DE MONTFORT UNIVERSITY DUBAI

MAY 2025

OPEN-SOURCE INTEGRATION OF SIEM, XDR, AND SOAR FOR CYBER DEFENSE

https://github.com/MeyyappanVenkatesh/Open-Source-SIEM-XDR-SOAR-Integration

CTEC3451D - DEVELOPMENT PROJECT
FINAL DELIVERABLE

CREATED BY: MEYYAPPAN VENKATESH - P2766441
SUPERVISOR: DR. ATHEER AL-MOUSA
MODULE LEADER: DR. ATHEER AL-MOUSA
WORD COUNT: 9756 (FROM TITLE PAGE TO APPENDIX EXCLUDING REFERENCE)

1 ABSTRACT

This project aims to design and implement an integrated, open-source cybersecurity framework capable of detecting and responding to real-world threats in real time. It addresses the critical need for affordable, integrated cybersecurity solutions for small-to-medium enterprises by developing a unified threat detection and response system. Leveraging open-source tools—Wazuh as a SIEM and XDR platform, Suricata for network-based intrusion detection, and Shuffle for SOAR automation—the framework provides real-time threat monitoring, automated incident response, and scalable security coverage without reliance on costly commercial products. The system was implemented in a VMware-based lab environment using Ubuntu endpoints and simulated attack scenarios including brute-force login attempts, malware drops, file system modifications, port scans, and data exfiltration.

Each use case was methodically validated through manual simulations, ensuring alert generation, response execution, and reporting were successful. Active responses such as automated IP blocking via iptables, malware quarantine with VirusTotal validation, and enriched email notifications were incorporated seamlessly. Custom Suricata rules, Auditd configurations, and Shuffle workflows allowed for proactive, targeted security measures.

Testing demonstrated that the unified system could detect, correlate, and respond to attacks with minimal manual intervention, aligning closely with the project's problem statement and objectives. Although effective against common attack vectors, limitations were observed in detecting encrypted exfiltration attempts and scalability was not stress-tested for large environments.

The findings confirm that modular, open-source cybersecurity solutions can meet operational needs affordably and efficiently. Future improvements could include integrating threat intelligence feeds, using lightweight machine learning for anomaly detection, and enhancing resilience against sophisticated evasion tactics. This work highlights the viability of open-source integration for practical, real-world cybersecurity defence.

2 TABLE OF CONTENTS

1 Abstract 1
3 List of Abbreviations and Glossary 4
4 Introduction 5
5 Literature review 6
5.1 Overview of Existing Research 6
5.2 Real-World implementation of SIEM, SOAR, and IDS integration 7
5.3 Critical analysis and theoretical framework 9
5.4 Justification for your work 9
6 Methodology 10
6.1 Research Design 10
6.2 Data Collection 10
6.3 Tools and Technologies 10
6.4 System/Network Architecture 11
6.5 Implementation Procedures 13
6.6 Security and Ethical Considerations 13
6.7 Validation and Testing 13
7 Implementation and Development 14
7.1 Development Process 14
7.2 Project Milestones 15
7.3 Use cases 16
7.4 Configuration and Implementation 17
7.4.1 Virtual Lab Setup 17
7.4.2 Wazuh Manager Installation and Endpoint Configuration 17
7.4.3 Suricata Deployment and Integration with Wazuh 20
7.4.4 Configuring Auditd for File Integrity Monitoring (FIM) 24
7.4.5 Shuffle SOAR Installation and Configuration 25
7.4.6 VirusTotal Integration with Wazuh 27
7.4.7 Monitoring a Directory for File Changes 28
7.4.8 Retrieving the API Key from Shuffler.io 28
7.4.9 Wazuh integration with Shuffle 29
7.4.10 UC01: Nmap Port Scan Detection 30
7.4.11 UC02: SSH Brute-Force Detection with Active Response 32
7.4.12 UC03: Malware Detection and Automated Response with VirusTotal and Shuffle 36
7.4.13 UC04: Logon monitoring with time-based alerting 41

