

Automated Threat Detection and Response using an Open-Source SIEM

Capstone Project

Phase – I Report

Submitted by

22BCY10196 REHAN RAJESH

22BCY10182 MADHAN MOHAN NAIDU .M

22BCY10257 V.A.AASWIN

22BCY10282 INSAF SADIK A S

in partial fulfillment for the award of the degree

Of

BATCHELOR OF TECHNOLOGY

in

**COMPUTER SCIENCE AND ENGINEERING
(Cyber Security and Digital Forensics)**



VIT[®]
BHOPAL
www.vitbhopal.ac.in

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRI KALAN,SEHORE

MADHYAPRADESH - 466114

OCT - 2025

**VIT BHOPAL UNIVERSITY, KOTRIKALAN,
SEHORE
MADHYA PRADESH – 466114**

Bonafide Certificate

Certified that this project report titled “**Automated Threat Detection and Response using an Open-Source SIEM**” is the bonafide work of “**22BCY10196 REHAN RAJESH, 22BCY10182 MADHAN MOHAN NAIDU .M, 22BCY10257 V.A.AASWIN, 22BCY10282 INSAF SADIK A S**” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROJECT SUPERVISOR

Dr.S. Harihara Sitaraman,

Associate Professor

School of Computing Science

and Artificial Intelligence,

VIT Bhopal University.

PROGRAM CHAIR

Dr. D. Saravanan,

Assistant Professor Sr.,

School of Computing Science

and Artificial Intelligence,

VIT Bhopal University.

The Capstone Project Examination is held on _____

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to **Dr D. Saravanan**, Program Chair, Cyber Security and Digital Forensics for much of his valuable support and encouragement in carrying out this work.

I would like to thank my supervisor **Dr HARIHARASITARAMAN.S**, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF ABBREVIATIONS

Abbreviation	Full Form
API	Application Programming Interface
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
EDR	Endpoint Detection and Response
ELK	Elasticsearch, Logstash, Kibana
GUI	Graphical User Interface
HDD	Hard Disk Drive
IP	Internet Protocol
LTS	Long-Term Support (for Ubuntu)
MISP	Malware Information Sharing Platform
NAT	Network Address Translation
PAM	Pluggable Authentication Modules
RAM	Random Access Memory
SIEM	Security Information and Event Management
SOAR	Security Orchestration, Automation, and Response
SOC	Security Operations Center
SSH	Secure Shell
VM	Virtual Machine

LIST OF FIGURES AND GRAPHS

Figure No.	Title	Page No
Fig 1.1	Oracle VirtualBox Manager with Ubuntu and Kali Linux VMs	18
Fig 1.2	VirtualBox Network Settings Showing "WazuhLabNet" NAT Network	18
Fig 1.3	Wazuh Server Services Operational Status	19,20
Fig 1.4	Wazuh Agent Service Status on Kali Linux	21
Fig 1.5	Verified Agent Connection in Wazuh Dashboard	21
Fig 1.6	Active Response Configuration in ossec.conf	22
Fig 1.7	Hydra Brute-Force Attack Execution	23
Fig 1.8	Real-time Detection of Individual Failed Logins (Rule 5760)	24

LIST OF TABLES

Table No.	Description	Page No.
1	Information About Technology Used	12

ABSTRACT

This project demonstrates the implementation of a robust and cost-effective Security Information and Event Management (SIEM) system for real-time threat detection and automated response. The primary motivation is the increasing volume and sophistication of cyber threats, which necessitate automated security solutions to reduce response times and enhance operational efficiency. By leveraging open-source technologies, this project provides a scalable alternative to expensive commercial SIEMs, making advanced security monitoring accessible. The core of the system is built on Wazuh, an open-source security platform that combines SIEM and Endpoint Detection and Response (EDR) capabilities.

The existing approach to security monitoring often involves manual log review or costly commercial tools with high licensing fees. Manual processes are slow, prone to human error. Commercial solutions, while powerful, can be prohibitively expensive for many organizations. These drawbacks create a significant gap in security posture, leaving systems vulnerable to delayed threat detection and response.

This project proposes a fully functional SIEM built within a virtualized lab environment using Oracle VirtualBox. The solution uses a Wazuh server on an Ubuntu Server 22.04 virtual machine to monitor a Kali Linux endpoint. We successfully demonstrate the system's ability to detect a simulated SSH brute-force attack, launched via the Hydra toolkit, in real-time. The system correctly generated low-level alerts (Rule 5760) for each individual failed login, which were then correlated to trigger a single, high-severity alert (Rule 5720) confirming the attack pattern. Furthermore, we implement an automated "Active Response" mechanism that instantly blocks the attacker's IP address upon detection of this sustained attack. This project successfully proves the viability

of using open-source tools to create an enterprise-grade, automated security monitoring and defense system.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	Bonafide Certificate	1
	Acknowledgement	2
	List of Abbreviations	3
	List of Figures and Graphs	4
	List of Tables	5
	Abstract	6
1	Project Description and Outline 1.1 Background and Context 1.2 Problem Statement 1.3 Scope and Objectives 1.4 Report Organization	9 9 9 10 10
2	Related Work Investigation 2.1 Initial Technical Approach 2.2 Strategic Pivot and Literature Survey 2.3 Rationale for Final Technology Selection	11 11 11 12
3	Requirement Artifacts 3.1 Hardware and Software Requirements	12 12,13,14,15
4	Design Methodology and its Novelty 4.1 Design Methodology: A Layered Approach 4.2 Novelty of the Approach 4.3 Logical Architecture of the SIEM	15 15 16 16,17

5	Technical Implementations and Analysis 5.1 Technical Implementation: Step-by-Step 5.2 Analysis: Testing and Results	17 17,18,19 20,21,22,23,24
6	Project Outcome and Applicability 6.1 Project Outcome 6.2 Applicability 6.3 Key Learnings 6.4 Limitations of the Current Work	24 24 24 25 25
7	Conclusions and Recommendations 5.1 Conclusions 5.2 Recommendation for Future Work: Integration with a SOAR Platform	26 26 26
	References Appendices	27 27 28,29,30

Chapter 1: Project Description and Outline

1.1 Background and Context

In the contemporary digital ecosystem, the security of information systems is paramount. Organizations of all sizes are increasingly reliant on digital infrastructure, making them attractive targets for a wide array of cyber threats. One of the most common and persistent attack vectors is the brute-force attack, a method used to gain unauthorized access by systematically guessing credentials. These attacks can range from simple dictionary attacks, which use common words, to more sophisticated credential stuffing, which uses lists of credentials stolen from other data breaches. A successful brute-force attack can lead to data breaches, ransomware deployment, and significant reputational and financial damage. The high volume of log data generated by modern IT environments makes manual detection of such intrusive activities practically impossible.

