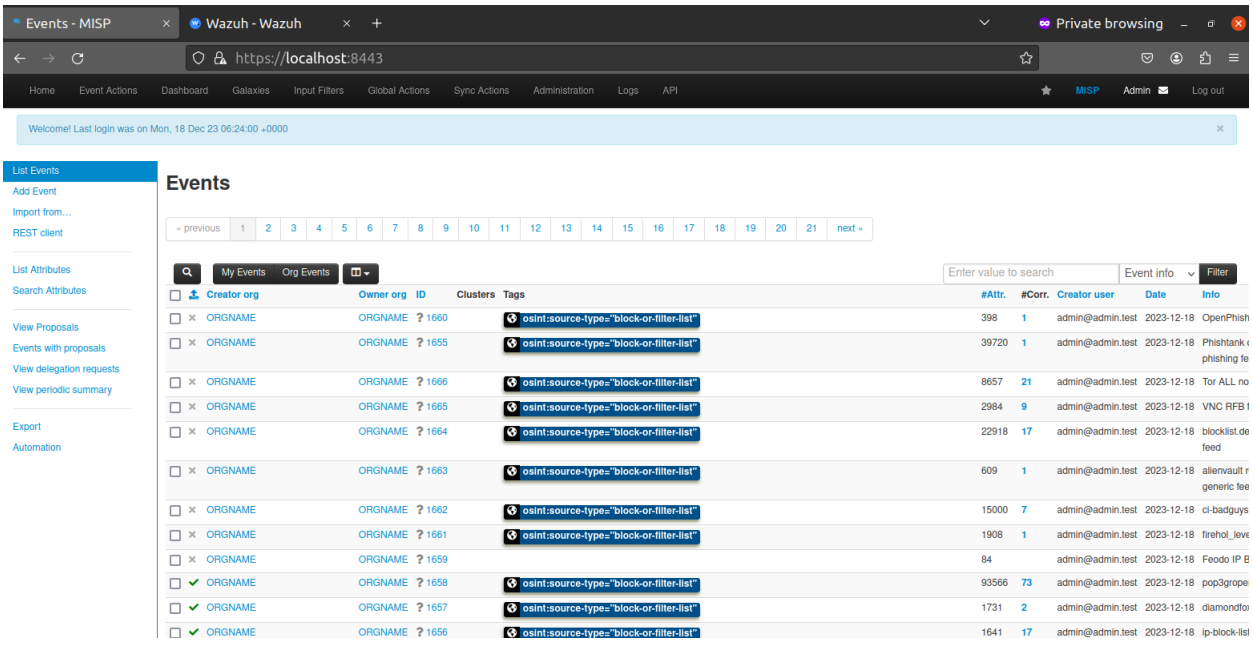


Figure 139 MISP Events

9.4.2.3. Agent Installation Process for Endpoints

9.4.2.3.1. Windows Endpoint Agent Installation

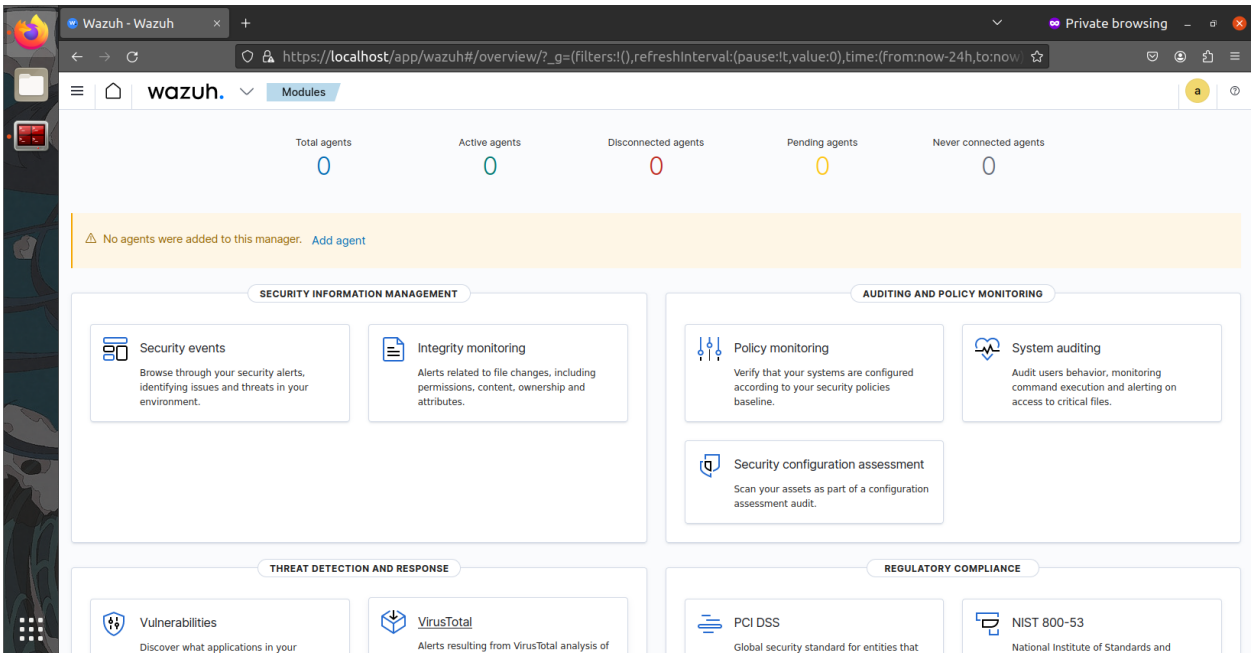


Figure 140 Adding an agent

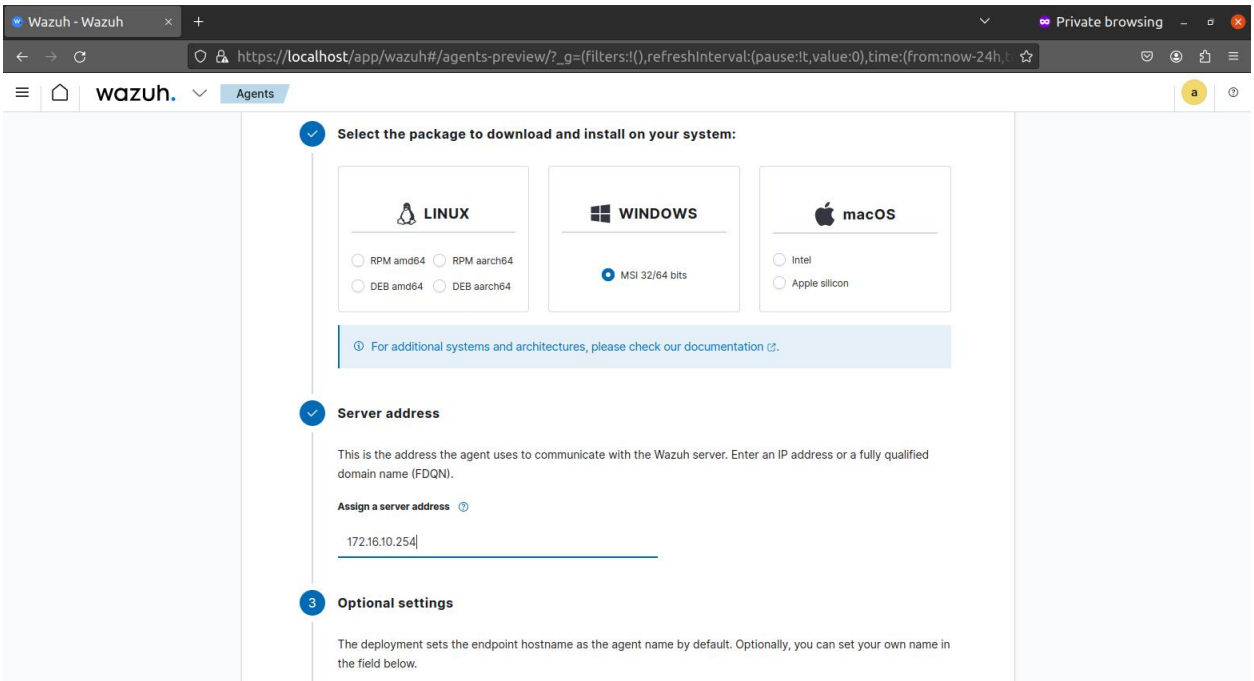


Figure 141 Setting up server address for windows agent

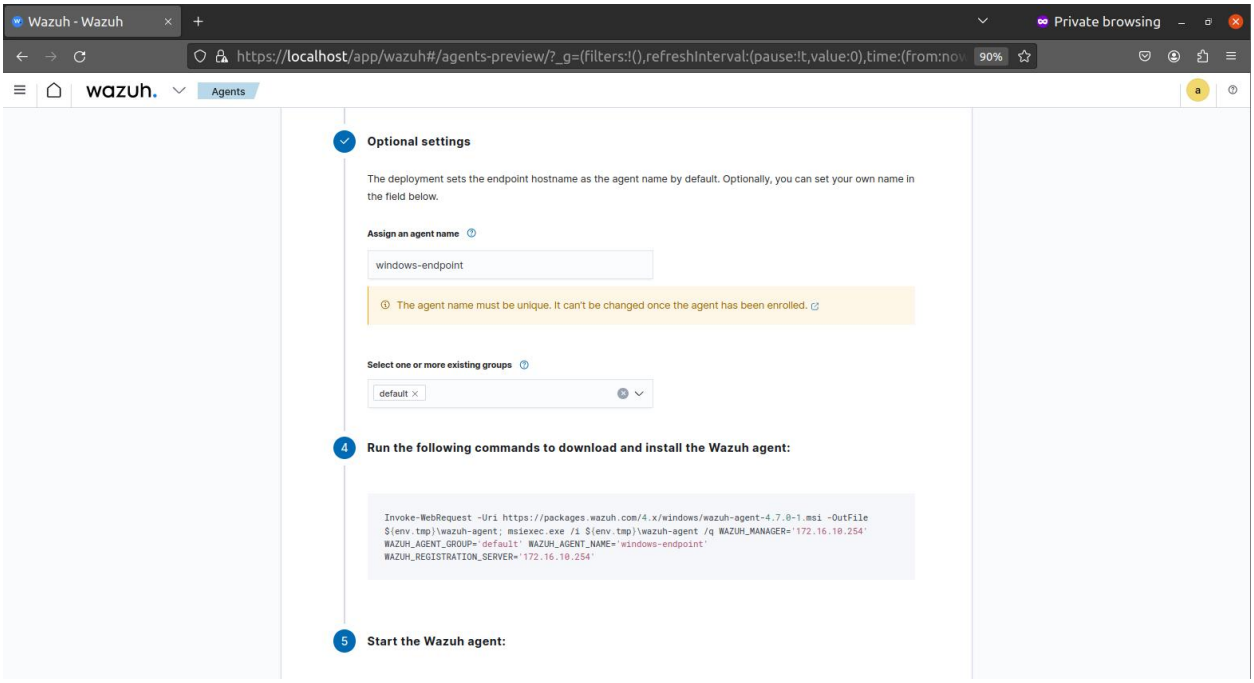


Figure 142 Assigning name to windows agent

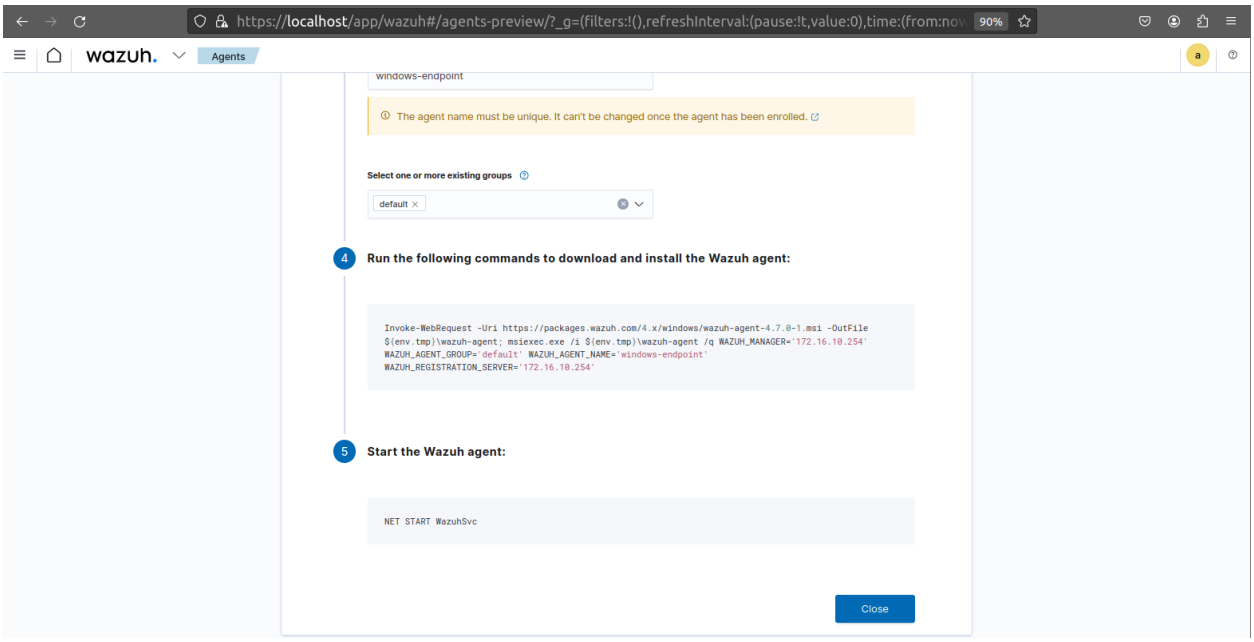


Figure 143 Commands to download, install and start the windows agent

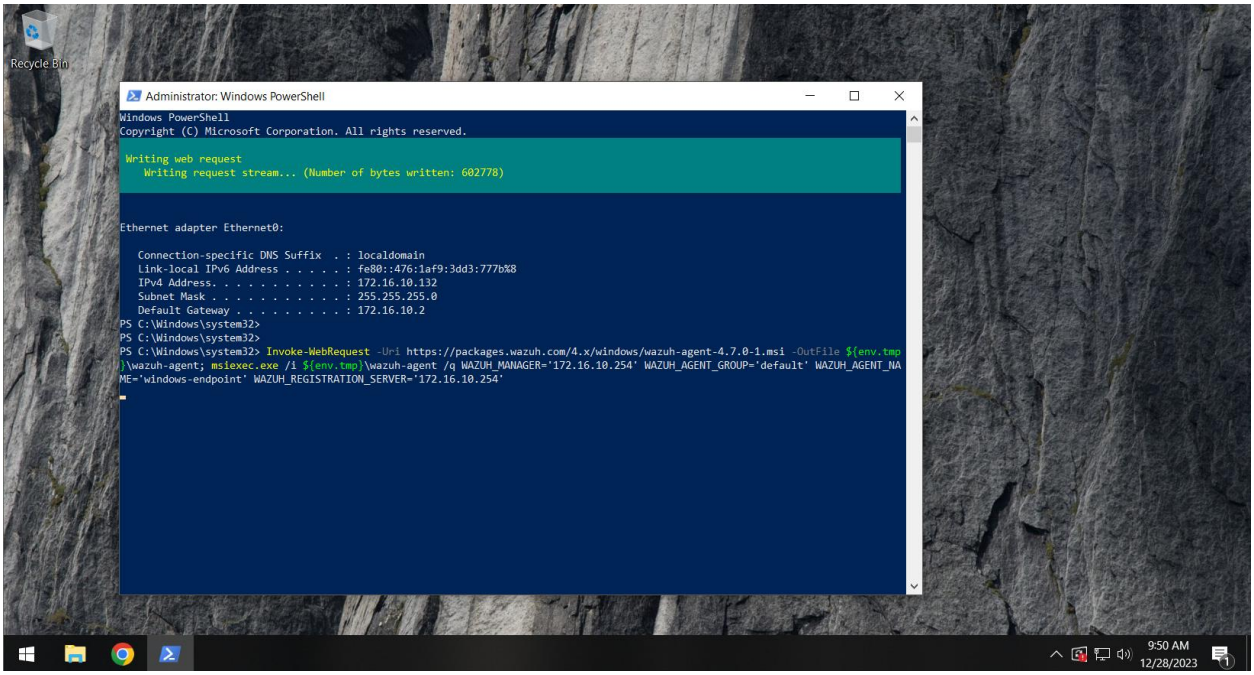


Figure 144 Running the command to download and install the windows agent

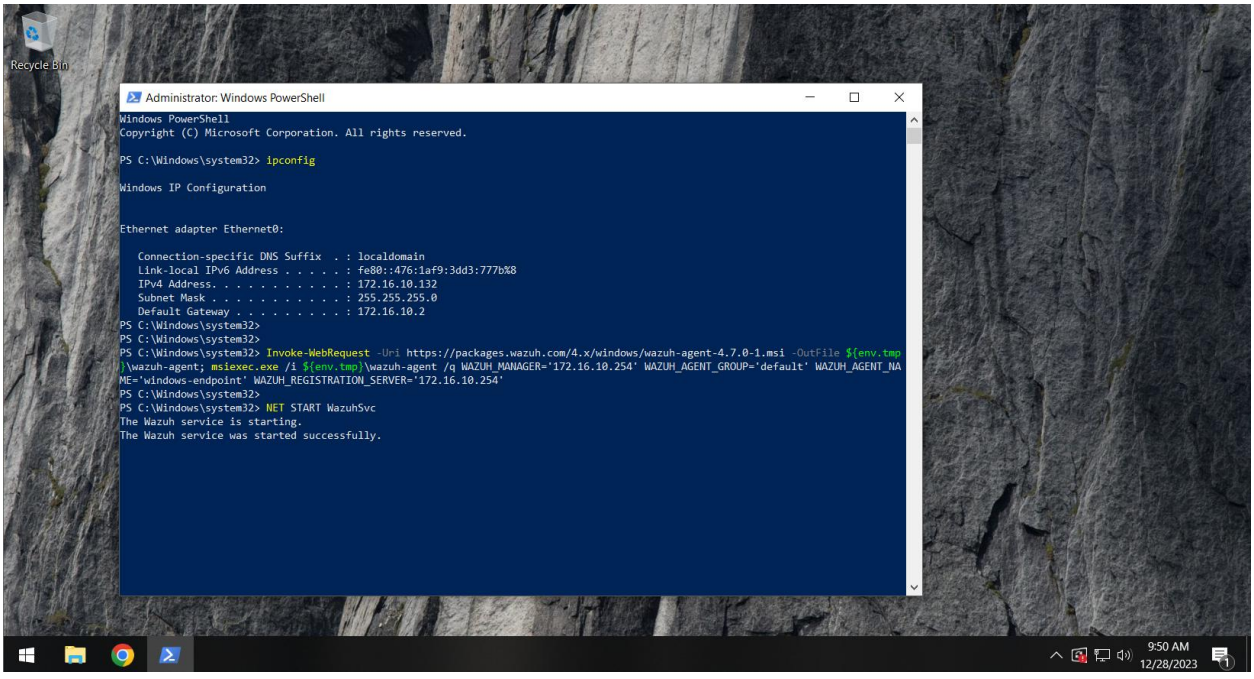


Figure 145 Starting the windows agent

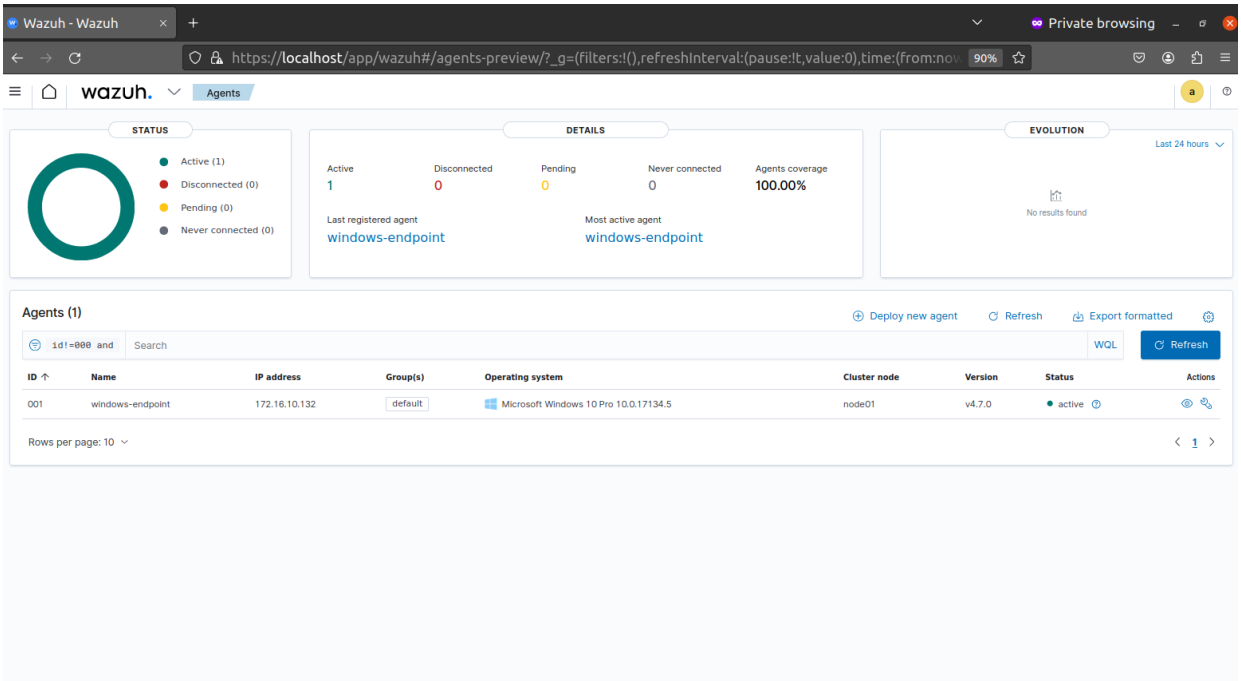


Figure 146 Viewing Windows Agent in Wazuh

9.4.2.3.2. Ubuntu Endpoint Agent Installation

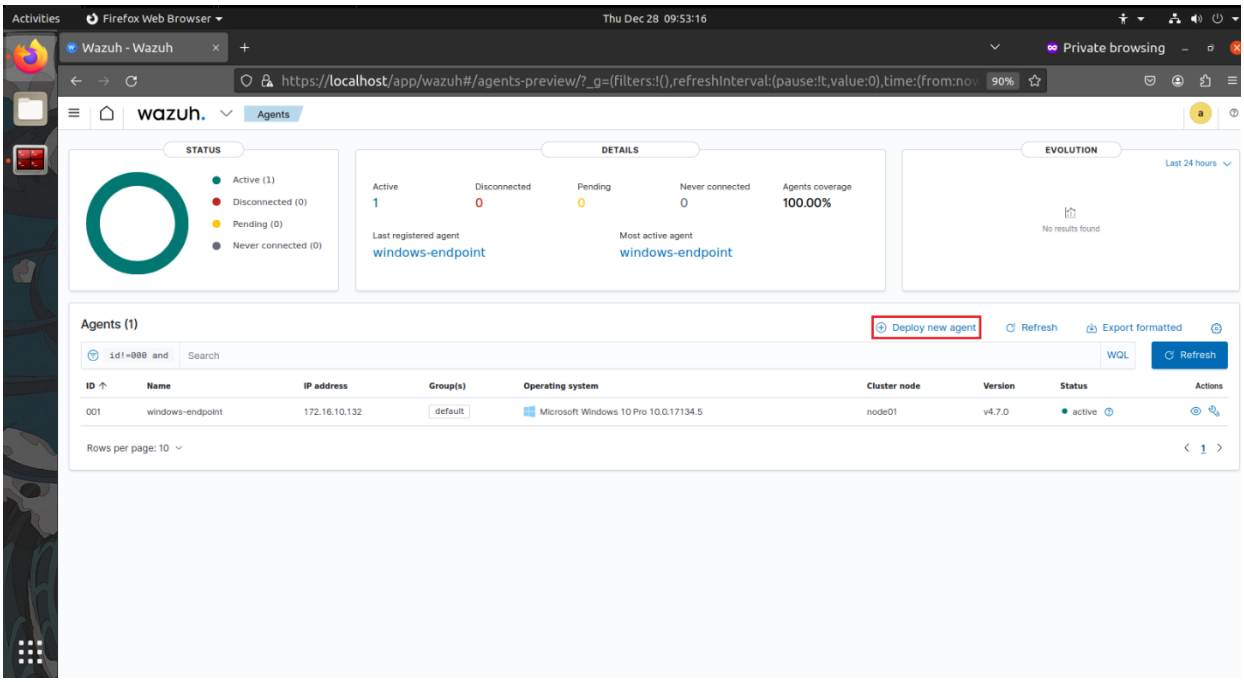


Figure 147 Deploying new agent for Ubuntu

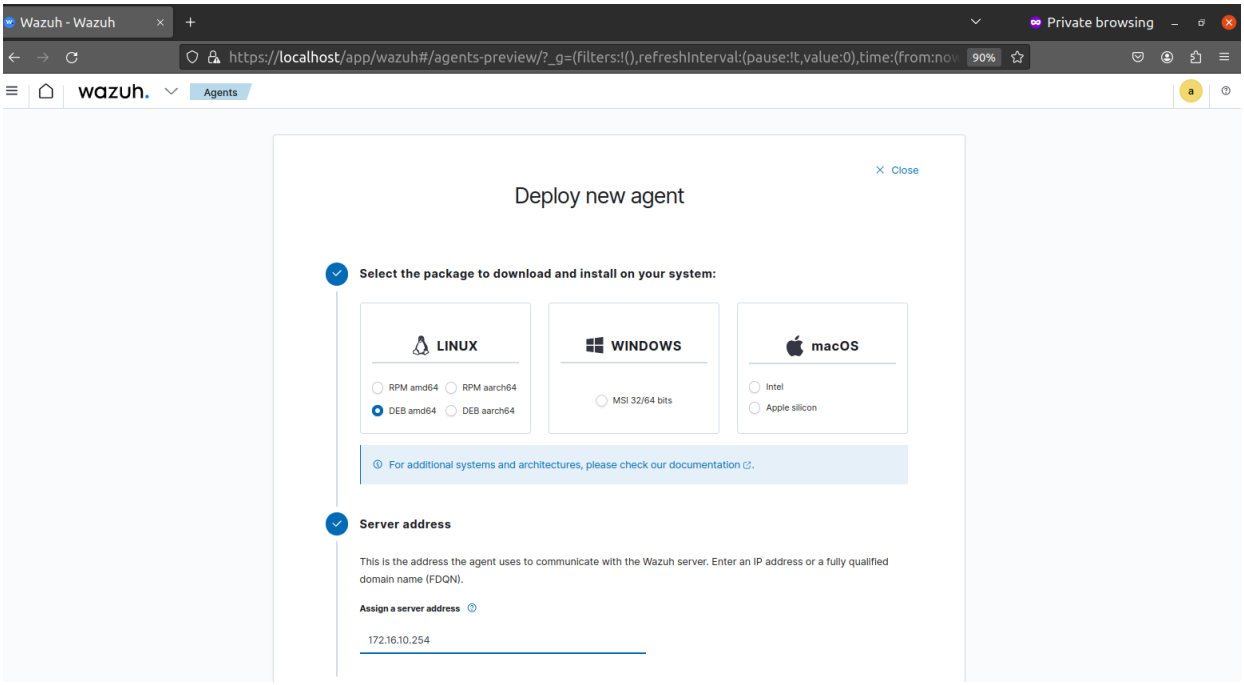


Figure 148 Setting up server address for ubuntu agent

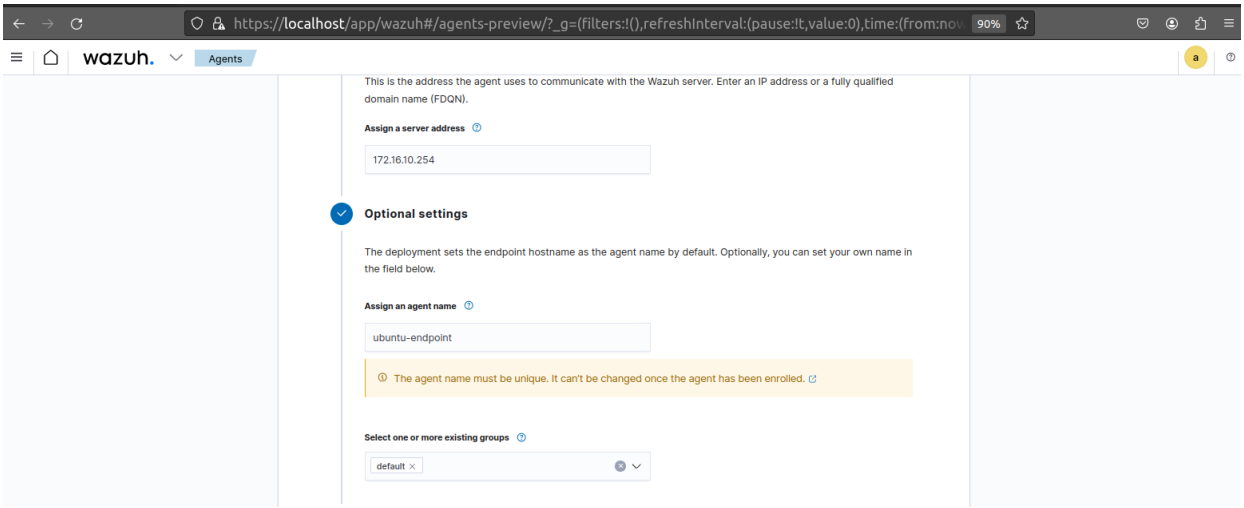


Figure 149 Assigning name for ubuntu agent

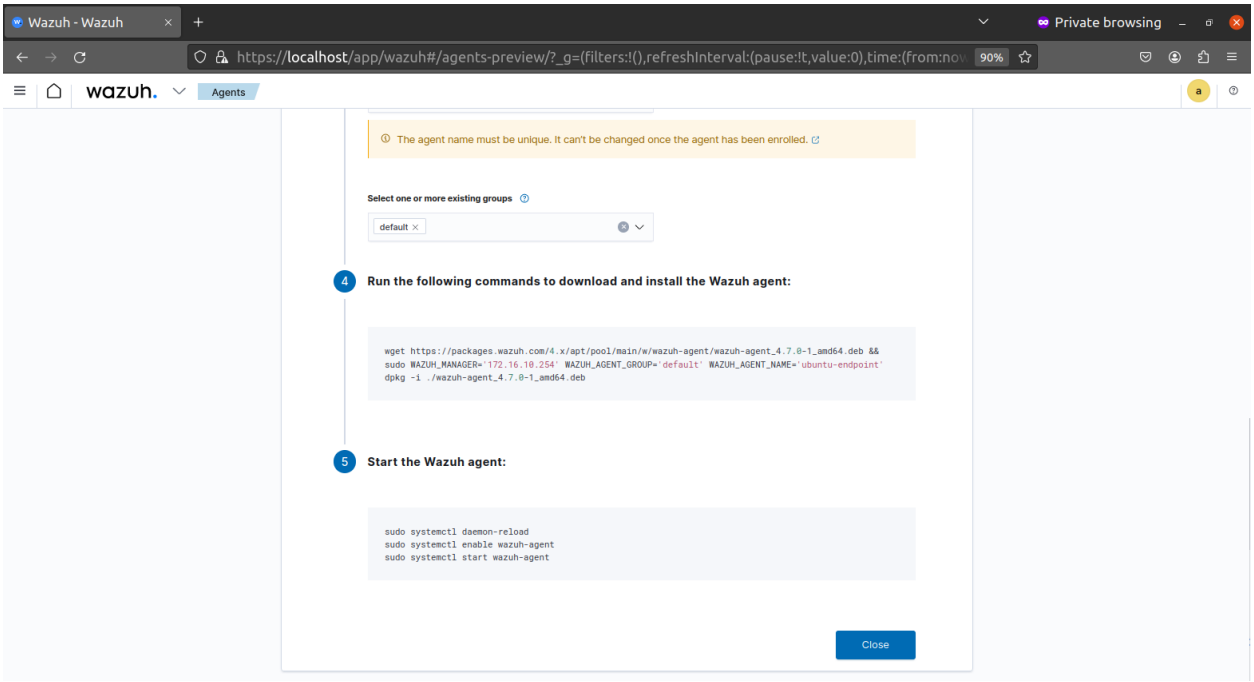


Figure 150 Commands to download, install and start the ubuntu agent

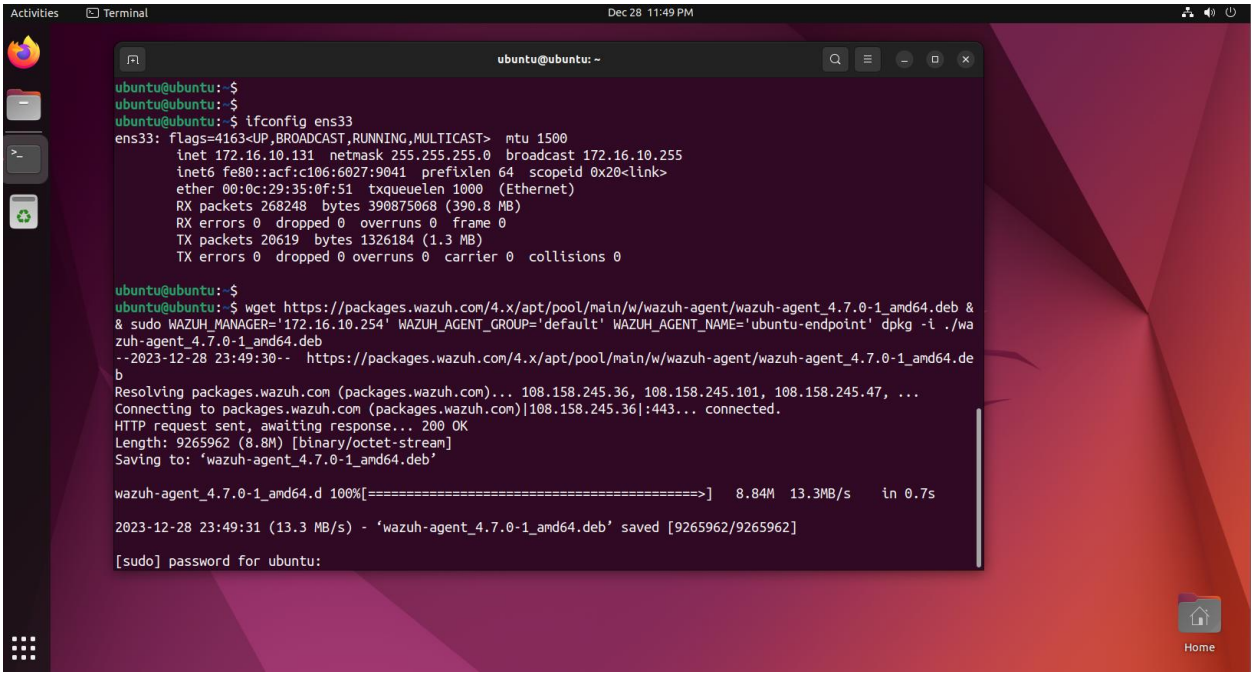


Figure 151 Running the commands to download and install the ubuntu agent

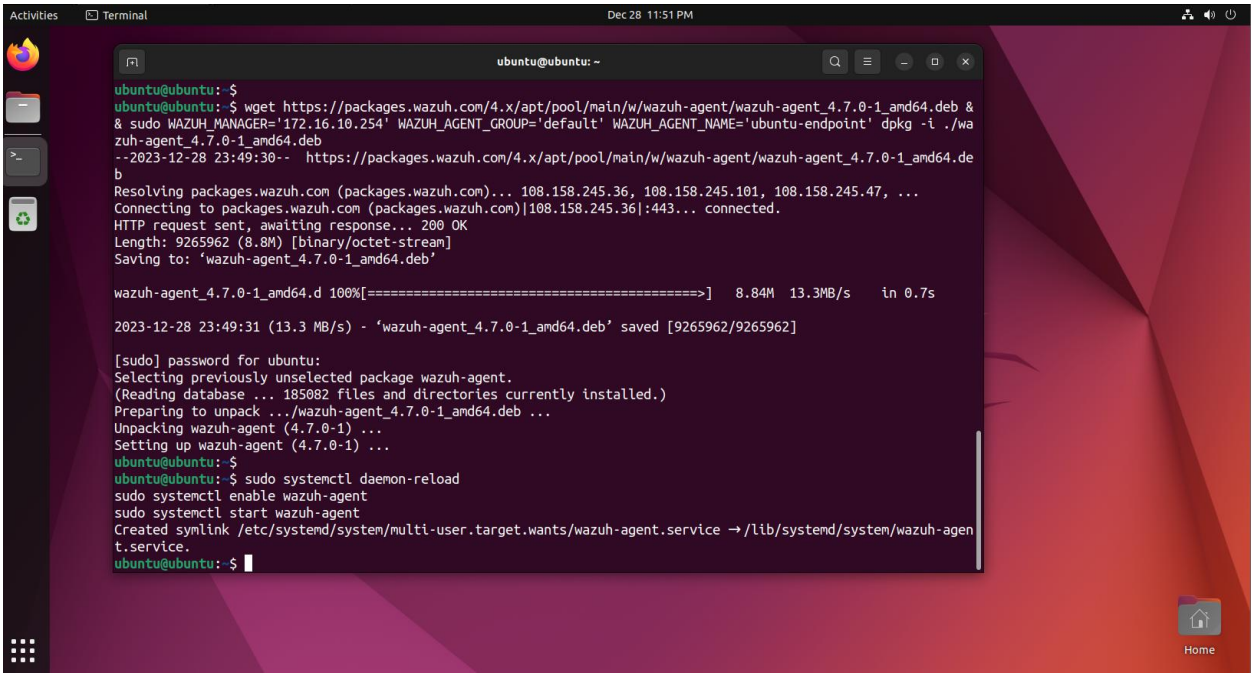


Figure 152 Starting the ubuntu agent

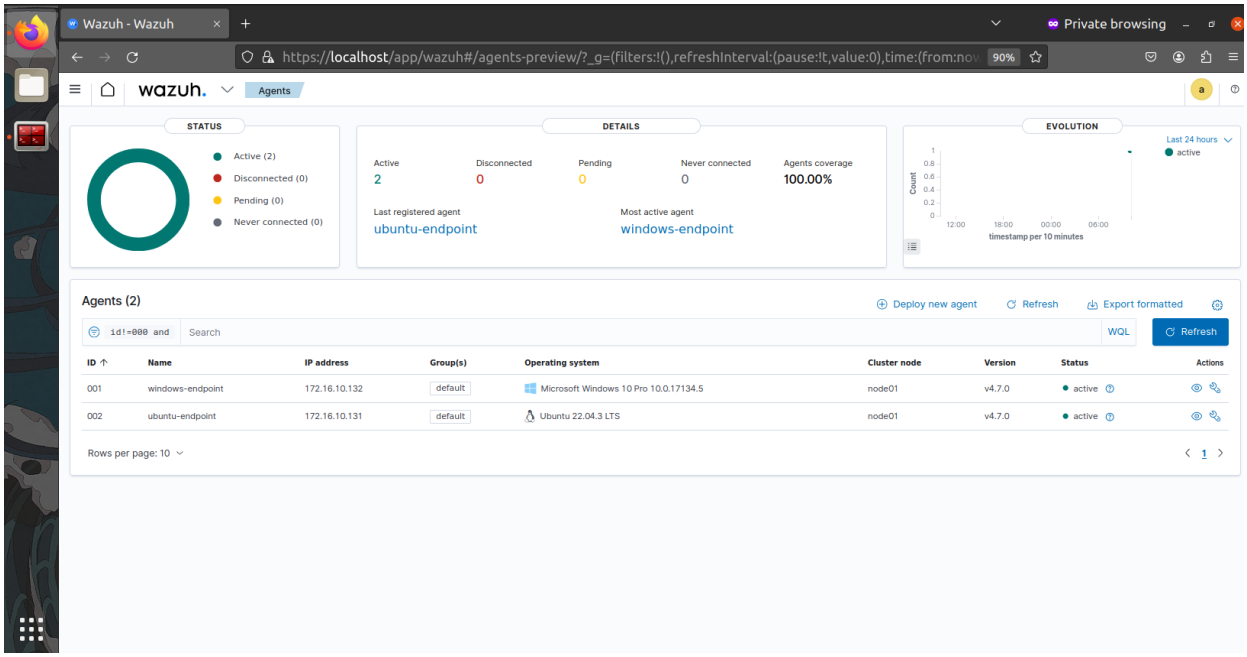


Figure 153 Viewing Ubuntu Agent in Wazuh

9.4.3. Third Iteration - Integration of Wazuh and MISP

9.4.3.1. Docker Compose Configuration

```
version: '3.7'

services:
  wazuh.manager:
    image: wazuh/wazuh-manager:4.7.0
    build: wazuh/.
    hostname: wazuh.manager
    container_name: airca_wazuh_manager
    restart: always
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 655360
        hard: 655360
    ports:
      - "1514:1514"
      - "1515:1515"
      - "514:514/udp"
      - "55000:55000"
    environment:
      - INDEXER_URL=https://wazuh.indexer:9200
      - INDEXER_USERNAME=admin
      - INDEXER_PASSWORD=SecretPassword
      - FILEBEAT_SSL_VERIFICATION_MODE=full
      - SSL_CERTIFICATE_AUTHORITIES=/etc/ssl/root-ca.pem
      - SSL_CERTIFICATE=/etc/ssl/filebeat.pem
      - SSL_KEY=/etc/ssl/filebeat.key
      - API_USERNAME=wazuh-wui
      - API_PASSWORD=MyS3cr37P450r.*-
    volumes:
      - wazuh_api_configuration:/var/ossec/api/configuration
      - wazuh_etc:/var/ossec/etc
      - wazuh_logs:/var/ossec/logs
      - wazuh_queue:/var/ossec/queue
      - wazuh_var_multigroups:/var/ossec/var/multigroups
      - wazuh_integrations:/var/ossec/integrations
      - wazuh_active_response:/var/ossec/active-response/bin
      - wazuh_agentless:/var/ossec/agentless
      - wazuh_wodles:/var/ossec/wodles
      - filebeat_etc:/etc/filebeat
```

```
    - filebeat_var:/var/lib/filebeat
    - ./wazuh/config/wazuh_indexer_ssl_certs/root-ca-manager.pem:/etc/ssl/root-ca.pem
    - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.manager.pem:/etc/ssl/filebeat.pem
    - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.manager-key.pem:/etc/ssl/filebeat.key
    - ./wazuh/config/wazuh_cluster/wazuh_manager.conf:/wazuh-config-mount/etc/ossec.conf


  wazuh.indexer:
    image: wazuh/wazuh-indexer:4.7.0
    hostname: wazuh.indexer
    container_name: airca_wazuh_indexer
    restart: always
    ports:
      - "9200:9200"
    environment:
      - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - wazuh-indexer-data:/var/lib/wazuh-indexer
      - ./wazuh/config/wazuh_indexer_ssl_certs/root-ca.pem:/usr/share/wazuh-indexer/certs/root-
ca.pem
      - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.indexer-key.pem:/usr/share/wazuh-
indexer/certs/wazuh.indexer.key
      - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.indexer.pem:/usr/share/wazuh-
indexer/certs/wazuh.indexer.pem
      - ./wazuh/config/wazuh_indexer_ssl_certs/admin.pem:/usr/share/wazuh-