7.4.14 UC05: File Integrity Monitoring and Ransomware Detection 44
7.4.15 UC06: Data Exfiltration Detection Using Suricata 47
7.4.16 UC07: Suspicious Command Monitoring and Data Exfiltration Detection 49
7.5 Challenges encountered 50
8 Test Plan 51
8.1 Test Cases 52
8.2 Results 54
8.3 Analysis 54
8.4 Risk Assessment 54
9 Discussion 55
9.1 Interpretation of Findings 55
9.2 Implications 55
9.3 Limitations 56
9.4 Recommendations for Future Work 56
10 Appendix 57
11 References and Bibliography 58

3 LIST OF ABBREVIATIONS AND GLOSSARY

Term / Abbreviation	Definition
SIEM	Security Information and Event Management – A system that collects, analyzes, and correlates security logs to detect threats.
SOAR	Security Orchestration, Automation, and Response – A platform that automates security workflows and responses.
XDR	Extended Detection and Response – A unified solution that integrates threat detection across multiple security layers.
IDS/IPS	Intrusion Detection/Prevention System – Monitors network traffic to detect or prevent unauthorized activity.
NAT	Network Address Translation – Enables multiple virtual machines to share one IP address while remaining isolated.
VM	Virtual Machine – A simulated computer environment that runs an operating system and applications on a physical host system.
Agent	A lightweight software component installed on a monitored system that collects logs and sends them to a central server for analysis.
API	Application Programming Interface – A set of functions allowing communication between software components.
MITRE ATT&CK	A standardized knowledge base of adversary tactics, techniques, and procedures used for threat detection and analysis.
Automation	The execution of tasks without human intervention, typically through scripts or workflows.
Webhook	A real-time notification mechanism that allows one system to send data to another via HTTP requests.
Endpoint	A device (such as a desktop, server, or VM) that is monitored for security events and activity.
Alert Correlation	The process of analysing and combining multiple alerts to identify meaningful patterns or real threats.
Log	A recorded event or system activity captured for analysis, auditing, or security monitoring.
Workflow	A sequence of automated steps used in SOAR systems to respond to alerts or manage security processes.
Telemetry	Security-relevant data collected from endpoints, networks, or systems for analysis and detection.

Threat Detection	The process of identifying potential security risks or suspicious activity within an IT environment.
ISO Image	A file containing the full contents of a disk, commonly used to install operating systems on virtual or physical machines.
OVA	Open Virtual Appliance – A packaged file that contains a pre-configured virtual machine image, used for quick deployment in virtualization software.

Table 1: Terms and Abbreviations

4 INTRODUCTION

In late 2024, a sophisticated cyberespionage campaign attributed to the Chinese state-sponsored group "Salt Typhoon" infiltrated multiple major U.S. telecommunications providers, including AT&T, Verizon, and T-Mobile. Exploiting vulnerabilities in network infrastructure, the attackers gained unauthorized access to sensitive metadata and communications, remaining undetected for over a year (Kapco, 2025). This high-profile breach highlights the consequences of delayed threat detection and the need for unified, intelligent security systems.

Recent studies underscore how widespread this issue is. According to IBM Security's 2024 data breach report, it takes organizations an average of **279 days to identify and contain a breach**, with most of the damage—both financial and reputational—occurring within the first 12 months. For smaller businesses, the average financial loss exceeds **$2.5 million**, while those relying on third-party services pay even more (Davies, 2019). The report also reveals that **security automation technologies can cut breach costs in half**, highlighting the immense value of integrated, automated solutions.

Despite these risks, many organizations continue to rely on disconnected tools like SIEM, IDS, XDR, and SOAR which lack proper integration. This fragmented approach reduces visibility, delays response time, and overwhelms analysts with disjointed alerts (Ali et al., 2023). As threat actors grow more advanced, cybersecurity operations must shift from manual and reactive to proactive, automated, and unified.