To address this challenge, the field of cybersecurity developed Security Information and Event Management (SIEM) systems. A SIEM provides a holistic approach to security by aggregating log data from disparate sources, identifying and correlating events to detect suspicious activity, and providing a centralized platform for threat monitoring and incident response. This project details the design, implementation, and validation of a fully functional SIEM using the open-source platform, Wazuh, to demonstrate an effective, automated defense against common cyber threats.

1.2 Problem Statement

Many organizations, particularly small to medium-sized enterprises (SMEs), face a significant challenge in implementing robust security monitoring. Commercial SIEM solutions are often prohibitively expensive, involving high licensing costs and requiring specialized personnel. This leaves a critical security gap where organizations are unable to effectively detect and respond to threats in a timely manner. The lack of an automated defense mechanism means that by the time a breach is manually discovered, significant damage may have already occurred. There is a pressing need for a cost-effective, scalable, and automated security solution that can provide enterprise-grade protection without the associated financial burden.

1.3 Scope and Objectives

The scope of this project is to create a proof-of-concept SIEM within a controlled virtual lab environment. The primary objective is to build and demonstrate a system capable of automated threat detection and response.

The specific objectives are:

- To design a secure, isolated virtual network for deploying and testing the SIEM.
- To deploy and configure the Wazuh server, including its core components: the Manager, Indexer, and Dashboard.
- To integrate a Linux endpoint by installing and configuring a Wazuh agent to forward security data.
- To simulate a realistic SSH brute-force attack to validate the system's detection capabilities.
- To implement and test an automated Active Response mechanism to block the attacker's IP address, demonstrating a full security cycle from detection to remediation.

1.4 Report Organization

This report is structured into seven chapters, each detailing a critical phase of the project.

Chapter 1 provides an introduction to the project, outlining the problem and objectives.

Chapter 2 details the project's execution timeline, including the initial research on the ELK stack, the challenges faced, and the strategic pivot to Wazuh.

Chapter 3 presents a detailed literature survey of SIEM technologies, providing a comparative analysis of different platforms and the final rationale for selecting Wazuh.

Chapter 4 describes the system's design, covering the hardware and software requirements, the network topology, and the detailed logical architecture of the Wazuh platform.

Chapter 5 provides a granular, step-by-step account of the technical implementation and configuration of the entire system, from the virtual environment to the Active Response rule.

Chapter 6 presents the testing methodology, a deep-dive analysis of the generated security alerts, and a discussion of the project's key learnings.

Chapter 7 concludes the report, summarizes the project outcomes and their applicability, and provides detailed recommendations for future work.

Chapter 2: Related Work Investigation

2.1 Initial Technical Approach

(Review 1): The ELK Stack

The project initially aimed to build a SIEM by manually integrating the components of the **Elastic Stack (ELK)**: Elasticsearch, Logstash, and Kibana. The plan was to use Logstash for log collection and parsing, Elasticsearch for storage and indexing, and Kibana for visualization. For the initial review, we explored this path by setting up the virtual lab and installing the basic ELK components.

Our work included researching methods to enrich the collected log data. For example, we investigated writing scripts that could extract an IP address from a log and query external threat intelligence APIs like VirusTotal (to check for malicious associations) and Shodan (to identify open ports and services). We also explored how custom alert rules could be created within the Kibana UI, for instance, a rule that would trigger if the number of failed logins from a single IP exceeded a certain threshold (e.g., > 500).

However, during this phase, we encountered significant challenges. Creating custom parsing rules in Logstash to correctly structure SSH logs was complex. Furthermore, building a robust alerting and automated response system on top of the base ELK stack would require significant custom scripting and the integration of third-party tools like ElastAlert. This "build-it-yourself" approach was deemed too complex and high-risk for the project's timeframe.

2.2 Strategic Pivot and Literature Survey

Following Review 1, a strategic decision was made to pivot to a more integrated, security-focused platform. This led to a literature survey of available open-source SIEM solutions.

ELK Stack Analysis: While incredibly powerful for log aggregation and search, the base ELK stack lacks the specialized security features of a true SIEM. Alerting, rule correlation,

and automated response are not core features and require significant manual effort or paid extensions.

Wazuh Analysis: Wazuh is an open-source security platform that has gained significant traction. It integrates SIEM and EDR capabilities. Its primary advantages are that it is free, highly customizable, and comes with a pre-built, extensive security rule set and a native Active Response framework.

2.3 Rationale for Final Technology Selection

The decision to select Wazuh was based on a comparative analysis of its ability to meet the project's core objective: demonstrating an automated detection and response cycle. Wazuh is a SIEM solution, whereas the ELK stack is a data platform that can be used to build a SIEM. For the scope of this project, Wazuh provided a faster, more direct, and more robust path to achieving our goals.

Chapter 3: Requirement Artifacts

3.1 Hardware and Software Requirements

The entire system was built using readily available software and modest hardware allocations within a virtual environment.

Table 1: Information about Technology Used

Component	Technology Used	Configuration	Purpose
Hypervisor	Oracle VM VirtualBox 7.0	N/A	To create and manage virtual machines.
Server VM	Ubuntu Server 22.04 LTS	8 GB RAM, 4 CPU Cores, 50 GB HDD	Hosts the Wazuh server components.
Endpoint VM	Kali Linux	4 GB RAM, 2 CPU Cores, 30 GB HDD	Acts as the monitored endpoint and attack source.
SIEM Platform	Wazuh 4.7.4	All-in-one deployment	The core security monitoring platform.
Attack Tool	THC-Hydra	v9.0	To simulate the SSH brute-force attack.

1. Hypervisor (Oracle VM VirtualBox)

A Hypervisor is a crucial piece of software that creates and runs virtual machines (VMs). Think of it as a "host" or a "manager" that allows you to run multiple, separate operating systems on a single physical computer. Your physical machine's resources (like CPU, RAM, and storage) are shared among these virtual machines.

- **Oracle VM VirtualBox:** This is a specific type of hypervisor known as a Type 2 hypervisor. This means it runs on top of a conventional operating system (like Windows, macOS, or Linux) as an application. This makes it very easy to set up and is ideal for development, testing, and educational purposes like your project. In your setup, VirtualBox is the foundation that allows both the Ubuntu Server and the Kali Linux machines to exist and run simultaneously on one computer.