indexer/certs/admin.pem
      - ./wazuh/config/wazuh_indexer_ssl_certs/admin-key.pem:/usr/share/wazuh-
indexer/certs/admin-key.pem
      - ./wazuh/config/wazuh_indexer/wazuh.indexer.yml:/usr/share/wazuh-
indexer/opensearch.yml
      - ./wazuh/config/wazuh_indexer/internal_users.yml:/usr/share/wazuh-indexer/opensearch-
security/internal_users.yml


  wazuh.dashboard:
    image: wazuh/wazuh-dashboard:4.7.0
    hostname: wazuh.dashboard
    container_name: airca_wazuh_dashboard
    restart: always
    ports:
```

```
    - 443:5601
  environment:
    - INDEXER_USERNAME=admin
    - INDEXER_PASSWORD=SecretPassword
    - WAZUH_API_URL=https://wazuh.manager
    - DASHBOARD_USERNAME=kibanaserver
    - DASHBOARD_PASSWORD=kibanaserver
    - API_USERNAME=wazuh-wui
    - API_PASSWORD=MyS3cr37P450r.*-
  volumes:
    - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.dashboard.pem:/usr/share/wazuh-
dashboard/certs/wazuh-dashboard.pem
    - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.dashboard-key.pem:/usr/share/wazuh-
dashboard/certs/wazuh-dashboard-key.pem
    - ./wazuh/config/wazuh_indexer_ssl_certs/root-ca.pem:/usr/share/wazuh-
dashboard/certs/root-ca.pem
    - ./wazuh/config/wazuh_dashboard/opensearch_dashboards.yml:/usr/share/wazuh-
dashboard/config/opensearch_dashboards.yml
    - ./wazuh/config/wazuh_dashboard/wazuh.yml:/usr/share/wazuh-
dashboard/data/wazuh/config/wazuh.yml
    - ./wazuh/files/logo/Only-AIRCA-WHITE.svg:/usr/share/wazuh-
dashboard/plugins/wazuh/public/assets/custom/images/Only-AIRCA-WHITE.svg
    - wazuh-dashboard-config:/usr/share/wazuh-dashboard/data/wazuh/config
    - wazuh-dashboard-custom:/usr/share/wazuh-dashboard/plugins/wazuh/public/assets/custom
  depends_on:
    - wazuh.indexer
  links:
    - wazuh.indexer:wazuh.indexer
    - wazuh.manager:wazuh.manager


 redis:
  image: redis:5.0.6
  container_name: airca_misp_redis


 db:
  image: mysql:8.0.19
  container_name: airca_misp_db
  command: --default-authentication-plugin=mysql_native_password
  restart: always
  environment:
    - "MYSQL_USER=misp"
    - "MYSQL_PASSWORD=password"
    - "MYSQL_ROOT_PASSWORD=password"
    - "MYSQL_DATABASE=misp"
  volumes:
```

```
    - mysql_data:/var/lib/mysql
  cap_add:
    - SYS_NICE  # CAP_SYS_NICE Prevent runaway mysql log


 misp:
  image: coolacid/misp-docker:core-latest
  build:
     context: misp/server/.
     args:
        - MISP_TAG=${MISP_TAG}
        - PHP_VER=${PHP_VER}

  container_name: airca_misp
  depends_on:
    - redis
    - db
  ports:
    - "8080:8080"
    - "8443:8443"
  volumes:
    - "./misp/server-configs/:/var/www/MISP/app/Config/"
    - "./misp/logs/:/var/www/MISP/app/tmp/logs/"
    - "./misp/files/:/var/www/MISP/app/files"
    - "./misp/ssl/:/etc/nginx/certs"
  environment:
    - "BASEURL=https://172.16.10.254:8443"
    - "REDIS_FQDN=redis"
    - "INIT=true"           # Initialze MISP, things includes, attempting to import SQL and the
Files DIR
    - "CRON_USER_ID=1"      # The MISP user ID to run cron jobs as
#    - "SYNCSERVERS=1 2 3 4"  # The MISP Feed servers to sync in the cron job

    # Database Configuration (And their defaults)
    - "MYSQL_HOST=airca_misp_db"
    - "MYSQL_USER=misp"
    - "MYSQL_PASSWORD=password"
    - "MYSQL_DATABASE=misp"

    # Optional Settings
#    - "NOREDIR=true" # Do not redirect port 80
    - "DISIPV6=true" # Disable IPV6 in nginx
#    - "CERTAUTH=optional" # Can be set to optional or on - Step 2 of
https://github.com/MISP/MISP/tree/2.4/app/Plugin/CertAuth is still required
#    - "SECURESSL=true" # Enable higher security SSL in nginx
#    - "MISP_MODULES_FQDN=http://misp-modules" # Set the MISP Modules FQDN, used
for Enrichment_services_url/Import_services_url/E    x_services_url
```

```
#     - "WORKERS=1" #If set to a value larger than 1 this will increase the number of parallel worker processes


  misp-modules:
    image: coolacid/misp-docker:modules-latest
    build:
       context: misp/modules/.
       args:
          - MODULES_TAG=${MODULES_TAG}