Problem Statement

Despite the increasing sophistication of cyber threats, many organizations continue to rely on disparate security tools that lack integration. This fragmentation hampers the ability to detect and respond to attacks instantly, placing a significant manual burden on security teams. There is an urgent need for an affordable, automated solution that unifies open-source tools like SIEM, XDR, SOAR, and IDS into a cohesive system. This project aims to design and implement such a framework using tools like Wazuh, Suricata and Shuffle to detect threats in real time, automate responses.

Objectives

- Develop a unified system combining SIEM, XDR, SOAR, and IDS functionalities using tools such as Wazuh, Suricata, and Shuffle.
- Detect and respond to real-world security threats, including brute-force attacks, malware activity, data exfiltration and ransomware attacks.

- Automate key security actions such as IP blocking, file quarantine, and alert notifications without human intervention.
- Provide an affordable, scalable solution specifically designed for small-to-medium enterprises and organizations with limited cybersecurity resources.

This project contributes to the field of cybersecurity by demonstrating how open-source tools can be integrated to build an effective and automated threat detection and response system. It highlights that organizations—especially those with limited budgets—can strengthen their security posture without relying on expensive commercial solutions. By automating responses, the system also reduces the manual workload typically required from analysts. The study addresses two key questions: How can integrating open-source SIEM, IDS, SOAR, and XDR tools improve cybersecurity efficiency? And to what extent can automation reduce the need for human involvement in routine security tasks?

The project is limited to on-premises, Linux-based environments. Constraints include resource limitations, time-bound testing, and the exclusive use of open-source tools.

5 LITERATURE REVIEW

5.1 OVERVIEW OF EXISTING RESEARCH

Cybersecurity threats are evolving rapidly, often outpacing the capabilities of traditional, fragmented defence systems. In response, modern strategies increasingly emphasize integrating multiple security layers - SIEM, SOAR, IDS, and XDR - into a unified framework. This convergence not only improves visibility across infrastructure but also enables faster and more consistent threat response. Despite this shift, many organizations, particularly small-to-medium enterprises (SMEs), continue to rely on disjointed tools, resulting in detection delays, increased manual workload, and analyst fatigue (Podzins and Romanovs, 2019; Amami et al., 2024).

Open-source platforms such as Wazuh, Suricata, and Shuffle have emerged as cost-effective alternatives to commercial solutions. These tools provide centralized logging, real-time monitoring, and automated incident response, offering enterprise-grade capabilities while remaining flexible and customizable for internal use (Bassey et al., 2024; Nguyen et al., 2024).

Table 2: Core Roles and Strengths of Selected Open-Source Tools

Tool	Primary Role	Strengths
Wazuh	SIEM & Host-based Monitoring	Log collection, file integrity monitoring, vulnerability detection, built-in XDR functions
Suricata	IDS/IPS & Network Threat Detection	Deep packet inspection, anomaly detection, high-performance event processing
Shuffle	SOAR & Security Automation	Playbooks, API integrations, low-code/no-code automation workflows

Table 2 outlines the key roles and strengths of popular open-source security tools.

Research by Bridges et al. (2023) and Roche and Dowling (2023) highlights that combining these tools into an integrated environment significantly reduces response times, lowers manual effort, and improves visibility across digital assets.

Table 3: Comparative Analysis of Open-Source SIEM and SOAR Tools

Tool	Function	Advantages	Limitations
Wazuh	SIEM & XDR-like	Scalable, cloud-ready, compliance support, strong community	Requires expertise for advanced tuning
Elastic SIEM	SIEM with analytics	High-quality visualizations, handles large data volumes	High resource use, complex to deploy
OSSIM	SIEM + IDS	Lightweight, bundled features	Limited scalability, outdated engine
Shuffle	SOAR & Automation	Easy setup, low-code playbooks, Wazuh integration	Small user base, lacks advanced logic
WALKOFF	SOAR framework	Highly extensible and flexible	Steep learning curve, higher maintenance

Table 3 provides a detailed comparison of leading open-source SIEM and SOAR tools, highlighting trade-offs between performance, ease of use, and extensibility.