2. Server VM (Ubuntu Server 22.04 LTS)

A **Virtual Machine (VM)** is a digital version of a physical computer. It behaves like a completely separate computer with its own operating system and applications, but it runs on software (the hypervisor) instead of physical hardware.

- **Server VM:** In your project, this VM is configured to act as a **server**. A server is a computer that provides services, data, or resources to other computers, known as clients, over a network.
- **Ubuntu Server 22.04 LTS:** This is the operating system chosen for your server.
 - **Ubuntu Server** is a popular Linux distribution known for its stability and strong community support.
 - **LTS** stands for **Long-Term Support**. This means it will receive security updates, bug fixes, and maintenance from its creators for a long period (usually five years), making it a reliable choice for servers.
- **Configuration (8 GB RAM, 4 CPU Cores, 50 GB HDD):** These specs define the virtual hardware dedicated to this server. It has been allocated 8 gigabytes of memory, access to 4 processor cores from the host machine, and a 50-gigabyte virtual hard drive. This configuration is robust enough to run the Wazuh SIEM platform and handle incoming network traffic without performance issues.

3. Endpoint VM (Kali Linux)

An **Endpoint** is any device connected to a computer network. This can include desktops, laptops, servers, or smartphones. In cybersecurity, endpoints are often the targets of attacks, as they are the entry points to a network.

- **Endpoint VM:** In your project, this VM is set up to be the "attacker" machine. It acts as the source of the simulated threat.
- **Kali Linux:** This is a specialized Linux distribution designed for digital forensics and **penetration testing** (often called "pen testing"). It comes pre-installed with hundreds of tools used for security auditing, vulnerability analysis, and, in this case, simulating attacks. By using Kali Linux, you have easy access to tools like THC-Hydra to launch the brute-force attack against your server VM.
- **Configuration (4 GB RAM, 2 CPU Cores, 30 GB HDD):** This configuration is sufficient for running the Kali Linux OS and the attack tools needed for the project.

4. SIEM Platform (Wazuh)

SIEM stands for **Security Information and Event Management**. A SIEM platform is a sophisticated security solution that provides a holistic view of an organization's information security. Its primary job is to collect log data and event information from various sources across the network (like servers, endpoints, firewalls), correlate this data, and analyze it in real-time to identify potential security threats or suspicious activities. 🕵️

- **Wazuh:** This is a popular **open-source** security platform that combines SIEM and XDR (Extended Detection and Response) capabilities. It's used for threat detection, integrity monitoring, incident response, and compliance.
- **All-in-one deployment:** This means all the core components of Wazuh (the Wazuh server, the Wazuh indexer, and the Wazuh dashboard) are installed on a single host—your Ubuntu Server VM. This simplifies the setup for smaller environments or projects like yours. In this project, Wazuh is configured to monitor the logs on the Ubuntu server. When the SSH brute-force attack begins, it will generate many failed login attempts. Wazuh will detect this abnormal pattern, analyze it, and generate a security alert, demonstrating its core purpose.

5. Attack Tool (THC-Hydra)

An **Attack Tool** is a software program designed to find and exploit security vulnerabilities in computer systems or networks. While malicious hackers use them for illegal activities, cybersecurity professionals (known as ethical hackers) use them in controlled environments to test defenses and identify weaknesses before criminals can exploit them.

- **THC-Hydra:** Often just called **Hydra**, this is a very popular and powerful network logon cracker. It is designed to perform **brute-force** or **dictionary attacks** against various network protocols and services. A brute-force attack involves systematically trying all possible combinations of passwords until the correct one is found.
- **Purpose (Simulate SSH brute-force attack):** In your project, Hydra is used to launch a simulated attack against the **SSH (Secure Shell)** service running on the Ubuntu Server VM. SSH is a protocol used for secure remote login and command execution. By using Hydra to flood the server with thousands of login attempts, you are creating the exact type of "noisy" security event that a SIEM system like Wazuh is designed to detect and alert on.

Chapter 4: Design Methodology and its Novelty

4.1 Design Methodology: A Layered Approach

The design of the system followed a layered, defense-in-depth approach within a virtualized environment.

- **Infrastructure Layer:** Oracle VirtualBox provided the foundational layer, allowing for the creation of a sandboxed environment.
- **Network Layer:** A private NAT Network was designed to create a secure, isolated subnet for inter-VM communication, mimicking a protected internal network segment.
- **Platform Layer:** The Wazuh server was designed to be the central hub, acting as the "brain" for all security operations.

- **Endpoint Layer:** The Wazuh agent was designed to be the "sensor" on the monitored asset, collecting and forwarding critical security data.

4.2 Novelty of the Approach

The novelty of this project does not lie in the creation of a new algorithm, but in the practical and comprehensive integration of powerful open-source tools to create a fully autonomous security monitoring and defense cycle. While individual components like log shippers and firewalls are common, this project demonstrates their seamless orchestration. The key novelty is in creating an accessible blueprint for a system where a threat is not just detected and logged, but is detected, correlated, identified as a sustained attack, and automatically neutralized without any human intervention. This demonstrates an enterprise-grade security capability using entirely cost-free, open-source technologies.

4.3 Logical Architecture of the SIEM

The Wazuh platform is not a single application but a suite of components that work in concert:

1. The Wazuh Agent: The Endpoint Sensor

The Wazuh agent is a lightweight, multi-platform piece of software installed on the endpoints to be monitored (in our case, the Kali Linux VM). It is the source of all security data. Its key capabilities include:

- **Log Data Collection:** It actively reads event logs from the operating system and applications (e.g., /var/log/auth.log) and securely forwards them to the Wazuh Manager for analysis.
- **File Integrity Monitoring (FIM):** It can monitor specific files and directories for changes, alerting when a critical system file is modified, which could indicate a compromise.
- **Rootkit and Malware Detection:** It performs scans to detect known rootkits and malware signatures.
- **Security Configuration Assessment (SCA):** It scans the endpoint against pre-defined security baselines (based on CIS benchmarks) to find misconfigurations.

2. The Wazuh Manager: The Central Brain

Wazuh Manager is the central component where all analysis occurs.

Analysis Engine: It receives log data from the agents and uses a multi-stage process to analyze it. First, a decoder identifies the type of log (e.g., SSHD, Apache). Then, it compares the decoded data against its rule set to look for patterns that match specific security events.

Rule Correlation: This is its most powerful feature. It can correlate multiple, seemingly unrelated events over time to identify a complex attack pattern, as demonstrated by our brute-force test case.