    container_name: airca_misp_modules
    depends_on:
      - redis
      - db
    environment:
      - "REDIS_BACKEND=redis"



volumes:
  wazuh_api_configuration:
  wazuh_etc:
  wazuh_logs:
  wazuh_queue:
  wazuh_var_multigroups:
  wazuh_integrations:
  wazuh_active_response:
  wazuh_agentless:
  wazuh_wodles:
  filebeat_etc:
  filebeat_var:
  wazuh-indexer-data:
  wazuh-dashboard-config:
  wazuh-dashboard-custom:
  mysql_data:
```

9.4.3.2. Custom MISP python script

```
#!/var/ossec/framework/python/bin/python3
## MISP API Integration
#
import sys
import os
from socket import socket, AF_UNIX, SOCK_DGRAM
from datetime import date, datetime, timedelta
import time
import requests
from requests.exceptions import ConnectionError
import json
import ipaddress
import hashlib
import re
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
socket_addr = '{0}/queue/sockets/queue'.format(pwd)
def send_event(msg, agent = None):
    if not agent or agent["id"] == "000":
        string = '1:misp:{0}'.format(json.dumps(msg))
    else:
        string = '1:[{0}] ({1}) {2}->misp:{3}'.format(agent["id"], agent["name"], agent["ip"] if "ip"
in agent else "any", json.dumps(msg))
    sock = socket(AF_UNIX, SOCK_DGRAM)
    sock.connect(socket_addr)
    sock.send(string.encode())
    sock.close()
false = False
# Read configuration parameters
alert_file = open(sys.argv[1])
# Read the alert file
alert = json.loads(alert_file.read())
alert_file.close()
# New Alert Output if MISP Alert or Error calling the API
alert_output = {}
# MISP Server Base URL
misp_base_url = "https://airca_misp:8443/attributes/restSearch/"
# MISP Server API AUTH KEY
misp_api_auth_key = "oRVerxzKc3JIMnpWGewIAwcd0JkWYU3EHLySgLoe"
# API - HTTP Headers
misp_apicall_headers = {"Content-Type":"application/json",
"Authorization":f"{misp_api_auth_key}", "Accept":"application/json"}
## Extract Sysmon for Windows/Sysmon for Linux and Sysmon Event ID
event_source = alert["rule"]["groups"][0]
event_type = alert["rule"]["groups"][3]
## Regex Pattern used based on SHA256 lenght (64 characters)
```

```
regex_file_hash = re.compile('\w{64}')
if event_source == 'windows':
    if event_type == 'sysmon_event1':
        try:
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event3' and alert["data"]["win"]["eventdata"]["destinationIsIpv6"]
== 'false':
        try:
            dst_ip = alert["data"]["win"]["eventdata"]["destinationIp"]
            if ipaddress.ip_address(dst_ip).is_global:
                wazuh_event_param = dst_ip
            else:
                sys.exit()
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event3' and
alert_output["data"]["win"]["eventdata"]["destinationIsIpv6"] == 'true':
        sys.exit()
    elif event_type == 'sysmon_event6':
        try:
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event7':
        try:
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event_15':
        try:
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event_22':
        try:
            wazuh_event_param = alert["data"]["win"]["eventdata"]["queryName"]
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event_23':
        try:
```

```
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event_24':
        try:
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    elif event_type == 'sysmon_event_25':
        try:
            wazuh_event_param =
regex_file_hash.search(alert["data"]["win"]["eventdata"]["hashes"]).group(0)
        except IndexError:
            sys.exit()
    else:
        sys.exit()
    misp_search_value = "value:"f"{wazuh_event_param}"
    misp_search_url = ''.join([misp_base_url, misp_search_value])
    try:
        misp_api_response = requests.get(misp_search_url, headers=misp_apicall_headers,
verify=False)
    except ConnectionError:
        alert_output["misp"] = {}
        alert_output["integration"] = "misp"
        alert_output["misp"]["error"] = 'Connection Error to MISP API'
        send_event(alert_output, alert["agent"])