XDR is gaining momentum for bridging the gaps left by traditional EDR and SIEM systems. It correlates data from endpoints, networks, and cloud environments to provide centralized threat visibility (Brandao and Nunes, 2021; George et al., 2021).

5.2 Real-World Implementation of SIEM, SOAR, and IDS Integration

Several studies have tested the integration of these tools in both lab and real-world environments. Bassey et al. (2024) implemented a unified setup using Wazuh for log correlation, Suricata for network intrusion detection, and TheHive for alert management—achieving real-time detection without reliance on commercial or cloud-based tools.

Nguyen et al. (2024) demonstrated Shuffle's ability to automate workflows and integrate with third-party platforms. Roche and Dowling (2023) deployed Splunk SIEM and later added Splunk SOAR to build a responsive and flexible security framework. While this integration required additional configuration, it ultimately enhanced operational efficiency.

Sridharan and Kanchana (2022) simulated attacks in a home lab to assess SIEM-SOAR synergy. Their results showed faster detection, reduced manual workload, and improved security posture, reinforcing the value of modular open-source tools for enterprise-grade defense. Roche and Dowling (2023) also reported that hybrid deployments combining SIEM with SOAR significantly reduced mean time to detect (MTTD) and respond (MTTR), further validating open-source integration strategies.

Integration of SIEM, SOAR, and XDR

Building on these findings, integrating SIEM, SOAR, and XDR creates a robust, layered defense by combining real-time visibility, automation, and cross-domain analytics. According to Sridharan and Kanchana (2022), each tool brings distinct capabilities, and their integration enhances operational responsiveness and efficiency.

Key Benefits Include:

- **Centralized Threat Visibility:** SIEM tools like Wazuh collect and correlate logs from diverse sources, enabling detection of suspicious behavior in real time.
- **Automated Response Workflows:** SOAR platforms such as Shuffle execute playbooks to isolate systems, block IPs, and escalate incidents—minimizing manual effort (Sridharan and Kanchana, 2022).
- **Cross-Domain Detection Accuracy:** XDR improves SIEM by correlating signals across endpoints, networks, and cloud environments, aiding detection of complex, multi-stage attacks (Kasturi et al., 2024).
- **Scalability and Flexibility:** The model adapts to hybrid infrastructures and is well-suited for resource-constrained organizations.
- **Proactive Risk Management:** Context-aware automation reduces false positives and enables faster, more targeted threat mitigation (Kasturi et al., 2024).

Kasturi et al. (2024) further note that unified systems consolidate data across multiple layers, enhancing threat correlation and detection accuracy for complex attack patterns.