Alert Generation: When a rule is matched, the Manager generates a formatted, enriched alert in JSON format.

Active Response Trigger: It is responsible for initiating the Active Response actions when an alert meets the configured criteria.

3. The Wazuh Indexer and Dashboard: The Data and Visualization Layers

Indexer: An optimized database that stores all the alerts generated by the Manager.

Dashboard: The web interface that allows analysts to query, visualize, and interact with the alert data stored in the Indexer.

Chapter 5: Technical Implementations and Analysis

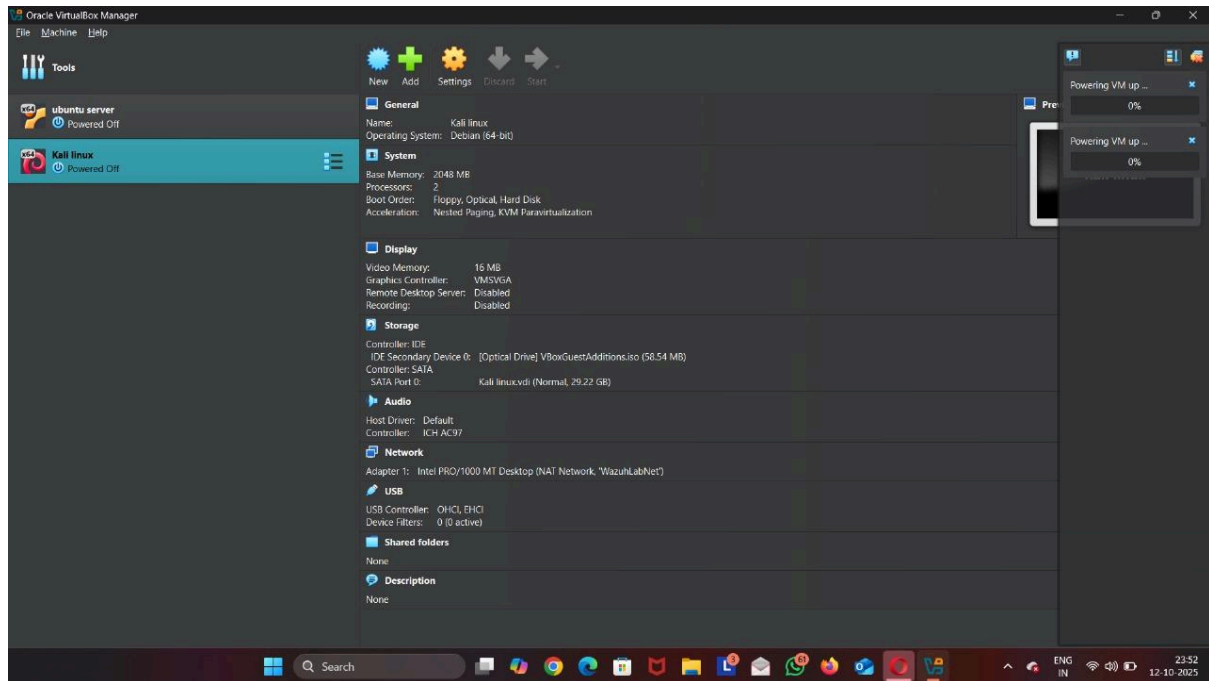
5.1 Technical Implementation: Step-by-Step

Phase 1: Virtual Environment Setup

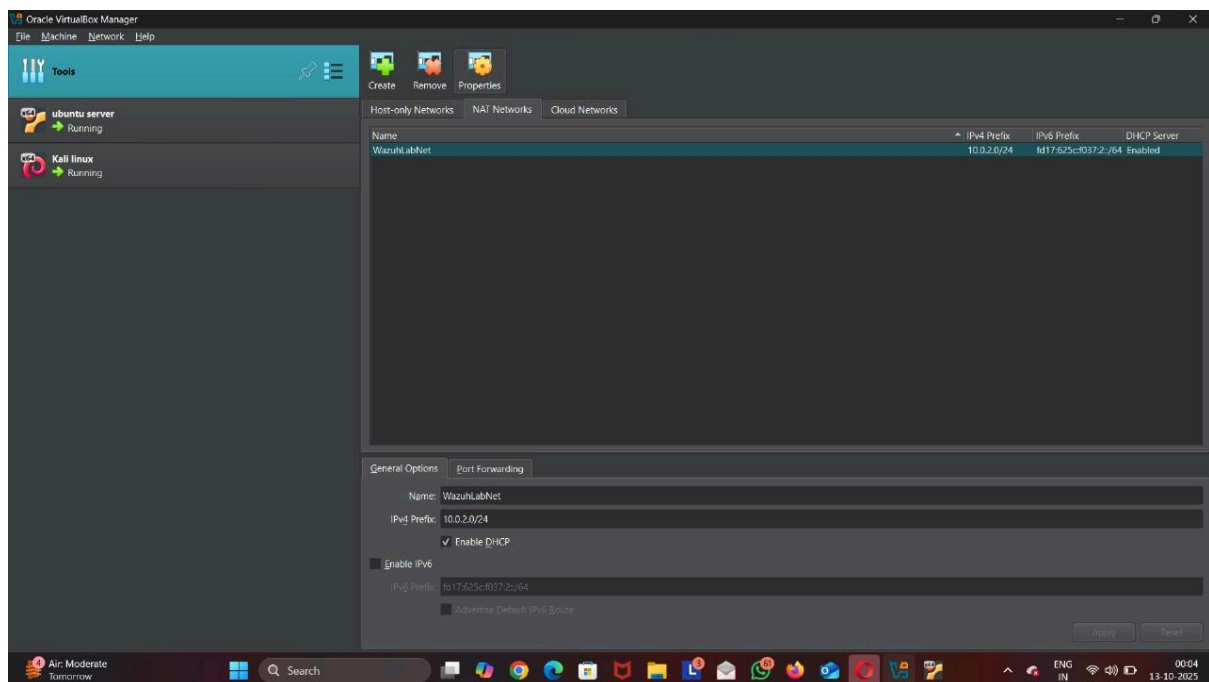
- **VM Creation and OS Installation:** Two new virtual machines were created. The Ubuntu Server was configured first to establish a baseline. The Kali Linux VM was created subsequently.
- **Initial Server Configuration:** Before deploying Wazuh, the Ubuntu server was prepared. A static IP address (10.0.2.4) was configured to ensure a stable endpoint for the agent. The system was then fully updated using `sudo apt update && sudo apt upgrade -y` to apply all security patches.
- **Network Configuration and Testing:** Both VMs were shut down and their network adapters in VirtualBox were changed from the default "NAT" to "NAT Network," and "WazuhLabNet" was selected. After booting, connectivity was tested using ping

10.0.2.5 from the server and vice-versa to ensure the network was functioning correctly.