    else:
        misp_api_response = misp_api_response.json()
    # Check if response includes Attributes (IoCs)
        if (misp_api_response["response"]["Attribute"]):
    # Generate Alert Output from MISP Response
            alert_output["misp"] = {}
            alert_output["misp"]["source"] = {}
            alert_output["misp"]["event_id"] =
misp_api_response["response"]["Attribute"][0]["event_id"]
            alert_output["misp"]["category"] =
misp_api_response["response"]["Attribute"][0]["category"]
            alert_output["misp"]["value"] = misp_api_response["response"]["Attribute"][0]["value"]
            alert_output["misp"]["type"] = misp_api_response["response"]["Attribute"][0]["type"]
            alert_output["misp"]["source"]["description"] = alert["rule"]["description"]
            send_event(alert_output, alert["agent"])
elif event_source == 'linux':
    if event_type == 'sysmon_event3' and alert["data"]["eventdata"]["destinationIsIpv6"] ==
'false':
        try:
```

```
        dst_ip = alert["data"]["eventdata"]["DestinationIp"]
        if ipaddress.ip_address(dst_ip).is_global:
            wazuh_event_param = dst_ip
            misp_search_value = "value:"f"{wazuh_event_param}"
            misp_search_url = ''.join([misp_base_url, misp_search_value])
            try:
                misp_api_response = requests.get(misp_search_url, headers=misp_apicall_headers,
verify=False)
            except ConnectionError:
                alert_output["misp"] = {}
                alert_output["integration"] = "misp"
                alert_output["misp"]["error"] = 'Connection Error to MISP API'
                send_event(alert_output, alert["agent"])
            else:
                misp_api_response = misp_api_response.json()
      # Check if response includes Attributes (IoCs)
                if (misp_api_response["response"]["Attribute"]):
            # Generate Alert Output from MISP Response
                    alert_output["misp"] = {}
                    alert_output["misp"]["event_id"] =
misp_api_response["response"]["Attribute"][0]["event_id"]
                    alert_output["misp"]["category"] =
misp_api_response["response"]["Attribute"][0]["category"]
                    alert_output["misp"]["value"] =
misp_api_response["response"]["Attribute"][0]["value"]
                    alert_output["misp"]["type"] =
misp_api_response["response"]["Attribute"][0]["type"]
                    send_event(alert_output, alert["agent"])
        else:
            sys.exit()
      except IndexError:
        sys.exit()
   else:
      sys.exit()
elif event_source == 'ossec' and event_type == "syscheck_entry_added":
   try:
      wazuh_event_param = alert["syscheck"]["sha256_after"]
   except IndexError:
      sys.exit()
   misp_search_value = "value:"f"{wazuh_event_param}"
   misp_search_url = ''.join([misp_base_url, misp_search_value])
   try:
      misp_api_response = requests.get(misp_search_url, headers=misp_apicall_headers,
verify=false)
   except ConnectionError:
      alert_output["misp"] = {}
      alert_output["integration"] = "misp"
```

```
        alert_output["misp"]["error"] = 'Connection Error to MISP API'
        send_event(alert_output, alert["agent"])
    else:
        misp_api_response = misp_api_response.json()
    # Check if response includes Attributes (IoCs)
        if (misp_api_response["response"]["Attribute"]):
    # Generate Alert Output from MISP Response
            alert_output["misp"] = {}
            alert_output["misp"]["event_id"] =
misp_api_response["response"]["Attribute"][0]["event_id"]
            alert_output["misp"]["category"] =
misp_api_response["response"]["Attribute"][0]["category"]
            alert_output["misp"]["value"] = misp_api_response["response"]["Attribute"][0]["value"]
            alert_output["misp"]["type"] = misp_api_response["response"]["Attribute"][0]["type"]
            send_event(alert_output, alert["agent"])
else:
    sys.exit()
```