(Figure 1.1:Screenshot of oracle virtual box containing of ubuntu server and kali linux)



(Figure 1.2: Screenshot of the NAT Network and “WazuhLabNet” selected)



Phase 2: Wazuh Server Deployment

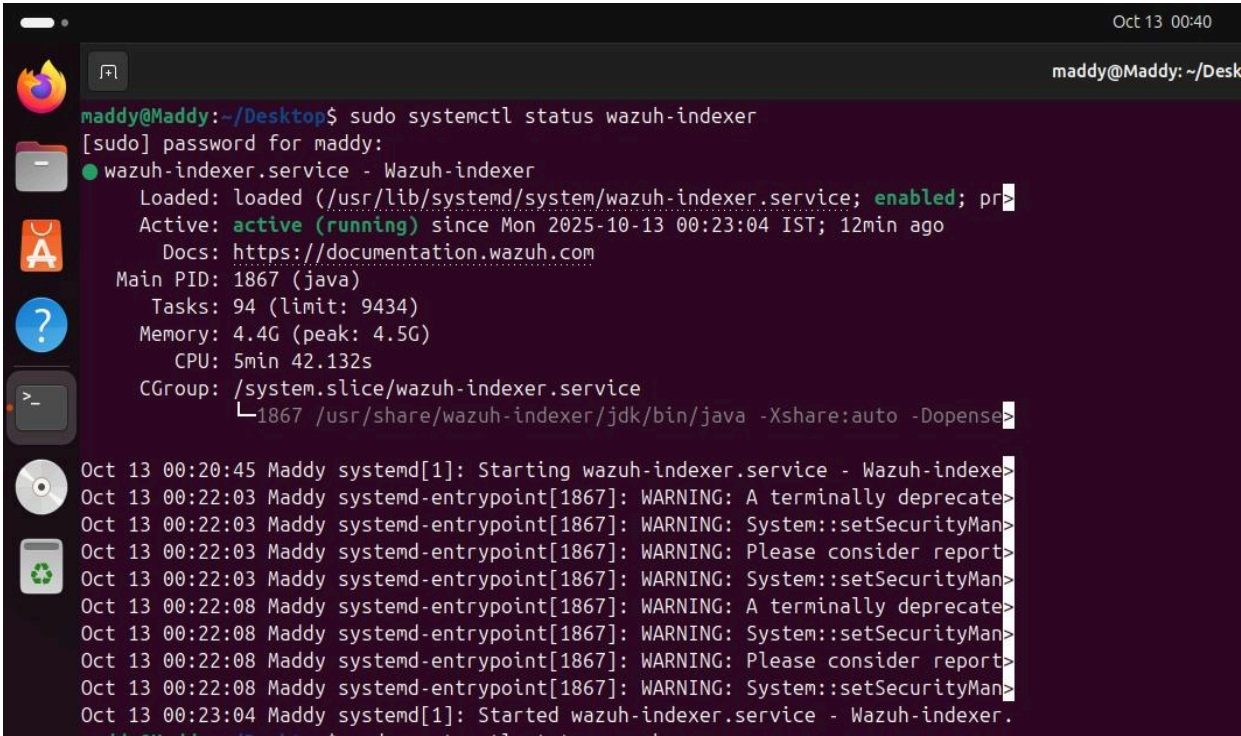
Prerequisites: Before installation, the system was updated using `sudo apt update && sudo apt upgrade -y`.

Script Download and Execution: The installation script was downloaded via `curl`. The script was then executed with `sudo bash ./wazuh-install.sh -a`, which stands for "all-in-one." This method is ideal for single-server deployments and automates the complex process of installing the Wazuh Indexer, Manager, and Dashboard, along with generating the necessary TLS certificates for secure communication between them.

Service Verification: After the script completed, it provided a summary with the randomly generated password for the admin user. The status of each service was then checked meticulously.

(Figure 1.3: Screenshot of the terminal output showing the status of all three Wazuh services)

(Wazuh-indexer)



```
maddy@Maddy: ~/Desktop$ sudo systemctl status wazuh-indexer
[sudo] password for maddy:
● wazuh-indexer.service - Wazuh-indexer
   Loaded: loaded (/usr/lib/systemd/system/wazuh-indexer.service; enabled; pr>
   Active: active (running) since Mon 2025-10-13 00:23:04 IST; 12min ago
     Docs: https://documentation.wazuh.com
    Main PID: 1867 (java)
      Tasks: 94 (limit: 9434)
     Memory: 4.4G (peak: 4.5G)
        CPU: 5min 42.132s
    CGroup: /system.slice/wazuh-indexer.service
           └─1867 /usr/share/wazuh-indexer/jdk/bin/java -Xshare:auto -Dopense>

Oct 13 00:20:45 Maddy systemd[1]: Starting wazuh-indexer.service - Wazuh-indexe>
Oct 13 00:22:03 Maddy systemd-entrypoint[1867]: WARNING: A terminally deprecate>
Oct 13 00:22:03 Maddy systemd-entrypoint[1867]: WARNING: System::setSecurityMan>
Oct 13 00:22:03 Maddy systemd-entrypoint[1867]: WARNING: Please consider report>
Oct 13 00:22:03 Maddy systemd-entrypoint[1867]: WARNING: System::setSecurityMan>
Oct 13 00:22:08 Maddy systemd-entrypoint[1867]: WARNING: A terminally deprecate>
Oct 13 00:22:08 Maddy systemd-entrypoint[1867]: WARNING: System::setSecurityMan>
Oct 13 00:22:08 Maddy systemd-entrypoint[1867]: WARNING: Please consider report>
Oct 13 00:22:08 Maddy systemd-entrypoint[1867]: WARNING: System::setSecurityMan>
Oct 13 00:23:04 Maddy systemd[1]: Started wazuh-indexer.service - Wazuh-indexer.>
maddy@Maddy: ~/Desktop$ sudo systemctl status wazuh-manager
```

(Wazuh-manager)

Agent Deployment Command Generation: We logged into the Wazuh Dashboard using the credentials provided at the end of the server installation. We navigated to the Wazuh menu -> Agents -> Deploy new agent. This interface provides a simple wizard for generating a deployment script tailored to the target operating system. We selected "DEB" for our Debian-based Kali Linux.

(Figure 1.4: Screenshot of the terminal output showing the agent service is active)

(Figure 1.5: Screenshot of the Wazuh Dashboard showing the active Kali agent)

Phase 4: Active Response Configuration

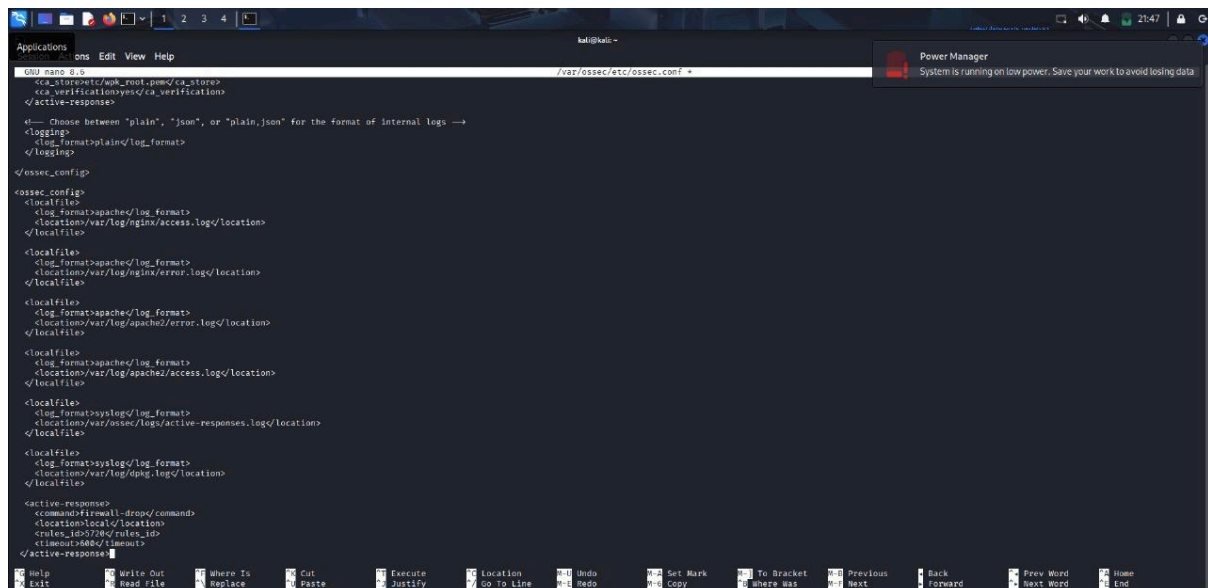
Configuration File Access: The file `/var/ossec/etc/ossec.conf` is the heart of the Wazuh Manager's configuration. It is an XML file that defines how the manager behaves.

Rule Definition: We added a new `<active-response>` block within the main `<ossec_config>` tags. This block acts as a trigger-action rule:

- `<command>firewall-drop</command>`: Specifies the name of the script to execute. `firewall-drop` is a built-in Wazuh script that automatically adds a iptables rule to drop packets from the source IP of the alert.
- `<location>local</location>`: Tells the manager to execute the command on the server itself.
- `<rules_id>5720</rules_id>`: This is the trigger. The response will only activate when an alert with this specific ID is generated.
- `<timeout>600</timeout>`: The duration, in seconds, for which the IP will be blocked.

Service Restart: Any change to `ossec.conf` requires a restart of the manager service to be loaded into memory.

(1.6: Screenshot of the `<active-response>` block within the main `<ossec_config>`)



```
GNU nano 8.0 /var/ossec/etc/ossec.conf
<ca_store>etc/wpk_root.pem</ca_store>
<ca_verification>yes</ca_verification>
</active-response>

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

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

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

  <localfile>
    <log_format>apache</log_format>
    <location>var/log/apache2/error.log</location>
  </localfile>

  <localfile>
    <log_format>apache</log_format>
    <location>var/log/apache2/access.log</location>
  </localfile>

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

  <localfile>
    <log_format>syslog</log_format>
    <location>var/log/dpkg.log</location>
  </localfile>

  <active-response>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>5720</rules_id>
    <timeout>600</timeout>
  </active-response>
</ossec_config>
```

5.2 Analysis: Testing and Results

Attack Execution:

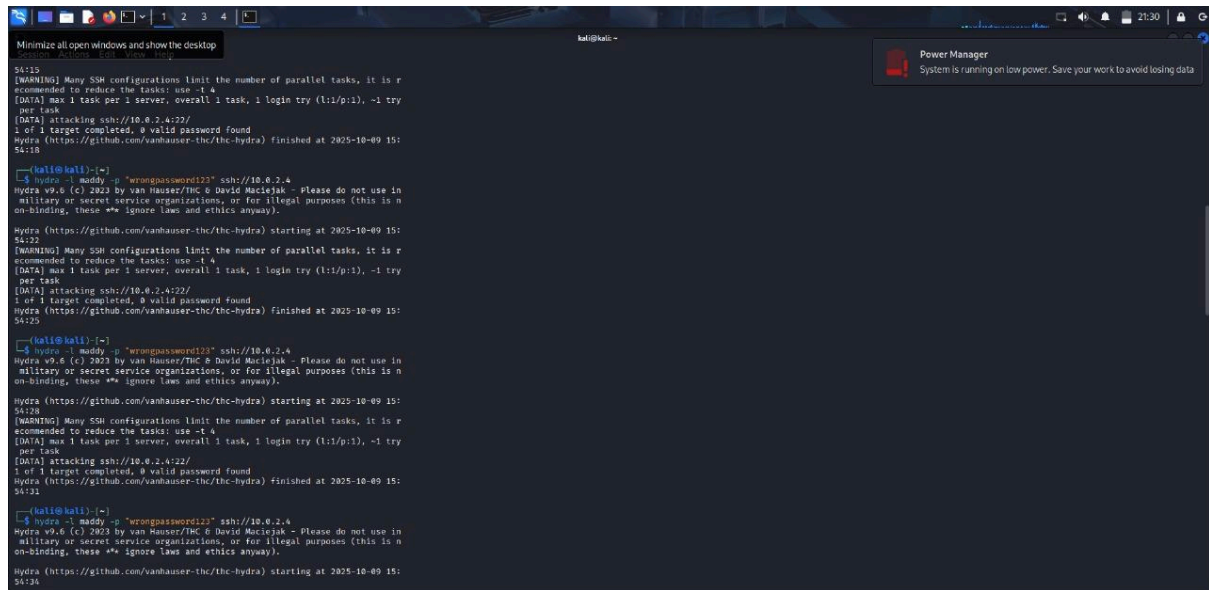
The Hydra tool was used to launch the attack. The command `hydra -l maddy -P passlist.txt ssh://10.0.2.4` has several key flags:

`-l maddy`: Specifies the target username.