9.4.3.3. MISP integration xml rule

```
<group name="misp,">
  <rule id="100620" level="10">
    <field name="integration">misp</field>
    <match>misp</match>
    <description>MISP Events</description>
    <options>no_full_log</options>
  </rule>
  <rule id="100621" level="5">
    <if_sid>100620</if_sid>
    <field name="misp.error">\.+</field>
    <description>MISP - Error connecting to API</description>
    <options>no_full_log</options>
    <group>misp_error,</group>
  </rule>
  <rule id="100622" level="12">
    <field name="misp.category">\.+</field>
    <description>MISP - IoC found in Threat Intel - Category: $(misp.category), Attribute:
$(misp.value)</description>
    <options>no_full_log</options>
    <group>misp_alert,</group>
  </rule>
</group>
```

9.4.3.4. Sysmon xml rule

```
<!--
  - Full event logging for Sysmon v15.0 via the Windows Event Channel
  - All rules set at minimum alerting threshhold (3)
  - ID ranges 61656 - 61658 used for new Event ID's 27 to 29
-->


<group name="windows,sysmon,sysmon_v15,">
  <rule id="61603" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^1$</field>
    <description>Process creation $(win.eventdata.description)</description>
    <options>no_full_log</options>
    <group>sysmon_event1,</group>
  </rule>

  <rule id="61604" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^2$</field>
    <description>$(win.eventdata.image) changed file $(win.eventdata.targetFilename) creation
time </description>
    <options>no_full_log</options>
    <group>sysmon_event2,</group>
  </rule>

  <rule id="61605" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^3$</field>
    <description>Network connection to
$(win.eventdata.destinationIp):$(win.eventdata.destinationPort) by
$(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event3,</group>
  </rule>

  <rule id="61606" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^4$</field>
    <description>Sysmon service state changed to "$(win.eventdata.state)"</description>
    <options>no_full_log</options>
    <group>sysmon_event4,</group>
  </rule>

  <rule id="61607" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
```

```
    <field name="win.system.eventID">^5$</field>
    <description>Process terminated $(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event5,</group>
  </rule>

  <rule id="61608" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^6$</field>
    <description>Driver loaded $(win.eventdata.imageLoaded)</description>
    <options>no_full_log</options>
    <group>sysmon_event6,</group>
  </rule>

  <rule id="61609" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^7$</field>
    <description>Image $(win.eventdata.imageLoaded) loaded by
$(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event7,</group>
  </rule>

  <rule id="61610" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^8$</field>
    <description>CreateRemoteThread by $(win.eventdata.sourceImage) on
$(win.eventdata.targetImage), possible process injection</description>
    <options>no_full_log</options>
    <group>sysmon_event8,</group>
  </rule>

  <rule id="61611" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^9$</field>
    <description>RawAccessRead by $(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event9,</group>
  </rule>

  <rule id="61612" level="3" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^10$</field>
    <description>$(win.eventdata.targetImage) process accessed by
$(win.eventdata.sourceImage)</description>
    <options>no_full_log</options>
    <group>sysmon_event_10,</group>
```

```
  </rule>

  <rule id="61613" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^11$</field>
    <description>FileCreated :: $(win.eventdata.targetFilename)</description>
    <options>no_full_log</options>
    <group>sysmon_event_11,</group>
  </rule>

  <rule id="61614" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^12$</field>
    <description>RegistryEvent $(win.eventdata.eventType) on $(win.eventdata.targetObject) by
$(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event_12,</group>
  </rule>

  <rule id="61615" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^13$</field>
    <description>RegistryEvent $(win.eventdata.eventType) on $(win.eventdata.targetObject) by
$(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event_13,</group>
  </rule>

  <rule id="61616" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^14$</field>
    <description>RegistryEvent (Key and Value Rename) by
$(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event_14,</group>
  </rule>

  <rule id="61617" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^15$</field>
    <description>$(win.eventdata.targetFilename) FileCreateStreamHash by process
$(win.eventdata.image)</description>
    <options>no_full_log</options>
    <group>sysmon_event_15,</group>
  </rule>

  <rule id="61644" level="3" overwrite="yes">
```

```
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^16$</field>
    <description>Sysmon configuration changed using file
$(win.eventdata.configuration)</description>
    <group>sysmon_event_16,</group>
  </rule>

  <rule id="61645" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^17$</field>
    <description>Pipe created</description>
    <options>no_full_log</options>
    <group>sysmon_event_17,</group>
  </rule>

  <rule id="61646" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^18$</field>
    <description>Pipe connected</description>
    <options>no_full_log</options>
    <group>sysmon_event_18,</group>
  </rule>

  <rule id="61647" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^19$</field>
    <description>WmiEventFilter activity</description>
    <options>no_full_log</options>
    <group>sysmon_event_19,</group>
  </rule>

  <rule id="61648" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^20$</field>
    <description>WmiEventConsumer activity</description>
    <options>no_full_log</options>
    <group>sysmon_event_20,</group>
  </rule>

  <rule id="61649" level="3" overwrite="yes" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^21$</field>
    <description>WmiEventConsumerToFilter activity</description>
    <options>no_full_log</options>
    <group>sysmon_event_21,</group>
  </rule>
```

```
<rule id="61650" level="3" overwrite="yes">
  <if_sid>61600</if_sid>
  <field name="win.system.eventID">^22$</field>
  <description>DNS Query :: $(win.eventdata.queryName)</description>
  <options>no_full_log</options>
  <group>sysmon_event_22,</group>
</rule>

<rule id="61651" level="3" overwrite="yes" noalert="1">
  <if_sid>61600</if_sid>
  <field name="win.system.eventID">^23$</field>
  <description>File deleted and archived</description>
  <options>no_full_log</options>
  <group>sysmon_event_23,</group>
</rule>

<rule id="61652" level="3" overwrite="yes" noalert="1">
  <if_sid>61600</if_sid>
  <field name="win.system.eventID">^24$</field>
  <description>Clipboard change</description>
  <options>no_full_log</options>
  <group>sysmon_event_24,</group>
</rule>

<rule id="61653" level="3" overwrite="yes" noalert="1">
  <if_sid>61600</if_sid>
  <field name="win.system.eventID">^25$</field>
  <description>Process tampering - Image change</description>
  <options>no_full_log</options>
  <group>sysmon_event_25,</group>
</rule>

<rule id="61654" level="3" overwrite="yes" noalert="1">
  <if_sid>61600</if_sid>
  <field name="win.system.eventID">^26$</field>
  <description>File deleted</description>
  <options>no_full_log</options>
  <group>sysmon_event_26,</group>
</rule>

<rule id="61655" level="3" overwrite="yes" noalert="1">
  <if_sid>61600</if_sid>
  <field name="win.system.eventID">^255$</field>
  <description>Sysmon error</description>
  <options>no_full_log</options>
  <group>sysmon_event_255,</group>
</rule>
```

```
  <rule id="61656" level="3" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^27$</field>
    <description>Detection and Block - Creation of executable files (PE format)</description>
    <options>no_full_log</options>
    <group>sysmon_event_27,</group>
  </rule>

  <rule id="61657" level="3" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^28$</field>
    <description>Detection and Block - File Shredding</description>
    <options>no_full_log</options>
    <group>sysmon_event_28,</group>
  </rule>

  <rule id="61658" level="5" noalert="1">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^29$</field>
    <description>Creation of new executable file detected (PE format)</description>
    <options>no_full_log</options>
    <group>sysmon_event_29,</group>
  </rule>
</group>
```

9.4.3.5. Start.sh script

```
#!/usr/bin/env bash

# Wazuh Manager's Container Name
MANAGER="airca_wazuh_manager"

# Wazuh Configuration Path @ Host
CONF_FILE_PATH="wazuh/files"

# Wazuh Configuration Path @ Container
AGENT_CONF="/var/ossec/etc/shared/default/agent.conf"
OSSEC_CONF="/var/ossec/etc/ossec.conf"
CUSTOM_RULES="/var/ossec/etc/rules"
CUSTOM_MISP="/var/ossec/integrations/custom-misp.py"
CUSTOM_DECODER="/var/ossec/etc/decoders/local_decoder.xml"

echo
echo "******************************"
echo "Starting your airca instances"
echo "******************************"
echo

# Start the docker containers
docker-compose up -d

# Push Wazuh Manager's Configuration Files
sleep 3
docker cp $CONF_FILE_PATH/agent.conf $MANAGER:$AGENT_CONF
docker cp $CONF_FILE_PATH/ossec.conf $MANAGER:$OSSEC_CONF
docker cp $CONF_FILE_PATH/rules $MANAGER:/var/ossec/etc/
docker cp $CONF_FILE_PATH/local_decoder.xml $MANAGER:$CUSTOM_DECODER
docker cp $CONF_FILE_PATH/custom-misp.py $MANAGER:$CUSTOM_MISP

# Change permission of the Wazuh Manager's Configuration File
docker exec -d $MANAGER --user root bash -c "chown root:wazuh $AGENT_CONF &&
chmod 660 $AGENT_CONF"
docker exec -d $MANAGER bash -c "chown root:wazuh $OSSEC_CONF && chmod 660
$OSSEC_CONF"
docker exec -d $MANAGER bash -c "chown -R root:wazuh $CUSTOM_RULES && chmod
640 $CUSTOM_RULES/*"
docker exec -d $MANAGER bash -c "chown root:wazuh $CUSTOM_DECODER && chmod
640 $CUSTOM_DECODER"
docker exec -d $MANAGER bash -c "chown root:wazuh $CUSTOM_MISP && chmod 750
$CUSTOM_MISP"
docker exec -d $MANAGER bash -c "service restart wazuh-manager"
sleep 2
```

9.4.3.6. Stop.sh script

```
#!/usr/bin/env bash

echo "******************************"
echo "Stoppng your airca instances"
echo "******************************"
echo
docker-compose down
echo
```

9.4.4. Fourth Iteration - Development of detection rules and active response

9.4.4.1. Ossec config

```
<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <logall_json>no</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>wazuh@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
  </global>

  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>12</email_alert_level>
  </alerts>

  <!-- Choose between "plain", "json", or "plain,json" for the format of internal logs -->
  <logging>
    <log_format>plain</log_format>
  </logging>

  <remote>
    <connection>secure</connection>
    <port>1514</port>
    <protocol>tcp</protocol>
    <queue_size>131072</queue_size>
  </remote>

  <!-- Policy monitoring -->
  <rootcheck>
    <disabled>no</disabled>
    <check_files>yes</check_files>
    <check_trojans>yes</check_trojans>
    <check_dev>yes</check_dev>
    <check_sys>yes</check_sys>
    <check_pids>yes</check_pids>
    <check_ports>yes</check_ports>
```

```
    <check_if>yes</check_if>

    <!-- Frequency that rootcheck is executed - every 12 hours -->
    <frequency>43200</frequency>

    <rootkit_files>etc/rootcheck/rootkit_files.txt</rootkit_files>
    <rootkit_trojans>etc/rootcheck/rootkit_trojans.txt</rootkit_trojans>

    <skip_nfs>yes</skip_nfs>
  </rootcheck>

  <wodle name="cis-cat">
    <disabled>yes</disabled>
    <timeout>1800</timeout>
    <interval>1d</interval>
    <scan-on-start>yes</scan-on-start>

    <java_path>wodles/java</java_path>
    <ciscat_path>wodles/ciscat</ciscat_path>
  </wodle>

  <!-- Osquery integration -->
  <wodle name="osquery">
    <disabled>yes</disabled>
    <run_daemon>yes</run_daemon>
    <log_path>/var/log/osquery/osqueryd.results.log</log_path>
    <config_path>/etc/osquery/osquery.conf</config_path>
    <add_labels>yes</add_labels>
  </wodle>

  <!-- Misp integration -->
  <integration>
    <name>custom-misp.py</name>

<group>sysmon_event1,sysmon_event3,sysmon_event6,sysmon_event7,sysmon_event_15,sysm
on_event_22,syscheck</group>
    <alert_format>json</alert_format>
  </integration>

  <!-- System inventory -->
  <wodle name="syscollector">
    <disabled>no</disabled>
    <interval>1h</interval>
    <scan_on_start>yes</scan_on_start>
    <hardware>yes</hardware>
    <os>yes</os>
    <network>yes</network>
```

```
    <packages>yes</packages>
    <ports all="no">yes</ports>
    <processes>yes</processes>

    <!-- Database synchronization settings -->
    <synchronization>
      <max_eps>10</max_eps>
    </synchronization>
  </wodle>

  <sca>
    <enabled>yes</enabled>
    <scan_on_start>yes</scan_on_start>
    <interval>12h</interval>
    <skip_nfs>yes</skip_nfs>
  </sca>

  <vulnerability-detector>
    <enabled>yes</enabled>
    <interval>5m</interval>
    <min_full_scan_interval>6h</min_full_scan_interval>
    <run_on_start>yes</run_on_start>

    <!-- Ubuntu OS vulnerabilities -->
    <provider name="canonical">
      <enabled>yes</enabled>
      <os>trusty</os>
      <os>xenial</os>
      <os>bionic</os>
      <os>focal</os>
      <os>jammy</os>
      <update_interval>1h</update_interval>
    </provider>

    <!-- Debian OS vulnerabilities -->
    <provider name="debian">
      <enabled>yes</enabled>
      <os>buster</os>
      <os>bullseye</os>
      <os>bookworm</os>
      <update_interval>1h</update_interval>
    </provider>

    <!-- RedHat OS vulnerabilities -->
    <provider name="redhat">
      <enabled>no</enabled>
      <os>5</os>
```

```
  <os>6</os>
  <os>7</os>
  <os>8</os>
  <os>9</os>
  <update_interval>1h</update_interval>
</provider>

<!-- Amazon Linux OS vulnerabilities -->
<provider name="alas">
  <enabled>no</enabled>
  <os>amazon-linux</os>
  <os>amazon-linux-2</os>
  <os>amazon-linux-2023</os>
  <update_interval>1h</update_interval>
</provider>

<!-- SUSE Linux Enterprise OS vulnerabilities -->
<provider name="suse">
  <enabled>no</enabled>
  <os>11-server</os>
  <os>11-desktop</os>
  <os>12-server</os>
  <os>12-desktop</os>
  <os>15-server</os>
  <os>15-desktop</os>
  <update_interval>1h</update_interval>
</provider>

<!-- Arch OS vulnerabilities -->
<provider name="arch">
  <enabled>no</enabled>
  <update_interval>1h</update_interval>
</provider>

<!-- Alma Linux OS vulnerabilities -->
<provider name="almalinux">
  <enabled>no</enabled>
  <os>8</os>
  <os>9</os>
  <update_interval>1h</update_interval>
</provider>

<!-- Windows OS vulnerabilities -->
<provider name="msu">
  <enabled>yes</enabled>
  <update_interval>1h</update_interval>
</provider>
```