`-P passlist.txt`: Specifies the path to the password file to use for the dictionary attack.

ssh://10.0.2.4: Specifies the target protocol and IP address.

(1.7: Screenshot of the Hydra command running in the Kali terminal)



```
Minimize all open windows and show the desktop
kali@kali: ~
$ hydra -l maddy -p "wrongpassword123" ssh://10.0.2.4
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
military or secret service organizations, or for illegal purposes (this is n
on-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-09 15:
54:19
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:/p:1), ~1 try
per task
[DATA] attacking ssh://10.0.2.4:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-09 15:
54:18

kali@kali: ~$
$ hydra -l maddy -p "wrongpassword123" ssh://10.0.2.4
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
military or secret service organizations, or for illegal purposes (this is n
on-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-09 15:
54:22
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:/p:1), ~1 try
per task
[DATA] attacking ssh://10.0.2.4:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-09 15:
54:23

kali@kali: ~$
$ hydra -l maddy -p "wrongpassword123" ssh://10.0.2.4
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
military or secret service organizations, or for illegal purposes (this is n
on-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-09 15:
54:26
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:/p:1), ~1 try
per task
[DATA] attacking ssh://10.0.2.4:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-09 15:
54:31

kali@kali: ~$
```

Analysis of the Alert Generation Lifecycle:

To fully understand the results, it is useful to trace the path of a single failed login from raw log to final alert.

Raw Log Generation: For every failed attempt, the SSH service on the Ubuntu server writes a line to /var/log/auth.log like this:

Oct 13 23:30:45 ubuntu-server sshd[1234]: Failed password for maddy from 10.0.2.5 port 54321 ssh2

Agent Collection: The Wazuh agent on the server reads this new line in real-time and sends it to the Manager.

Decoding and Rule Matching: The Manager receives the log. Its sshd decoder parses the log into structured fields (e.g., src_ip: "10.0.2.5", user: "maddy"). The Manager then matches this decoded event to Rule 5760, generating a Level 5 alert.

The screenshot shows the Wazuh Security Alerts interface. The table displays the following data:

Time	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Oct 8, 2025 @ 19:18:59.517	000	Maddy	T1110.001 T1021.004	Credential Access, Lateral Movement	ssh: authentication failed.	5	5760
Oct 8, 2025 @ 19:18:57.510	000	Maddy	T1110.001	Credential Access	PAM: User login failed.	5	5503
Oct 8, 2025 @ 19:18:49.508	000	Maddy	T1110.001 T1021.004	Credential Access, Lateral Movement	ssh: authentication failed.	5	5760
Oct 8, 2025 @ 19:18:47.474	000	Maddy	T1110.001	Credential Access	PAM: User login failed.	5	5503
Oct 8, 2025 @ 19:18:43.471	000	Maddy	T1110.001 T1021.004	Credential Access, Lateral Movement	ssh: authentication failed.	5	5760
Oct 8, 2025 @ 19:18:41.470	000	Maddy	T1110.001	Credential Access	PAM: User login failed.	5	5503
Oct 8, 2025 @ 19:18:31.460	000	Maddy	T1110.001 T1021.004	Credential Access, Lateral Movement	ssh: authentication failed.	5	5760
Oct 8, 2025 @ 19:18:29.462	000	Maddy	T1110.001	Credential Access	PAM: User login failed.	5	5503
Oct 8, 2025 @ 19:16:03.174	000	Maddy	T1078	Defense Evasion, Persistence, Privilege Escalation, Initial Access	PAM: Login session opened.	3	5501
Oct 8, 2025 @ 19:14:26.013	000	Maddy			PAM: Login session closed.	3	5502

Rows per page: 10

Correlation and Escalation: After the 6th alert of this type from the same source IP (10.0.2.5) arrives within a short period, the Manager's correlation logic is satisfied, and it generates the single, high-severity Rule 5720 alert. This is the crucial intelligence that separates the signal from the noise.

Chapter 6: Project Outcome and Applicability

6.1 Project Outcome

The project successfully resulted in a fully functional, automated threat detection and response system. The key outcomes are:

- A working proof-of-concept SIEM built entirely on open-source software.
- Demonstrated real-time detection of a common cyberattack (SSH brute-force).
- Successful implementation of an automated response mechanism configured to block the attacker.
- A comprehensive understanding of the architectural components and operational workflow of a modern SIEM.

6.2 Applicability

The outcome of this project is highly applicable, particularly for small to medium-sized enterprises (SMEs), educational institutions, and research labs. These organizations often have limited budgets and cannot afford commercial SIEM solutions. This project provides a detailed, practical blueprint for them to implement a powerful, enterprise-grade security monitoring system at a fraction of the cost, significantly improving their security posture against common threats.

6.3 Key Learnings

This project provided several key practical learnings beyond the main objectives:

- **Importance of Integrated Systems:** The challenges faced with the initial ELK stack highlighted the significant advantages of using an integrated, security-focused platform like Wazuh, which handles much of the complex plumbing out of the box.
- **System Service Dependencies:** Troubleshooting the Wazuh Manager's startup failures taught us a valuable lesson in how system services depend on each other and the importance of ensuring a proper startup order.
- **The Power of Correlation:** Witnessing the system transform multiple low-level logs into a single, high-confidence alert provided a practical understanding of how a SIEM generates actionable intelligence.

6.4 Limitations of the Current Work

While the project successfully meets its objectives, it is important to acknowledge its limitations as a proof-of-concept:

Controlled Environment: The entire system operates within an isolated lab. Real-world networks present far greater complexity, including diverse operating systems, network segmentation, and higher volumes of log data, which would require further performance tuning.

Limited Scope of Response: The configured Active Response (firewall-drop) is host-based, meaning it is executed by the agent on the target machine itself. While effective for protecting that single server, it does not prevent the attacker from targeting other assets on the network. A more robust solution would integrate with network-level firewalls.

Default Rule Set: The system relies on the powerful default rule set provided by Wazuh. In a production environment, this would need to be augmented with custom rules tailored to the organization's specific applications and threat model.

Lack of Threat Intelligence: The current system does not integrate external threat intelligence feeds. It can identify that a brute-force attack is happening, but it cannot determine if the attacking IP is a known malicious actor or part of a larger botnet.

Chapter 7: Conclusions and Recommendation

7.1 Conclusion

This project successfully demonstrates the design, implementation, and validation of an effective, automated threat detection and response system using the open-source Wazuh platform. The successful pivot from an initial ELK-based approach to the more integrated Wazuh platform underscores a key finding: for achieving a full security cycle, a security-centric platform like Wazuh offers a more direct and robust solution. This work affirms the maturity and viability of open-source tools in building sophisticated, autonomous cybersecurity defenses.

7.2 Recommendation for Future Work: Integration with a SOAR Platform

The most significant recommendation for future work is to expand this system into a full Security Orchestration, Automation, and Response (SOAR) platform. This would involve integrating Wazuh with two other powerful open-source tools: TheHive and Cortex.

- **TheHive (The Incident Command Center):** High-severity Wazuh alerts would automatically create structured incident cases in TheHive for tracking and management.
- **Cortex (The Automated Analyst Assistant):** Cortex would automatically enrich these cases by querying threat intelligence feeds (building on our Review 1 research) and could execute advanced, orchestrated responses, such as blocking an IP on a network firewall or disabling a compromised user account.

This future integration would complete the full incident lifecycle, transforming the project into a comprehensive, semi-autonomous Security Operations Center (SOC) platform.

REFERENCES

- [1] Oracle Corporation, "Oracle VM VirtualBox," Oracle. [Online]. Available: <https://www.virtualbox.org/>. [Accessed: Oct. 13, 2025].
- [2] Wazuh, Inc., "Wazuh - The Open Source Security Platform," Wazuh. [Online]. Available: <https://wazuh.com/>. [Accessed: Oct. 13, 2025].
- [3] Canonical Ltd., "Ubuntu Server," Ubuntu. [Online]. Available: <https://ubuntu.com/server>. [Accessed: Oct. 13, 2025].
- [4] Offensive Security, "Kali Linux," Kali. [Online]. Available: <https://www.kali.org/>. [Accessed: Oct. 13, 2025].
- [5] T. H. C. van Hauser, "THC-Hydra," GitHub. [Online]. Available: <https://github.com/vanhauser-thc/thc-hydra>. [Accessed: Oct. 13, 2025].
- [6] TheHive Project, "TheHive - A Scalable, Open Source and Free Security Incident Response Platform," TheHive Project. [Online]. Available: <https://thehive-project.org/>. [Accessed: Oct. 13, 2025].
- [7] Cortex Project, "Cortex - A Powerful Observable Analysis and Active Response Engine," Cortex Project. [Online]. Available: <https://thehive-project.github.io/Cortex-Docs/>. [Accessed: Oct. 13, 2025].
- [8] Elastic, "The ELK Stack: Elasticsearch, Logstash, Kibana," Elastic. [Online]. Available: <https://www.elastic.co/what-is/elk-stack>. [Accessed: Oct. 13, 2025].

APPENDICES

(i) Program Code

The following code block was added to the Wazuh Manager's configuration file (/var/ossec/etc/ossec.conf) to enable the automated Active Response. This configuration instructs the manager to execute the firewall-drop script for 600 seconds (10 minutes) on any IP address that triggers the brute-force detection rule (5720).

(Use the "Code Format" add-in or screenshot method here to format this block)

XML

```
<ossec_config>
...
<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>5720</rules_id>
  <timeout>600</timeout>
</active-response>
...
</ossec_config>
```

(ii)Team Members Contribution

MADHAN MOHAN NAIDU .M (22BCY10182):

Madhan Mohan Naidu served as the technical lead and primary architect for the project. His contributions spanned the entire project lifecycle, from initial research to final implementation. He played a pivotal role in the strategic decision to pivot from the ELK Stack to the Wazuh platform after identifying key limitations during the initial review phase.

- Led the core implementation of the Wazuh server, including the installation of the Manager, Indexer, and Dashboard on the Ubuntu VM.
- Established and troubleshooted the connection between the server and the Kali Linux agent, ensuring a stable flow of log data.
- Designed and implemented the Active Response mechanism by modifying the ossec.conf file to automatically block attackers.
- Managed the primary technical tasks of building, configuring, and validating the end-to-end functionality of the final SIEM platform.

REHAN RAJESH (22BCY10196):

Rehan Rajesh was instrumental in the auxiliary development and configuration of the project's infrastructure. He was deeply involved in the initial research phase and the setup of the foundational environment upon which the SIEM was built.

- Contributed significantly to the initial Review 1 work, focusing on the ELK stack. This included researching Logstash configurations and exploring potential integrations with threat intelligence APIs like VirusTotal.
- Managed the setup and hardening of the Ubuntu Server environment, ensuring the base OS was stable and secure before the deployment of both the initial ELK stack and the final Wazuh platform.
- Assisted in troubleshooting network configurations within VirtualBox, resolving connectivity issues that were critical to the project's success.
- Played a key supporting role in the initial testing of the Wazuh agent's communication with the manager.

V.A.AASWIN (22BCY10257):

V.A.Aaswin was responsible for the comprehensive documentation, research consolidation, and compilation of this project report. His role was to translate the team's technical efforts into a clear, structured, and professional academic document.

- Gathered all technical details, methodologies, and results from both the initial ELK phase and the final Wazuh implementation.
- Structured and authored all chapters of the report, ensuring it followed the specified format and maintained a coherent narrative, including the detailed explanation of the strategic pivot.
- Researched and drafted the literature survey, comparing different SIEM technologies and providing the rationale for the team's final choice.
- Ensured the final report was clear, professional, and accurately reflected the team's collective efforts, challenges, and achievements.

INSAF SADIK A S (22BCY10282):

Insaf Sadik A S was in charge of system testing, validation, and threat simulation. His role was to act as the "red team," testing the efficacy of the implemented security measures and verifying the system's detection capabilities.

- Involved in the initial Review 1 testing, which included exploring how to create custom alert rules in a Kibana UI and scripting checks against external services like Shodan to identify open ports on a given IP.
- Designed and executed the final attack simulations for the Wazuh platform using the Hydra tool to perform the brute-force attacks.
- Meticulously monitored the Wazuh Dashboard during tests to verify that alerts were being generated correctly in real-time.
- Analyzed the different rule IDs triggered during the attack, confirming the successful correlation from low-level 5760 alerts to the high-severity 5720 alert, thereby validating that the project's objectives were fully met.