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

 </vulnerability-detector>

 <!-- File integrity monitoring -->
 <syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Generate alert when new file detected -->
  <alert_new_files>yes</alert_new_files>

  <!-- Don't ignore files that change more than 'frequency' times -->
  <auto_ignore frequency="10" timeframe="3600">no</auto_ignore>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories>/etc,/usr/bin,/usr/sbin</directories>
  <directories>/bin,/sbin,/boot</directories>

  <!-- Files/directories to ignore -->
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/hosts.deny</ignore>
  <ignore>/etc/mail/statistics</ignore>
  <ignore>/etc/random-seed</ignore>
  <ignore>/etc/random.seed</ignore>
  <ignore>/etc/adjtime</ignore>
  <ignore>/etc/httpd/logs</ignore>
  <ignore>/etc/utmpx</ignore>
  <ignore>/etc/wtmpx</ignore>
  <ignore>/etc/cups/certs</ignore>
  <ignore>/etc/dumpdates</ignore>
  <ignore>/etc/svc/volatile</ignore>

  <!-- File types to ignore -->
  <ignore type="sregex">.log$|.swp$</ignore>

  <!-- Check the file, but never compute the diff -->
  <nodiff>/etc/ssl/private.key</nodiff>
```

```
  <skip_nfs>yes</skip_nfs>
  <skip_dev>yes</skip_dev>
  <skip_proc>yes</skip_proc>
  <skip_sys>yes</skip_sys>

  <!-- Nice value for Syscheck process -->
  <process_priority>10</process_priority>

  <!-- Maximum output throughput -->
  <max_eps>100</max_eps>

  <!-- Database synchronization settings -->
  <synchronization>
    <enabled>yes</enabled>
    <interval>5m</interval>
    <max_interval>1h</max_interval>
    <max_eps>10</max_eps>
  </synchronization>
</syscheck>

<!-- Active response -->
<global>
  <white_list>127.0.0.1</white_list>
  <white_list>^localhost.localdomain$</white_list>
</global>

<command>
  <name>disable-account</name>
  <executable>disable-account</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>restart-wazuh</name>
  <executable>restart-wazuh</executable>
</command>

<command>
  <name>firewall-drop</name>
  <executable>firewall-drop</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>host-deny</name>
  <executable>host-deny</executable>
```

```
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>route-null</name>
    <executable>route-null</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>win_route-null</name>
    <executable>route-null.exe</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <command>
    <name>netsh</name>
    <executable>netsh.exe</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

  <!--
  <active-response>
    active-response options here
  </active-response>
  -->

  <!-- Log analysis -->
  <localfile>
    <log_format>command</log_format>
    <command>df -P</command>
    <frequency>360</frequency>
  </localfile>

  <localfile>
    <log_format>full_command</log_format>
    <command>netstat -tulpn | sed 's/\([[:alnum:]]\+\)\ \+[[:digit:]]\+\ \+[[:digit:]]\+\
\+\(.*\):\([[:digit:]]*\)\ \+\([0-9\.\:\*]\+\).\+\ \([[:digit:]]*\/[[:alnum:]\-]*\).*/\1 \2 == \3 == \4 \5/' |
sort -k 4 -g | sed 's/ == \(.*\) ==/:\1/' | sed 1,2d</command>
    <alias>netstat listening ports</alias>
    <frequency>360</frequency>
  </localfile>

  <localfile>
    <log_format>full_command</log_format>
    <command>last -n 20</command>
    <frequency>360</frequency>
```

```
</localfile>

<ruleset>
  <!-- Default ruleset -->
  <decoder_dir>ruleset/decoders</decoder_dir>
  <rule_dir>ruleset/rules</rule_dir>
  <rule_exclude>0215-policy_rules.xml</rule_exclude>
  <list>etc/lists/audit-keys</list>
  <list>etc/lists/amazon/aws-eventnames</list>
  <list>etc/lists/security-eventchannel</list>

  <!-- User-defined ruleset -->
  <decoder_dir>etc/decoders</decoder_dir>
  <rule_dir>etc/rules</rule_dir>
</ruleset>

<rule_test>
  <enabled>yes</enabled>
  <threads>1</threads>
  <max_sessions>64</max_sessions>
  <session_timeout>15m</session_timeout>
</rule_test>

<!-- Configuration for wazuh-authd -->
<auth>
  <disabled>no</disabled>
  <port>1515</port>
  <use_source_ip>no</use_source_ip>
  <purge>yes</purge>
  <use_password>no</use_password>
  <ciphers>HIGH:!ADH:!EXP:!MD5:!RC4:!3DES:!CAMELLIA:@STRENGTH</ciphers>
  <!-- <ssl_agent_ca></ssl_agent_ca> -->
  <ssl_verify_host>no</ssl_verify_host>
  <ssl_manager_cert>etc/sslmanager.cert</ssl_manager_cert>
  <ssl_manager_key>etc/sslmanager.key</ssl_manager_key>
  <ssl_auto_negotiate>no</ssl_auto_negotiate>
</auth>

<cluster>
  <name>wazuh</name>
  <node_name>node01</node_name>
  <node_type>master</node_type>
  <key>aa093264ef885029653eea20dfcf51ae</key>
  <port>1516</port>
  <bind_addr>0.0.0.0</bind_addr>
  <nodes>
     <node>wazuh.manager</node>
```

```
    </nodes>
    <hidden>no</hidden>
    <disabled>yes</disabled>
  </cluster>

</ossec_config>

<ossec_config>
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/ossec/logs/active-responses.log</location>
  </localfile>

</ossec_config>

<ossec_config>
  <command>
    <name>yara_linux</name>
    <executable>yara.sh</executable>
    <extra_args>-yara_path /usr/local/bin -yara_rules /var/yara/rules/yara_rules.yar</extra_args>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <command>yara_linux</command>
    <location>local</location>
    <rules_id>100200,100201</rules_id>
  </active-response>
</ossec_config>


<ossec_config>
  <command>
    <name>yara_windows</name>
    <executable>yara.bat</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <command>yara_windows</command>
    <location>local</location>
    <rules_id>100303,100304</rules_id>
  </active-response>
</ossec_config>

<ossec_config>
  <command>
```

```
    <name>quarantine_file</name>
    <executable>quarantine_file.bat</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <command>quarantine_file</command>
    <location>local</location>
    <rules_id>108001</rules_id>
  </active-response>
</ossec_config>
```

9.4.4.2. Agent config

```
<agent_config os="windows">
<!-- Sysmon -->
  <localfile>
    <location>Microsoft-Windows-Sysmon/Operational</location>
    <log_format>eventchannel</log_format>
  </localfile>
<!-- PowerShell -->
  <localfile>
    <location>Microsoft-Windows-PowerShell/Operational</location>
    <log_format>eventchannel</log_format>
  </localfile>
</agent_config>


<agent_config>
<wodle name="syscollector">
   <disabled>no</disabled>
   <interval>1h</interval>
   <os>yes</os>
   <packages>yes</packages>
   <hotfixes>no</hotfixes>
</wodle>
</agent_config>
```

9.4.4.3. Local decoder for yara

```
<decoder name="yara_decoder">
  <prematch>wazuh-yara:</prematch>
</decoder>

<decoder name="yara_decoder1">
  <parent>yara_decoder</parent>
  <regex>wazuh-yara: (\S+) - Scan result: (\S+) (\S+)</regex>
  <order>log_type, yara_rule, yara_scanned_file</order>
</decoder>
```

9.4.4.4. Local rules for yara

```
<group name="syscheck,">
  <rule id="100200" level="7">
    <if_sid>550</if_sid>
    <field name="file">/var/www/html/images/</field>
    <description>File modified in /var/www/html/images directory.</description>
    <group>syscheck,</group>
  </rule>
  <rule id="100201" level="7">
    <if_sid>554</if_sid>
    <field name="file">/var/www/html/images/</field>
    <description>File added in /var/www/html/images directory.</description>
    <group>syscheck,</group>
  </rule>
</group>

<group name="syscheck,">
  <rule id="100303" level="7">
    <if_sid>550</if_sid>
    <field name="file">C:\\Users\\bakra\\Downloads</field>
    <description>File modified in C:\Users\bakra\Downloads directory.</description>
  </rule>
  <rule id="100304" level="7">
    <if_sid>554</if_sid>
    <field name="file">C:\\Users\\bakra\\Downloads</field>
    <description>File added to C:\Users\bakra\Downloads  directory.</description>
  </rule>
</group>

<group name="yara,">
  <rule id="108000" level="0">
    <decoded_as>yara_decoder</decoded_as>
    <description>Yara grouping rule</description>
  </rule>

  <rule id="108001" level="12">
    <if_sid>108000</if_sid>
    <match>wazuh-yara: INFO - Scan result: </match>
    <description>File "$(yara_scanned_file)" is a positive match. Yara rule:
$(yara_rule)</description>
  </rule>
</group>
```

9.4.4.5. Yara scan script for ubuntu

```
#!/bin/bash

# Extra arguments
read INPUT_JSON
YARA_PATH=$(echo $INPUT_JSON | jq -r .parameters.extra_args[1])
YARA_RULES=$(echo $INPUT_JSON | jq -r .parameters.extra_args[3])
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.syscheck.path)

# Set LOG_FILE path
LOG_FILE="logs/active-responses.log"

size=0
actual_size=$(stat -c %s ${FILENAME})
while [ ${size} -ne ${actual_size} ]; do
   sleep 1
   size=${actual_size}
   actual_size=$(stat -c %s ${FILENAME})
done

#------------------------ Analyze parameters -------------------------#

if [[ ! $YARA_PATH ]] || [[ ! $YARA_RULES ]]
then
   echo "wazuh-yara: ERROR - Yara active response error. Yara path and rules parameters are
mandatory." >> ${LOG_FILE}
   exit 1
fi

#-------------------------- Main workflow --------------------------#

# Execute Yara scan on the specified filename
yara_output="$("${YARA_PATH}"/yara -w -r "$YARA_RULES" "$FILENAME")"

if [[ $yara_output != "" ]]
then
   # Iterate every detected rule and append it to the LOG_FILE
   while read -r line; do
      echo "wazuh-yara: INFO - Scan result: $line" >> ${LOG_FILE}
   done <<< "$yara_output"
fi

exit 0;
```

9.4.4.6. Yara scan script for windows

```
@echo off

setlocal enableDelayedExpansion

reg Query "HKLM\Hardware\Description\System\CentralProcessor\0" | find /i "x86" > NUL &&
SET OS=32BIT || SET OS=64BIT


if %OS%==32BIT (
   SET log_file_path="%programfiles%\ossec-agent\active-response\active-responses.log"
)

if %OS%==64BIT (
   SET log_file_path="%programfiles(x86)%\ossec-agent\active-response\active-responses.log"
)

set input=
for /f "delims=" %%a in ('PowerShell -command "$logInput = Read-Host; Write-Output
$logInput"') do (
   set input=%%a
)


set json_file_path="C:\Program Files (x86)\ossec-agent\active-response\stdin.txt"
set syscheck_file_path=
echo %input% > %json_file_path%

for /F "tokens=* USEBACKQ" %%F in (`Powershell -Nop -C "(Get-Content 'C:\Program Files
(x86)\ossec-agent\active-response\stdin.txt'|ConvertFrom-Json).parameters.alert.syscheck.path"`)
do (
set syscheck_file_path=%%F
)

del /f %json_file_path%
set yara_exe_path="C:\Program Files (x86)\ossec-agent\active-response\bin\yara\yara64.exe"
set yara_rules_path="C:\Program Files (x86)\ossec-agent\active-
response\bin\yara\rules\yara_rules.yar"
echo %syscheck_file_path% >> %log_file_path%
for /f "delims=" %%a in ('powershell -command "& \"%yara_exe_path%\"
\"%yara_rules_path%\" \"%syscheck_file_path%\"'') do (
   echo wazuh-yara: INFO - Scan result: %%a >> %log_file_path%
)

exit /b
```

9.4.4.7. Active response script

```
@echo off
setlocal enableDelayedExpansion

rem Determine the OS architecture
reg Query "HKLM\Hardware\Description\System\CentralProcessor\0" | find /i "x86" > NUL &&
SET OS=32BIT || SET OS=64BIT

rem Set the log file path based on OS architecture
if %OS%==32BIT (
   SET log_file_path="%programfiles%\ossec-agent\active-response\active-responses.log"
) else (
   SET log_file_path="%programfiles(x86)%\ossec-agent\active-response\active-responses.log"
)

rem Prompt for input (you can modify this part as needed)
set input=
for /f "delims=" %%a in ('PowerShell -command "$logInput = Read-Host; Write-Output
$logInput"') do (
   set input=%%a
)

rem Create JSON file to store input
set json_file_path="C:\Program Files (x86)\ossec-agent\active-response\stdin.txt"
echo %input% > %json_file_path%

rem Extract the file path from JSON input
set syscheck_file_path=
for /F "tokens=* USEBACKQ" %%F in (`Powershell -Nop -C "(Get-Content 'C:\Program Files
(x86)\ossec-agent\active-response\stdin.txt'|ConvertFrom-
Json).parameters.alert.data.yara_scanned_file"`) do (
   set syscheck_file_path=%%F
)

rem Delete JSON file
del /f %json_file_path%

rem Set quarantine directory
set "quarantineDir=C:\Users\bakra\AppData\Local\Temp\Quarantine"

rem Check if quarantine directory exists, if not, create it
if not exist "%quarantineDir%" (
   mkdir "%quarantineDir%"
)

rem Set the file to read-only
```

```
attrib +r "%syscheck_file_path%"

rem Move the flagged file to the quarantine directory
move /y "%syscheck_file_path%" "%quarantineDir%"

rem Rename the file with new extension
for %%F in ("%quarantineDir%\*") do (
   ren "%%F" "%%~nF.airca"
)

rem Add custom message to indicate that the file has been quarantined
echo File has been quarantined: %syscheck_file_path% >> %log_file_path%

exit /b
```

9.4.5. Fifth Iteration - Customization of Wazuh and completion of project

9.4.5.1. Customized Screen Process

```
wazuh.dashboard:
  image: wazuh/wazuh-dashboard:4.7.0
  hostname: wazuh.dashboard
  container_name: airca_wazuh_dashboard
  restart: always
  ports:
    - 443:5601
  environment:
    - INDEXER_USERNAME=admin
    - INDEXER_PASSWORD=SecretPassword
    - WAZUH_API_URL=https://wazuh.manager
    - DASHBOARD_USERNAME=kibanaserver
    - DASHBOARD_PASSWORD=kibanaserver
    - API_USERNAME=wazuh-wui
    - API_PASSWORD=MyS3cr37P450r.*-
  volumes:
    - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.dashboard.pem:/usr/share/wazuh-dashboard/certs/wazuh-dashboard.pem
    - ./wazuh/config/wazuh_indexer_ssl_certs/wazuh.dashboard-key.pem:/usr/share/wazuh-dashboard/certs/wazuh-dashboard-key.pem
    - ./wazuh/config/wazuh_indexer_ssl_certs/root-ca.pem:/usr/share/wazuh-dashboard/certs/root-ca.pem
    - ./wazuh/config/wazuh_dashboard/opensearch_dashboards.yml:/usr/share/wazuh-dashboard/config/opensearch_dashboards.yml
    - ./wazuh/config/wazuh_dashboard/wazuh.yml:/usr/share/wazuh-dashboard/data/wazuh/config/wazuh.yml
    - ./wazuh/files/logo/Only-AIRCA-WHITE.svg:/usr/share/wazuh-dashboard/plugins/wazuh/public/assets/custom/images/Only-AIRCA-WHITE.svg
    - ./wazuh/files/logo/AIRCA-LOGO.svg:/usr/share/wazuh-dashboard/src/core/server/core_app/assets/Wazuh-Logo.svg
    - ./wazuh/files/template.js:/usr/share/wazuh-dashboard/src/core/server/rendering/views/template.js
    - wazuh-dashboard-config:/usr/share/wazuh-dashboard/data/wazuh/config
    - wazuh-dashboard-custom:/usr/share/wazuh-dashboard/plugins/wazuh/public/assets/custom
  depends_on:
    - wazuh.indexer
  links:
    - wazuh.indexer:wazuh.indexer
    - wazuh.manager:wazuh.manager
```

Figure 154 Specifying file location of logo in host machine to use in container

```
airca@airca:~/opt/airca-docker/wazuh/config/wazuh_dashboard$
airca@airca:~/opt/airca-docker/wazuh/config/wazuh_dashboard$
airca@airca:~/opt/airca-docker/wazuh/config/wazuh_dashboard$ cat wazuh.yml
hosts:
  - 1513629884013:
      url: "https://wazuh.manager"
      port: 55000
      username: wazuh-wui
      password: "MyS3cr37P450r.*-"
      run_as: true

customization.enabled: true
customization.logo.app: "custom/images/Only-AIRCA-WHITE.svg"
customization.logo.healthcheck: "custom/images/Only-AIRCA-WHITE.svg"

wazuh.monitoring.enabled: false
checks.template: false
hideManagerAlerts: true
airca@airca:~/opt/airca-docker/wazuh/config/wazuh_dashboard$
airca@airca:~/opt/airca-docker/wazuh/config/wazuh_dashboard$
```

Figure 155 Enabling customization for Wazuh Dashboard

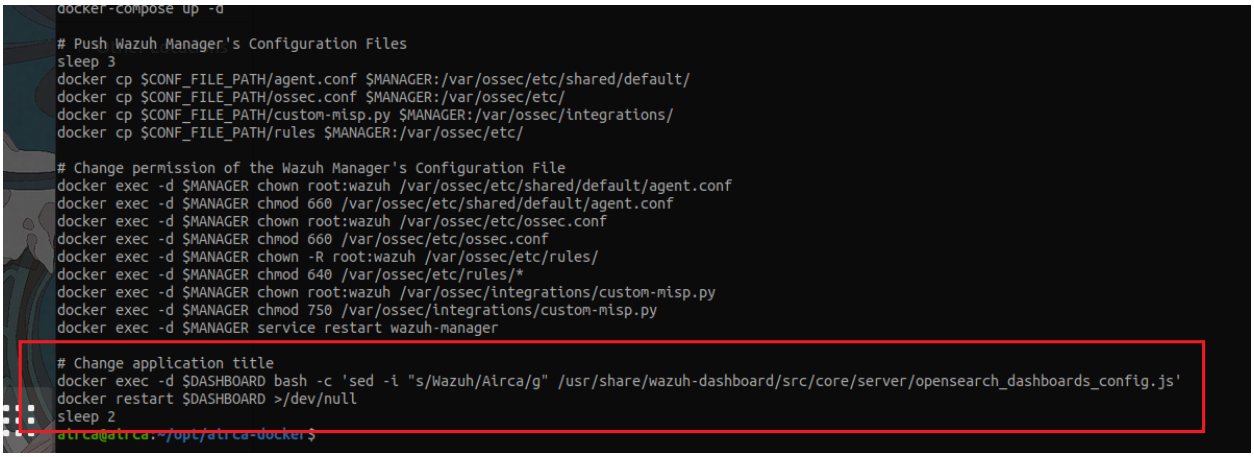


Figure 156 Changing application title from start.sh script

9.4.5.2. Customized Screen Overview

Loading ...

Figure 157 Loading Screen of AIRCA

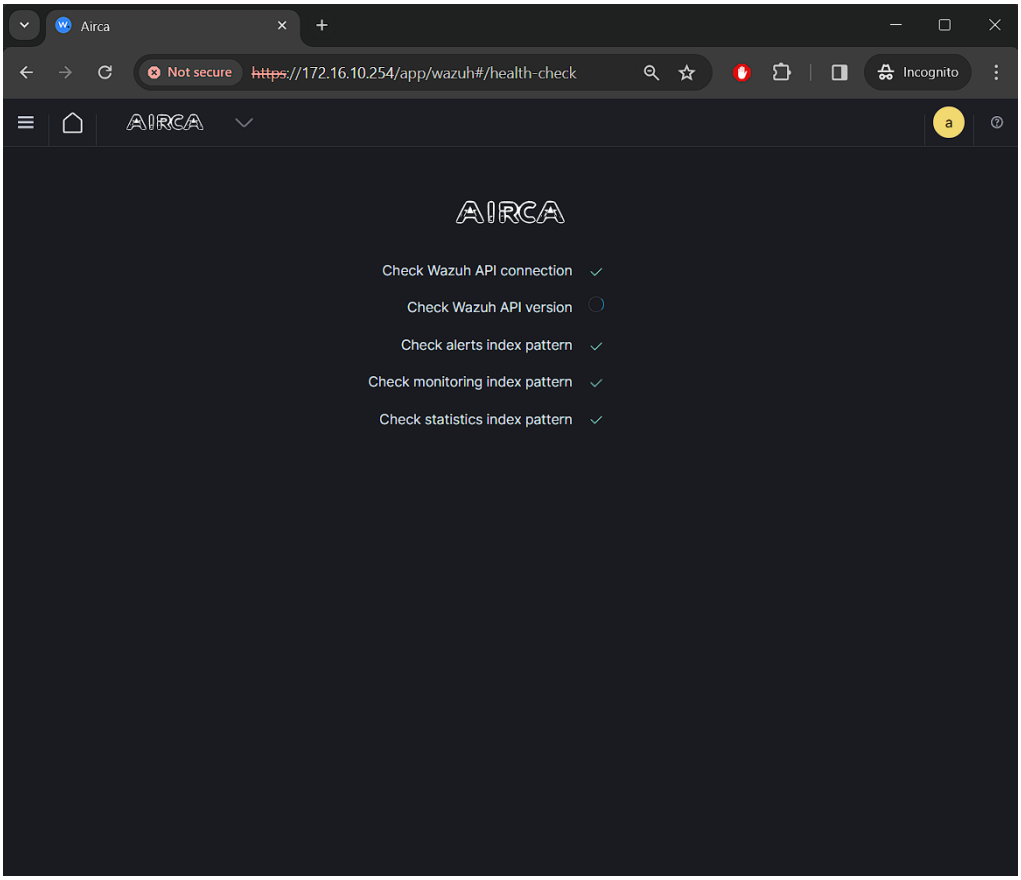


Figure 158 Health Check UI in AIRCA

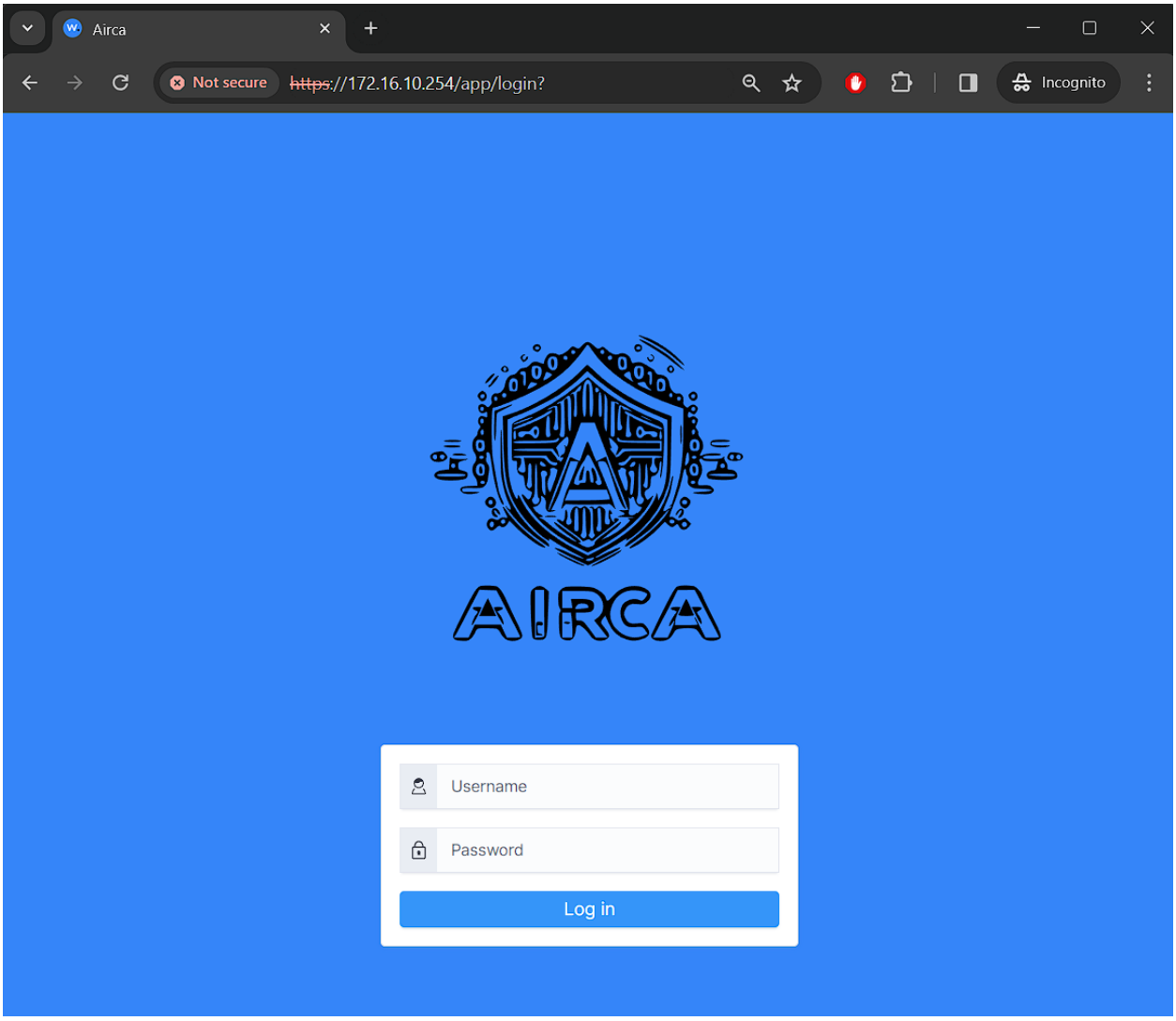


Figure 159 Login Page UI of AIRCA

9.5. Appendix E: Collection of Charts

9.5.1. Work Breakdown Structure

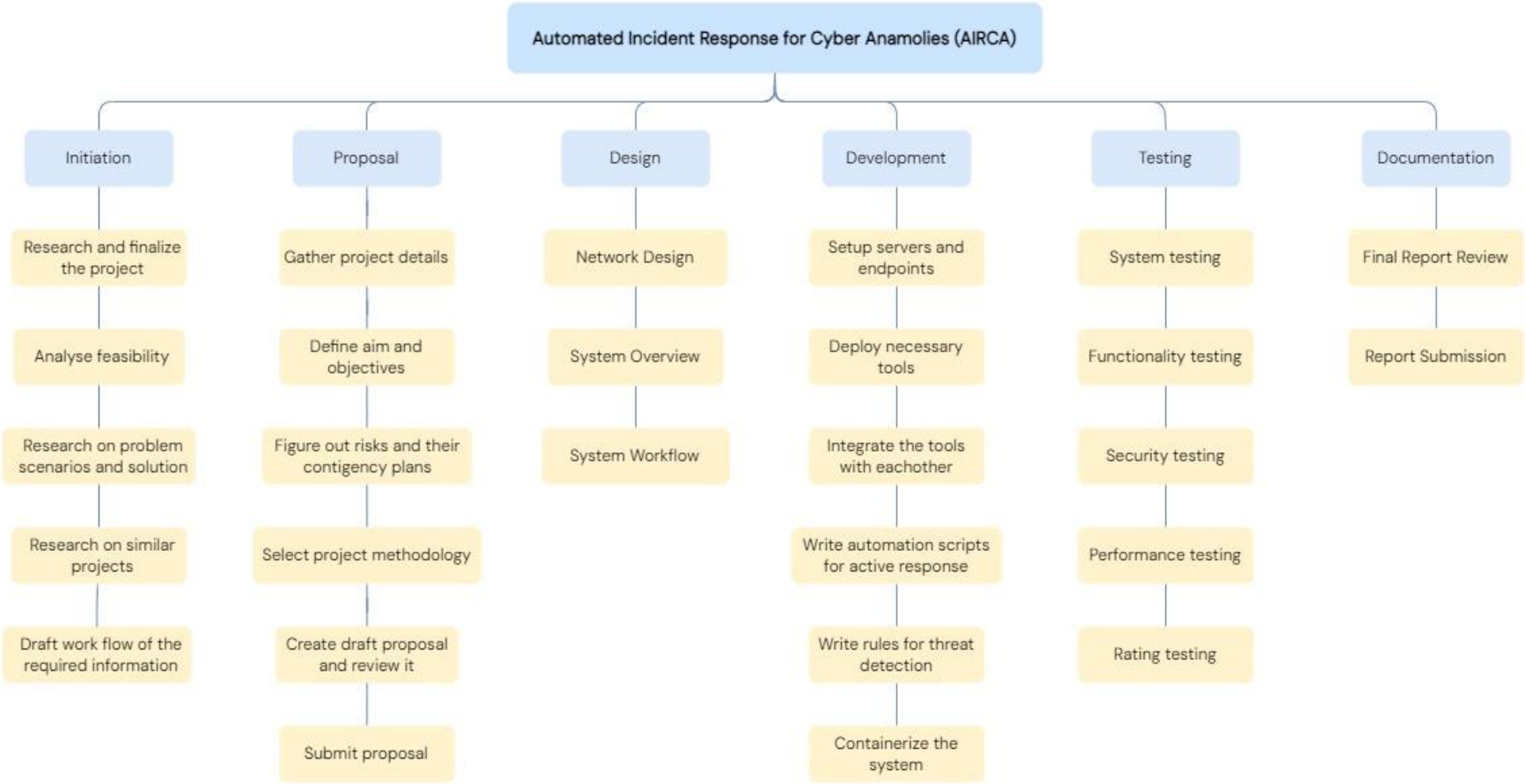


Figure 160 Work Breakdown Structure

9.5.2. Milestones

Figure 161 Project Milestones

9.5.3. Project Gantt Chart

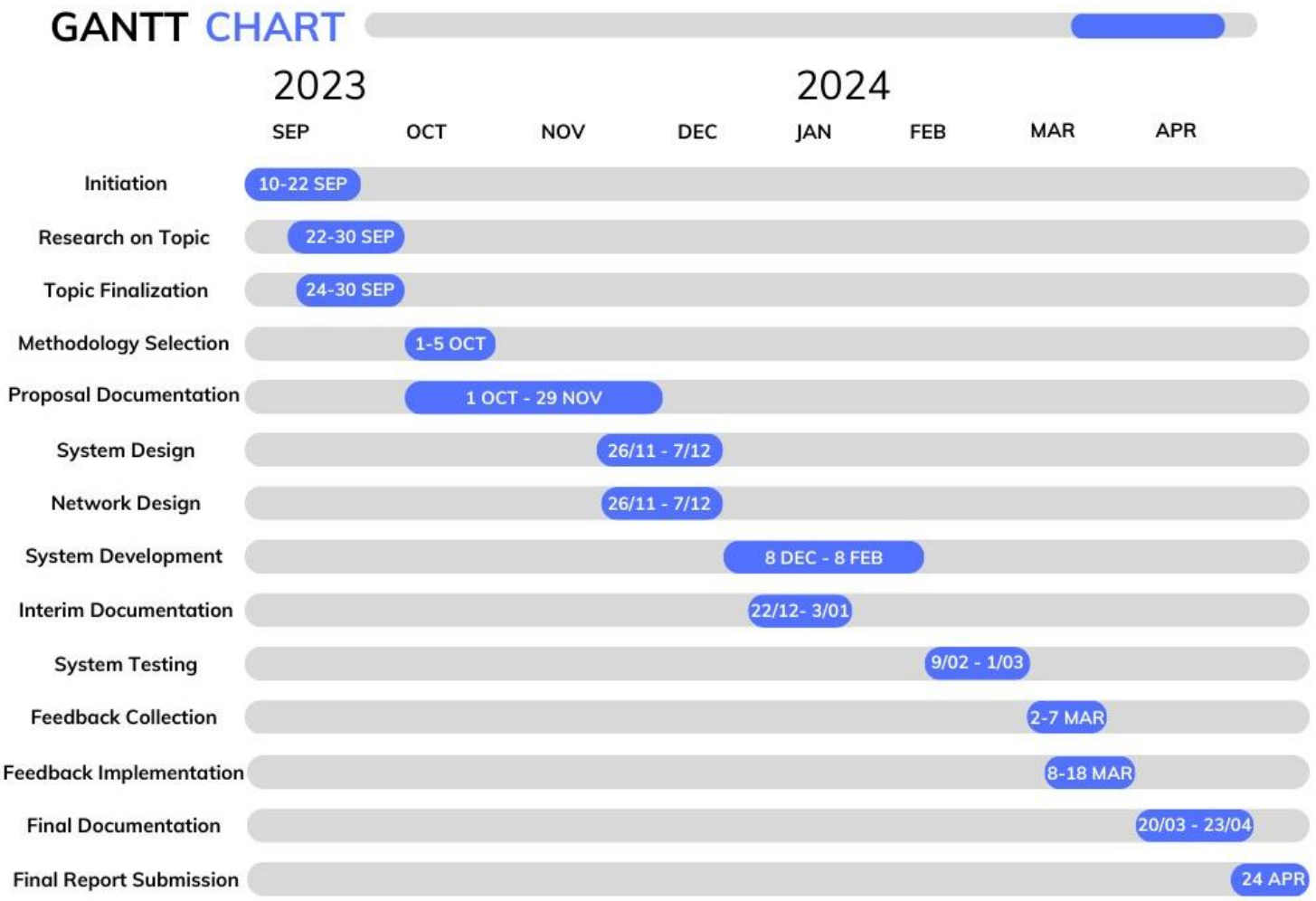


Figure 162 Gantt Chart

9.5.4. System & Network Diagram

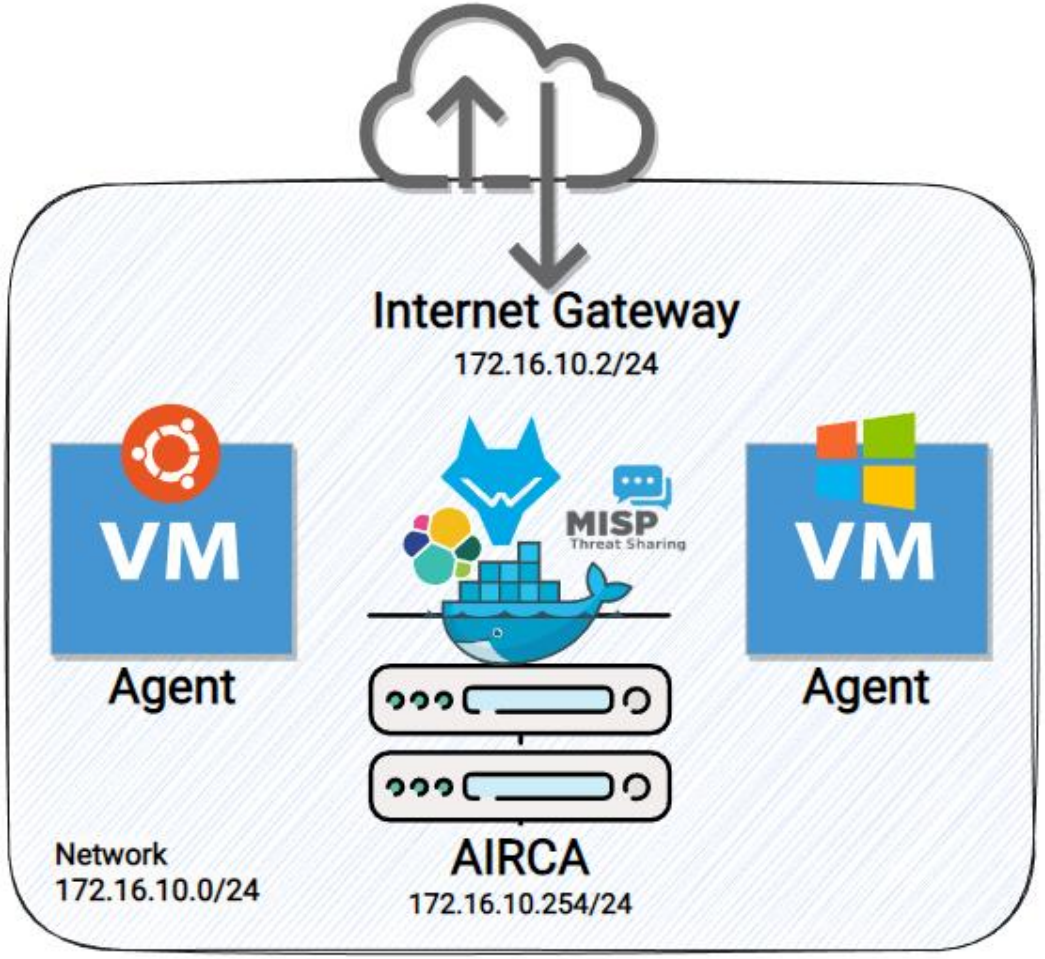


Figure 163 System and Network Architecture Diagram

[*Note: The system & network diagram above was created for this project in reference to resources that was created and configured inside VMware which is a hypervisor application and was used for the overall development, deployment and testing of the systems.*]

9.6. Appendix F: Progress Review Table

S.N.	Task Description	Status (Percent %)	Completed Time
Phase 1: Pre-Project Phase			
1.	Research and finalize the project	Completed (100%)	On-time
Phase 2: Project - Life Cycle Phase : Feasibility Study + Business Study			
2.	Analyse Feasibility	Completed (100%)	On-time
3.	Research on problem scenario and solution	Completed (100%)	On-time
4.	Define aim and objectives	Completed (100%)	On-time
5.	Figure out risks and their contingency plans	Completed (100%)	On-time
6.	Proposal review and submission	Completed (100%)	On-time
Phase 2: Project - Life Cycle Phase : Functional Model Iteration + System Design and Build Iteration			
7.	System and Network Design	Completed (100%)	On-time
8.	Setup servers and endpoints	Completed (100%)	Early
9.	Deploy Wazuh and MISP	Completed (100%)	Early
10.	Setup agents for the endpoints	Completed (100%)	Early
11.	Integrate Wazuh with MISP	Completed (100%)	On-time
12.	Clean up possible false positive/duplicate threat intel feeds	Completed (100%)	On-time
13.	Containerize the system	Completed (100%)	On-time
14.	Interim report documentation	Completed (100%)	On-time
15.	Add rules for threat detection	Completed (100%)	On-time
16.	Write automation scripts for active response	Completed (100%)	Delayed
Phase 2: Project - Life Cycle Phase : Implementation			
17.	Testing and Review	Completed (100%)	Delayed
18.	Feedback Collection and Implementation	Completed (100%)	Delayed
Phase 3: Post-Project Phase			
19.	Final Documentation	Completed (100%)	On-time
20.	Project Submission	Completed (100%)	On-time

Table 27 Project Progress Table