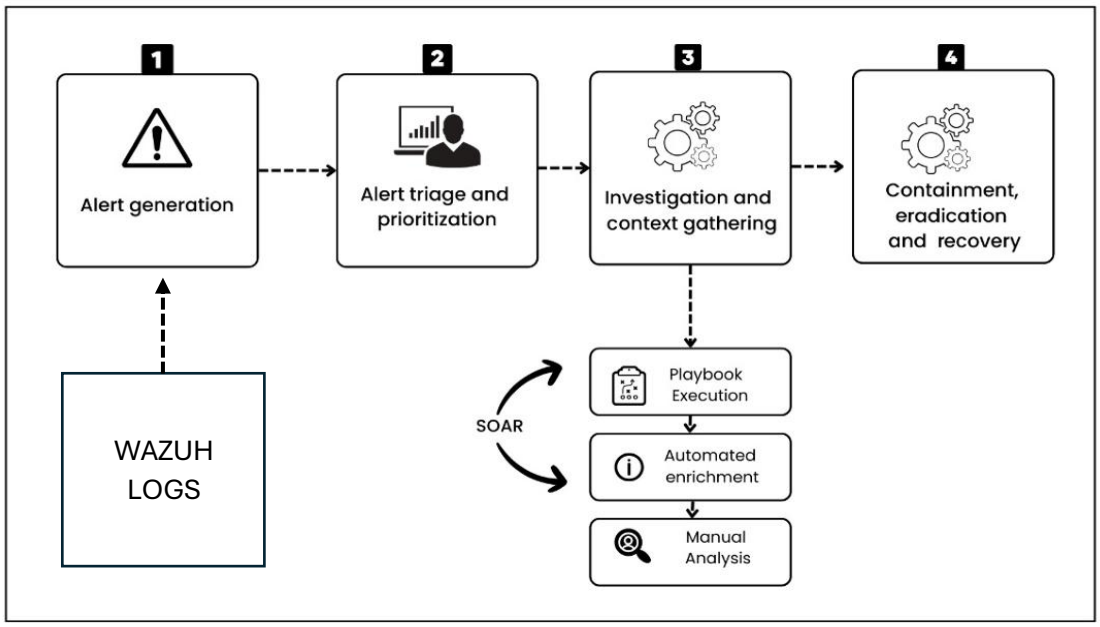


Figure 1: SOAR-enhanced incident response cycle (Gupta, 2024)

5.3 CRITICAL ANALYSIS AND THEORETICAL FRAMEWORK

Theoretical Framework:

- The MITRE ATT&CK framework is used to structure detection logic and align response actions (Nguyen et al., 2024; Kasturi et al., 2024).

The literature strongly supports the integrated use of SIEM, SOAR, and XDR to enhance detection and response. SIEM tools like Wazuh offer powerful log aggregation, threat correlation, and compliance features (Ali et al., 2024). However, they are resource-intensive and require skilled staff for continuous tuning (Podzins and Romanovs, 2024).

SOAR platforms like Shuffle automate repetitive tasks and enforce consistent workflows, reducing manual overhead (Nguyen et al., 2024). Still, they rely heavily on predefined playbooks, which must be regularly updated to remain effective. Over-automation can also limit flexibility in novel attack scenarios.

XDR adds context and improves accuracy by correlating signals across multiple layers (Brandao and Nunes, 2021). Yet, it remains relatively new and can be complex to implement in open-source settings, where integration standards are still developing.

While open-source tools offer low-cost, customizable solutions, they often require greater effort for integration. Scalability and alert fatigue are persistent concerns. Overall, the research underscores the advantages of integration but also identifies real-world deployment challenges.

5.4 JUSTIFICATION FOR YOUR WORK

Despite growing support for integrated cybersecurity models, several key gaps remain in the existing literature:

- Limited exploration of XDR in open-source settings aligned with SME requirements (Brandao and Nunes, 2021).
- Lack of real-world examples detailing SOAR playbook design and automation performance—especially using Shuffle (Nguyen et al., 2024).
- Absence of threat-aware, custom rules for attacks like brute-force login or data exfiltration (Kasturi et al., 2024).
- Minimal demonstrations of closed-loop alert-response actions such as automated blocking or ransomware containment (Roche and Dowling, 2023).

While many papers outline the individual benefits of SIEM, SOAR, IDS, and XDR, few provide a complete, low-cost implementation using open-source tools. Most studies focus on commercial setups or remain theoretical.

This project addresses those gaps by integrating Wazuh, Suricata, and Shuffle into a cohesive, on-premises framework tailored for SMEs with limited budgets. It builds on established principles of detection, correlation, and automation while directly tackling practical issues like integration complexity and scalability. The solution is designed to be replicable, real-time, and independent of cloud services.

6 METHODOLOGY

6.1 RESEARCH DESIGN

This project uses an experimental design with controlled simulations in a virtual lab to evaluate how integrated open-source tools detect, analyse, and respond to threats in real time. Instead of user data or surveys, it focuses on system performance, automation, and usability through repeatable scenarios.

6.2 DATA COLLECTION

Data is collected through **simulated attacks and system activities** on monitored endpoints. These include logins, file modifications, network scans, and command executions. Security events are generated within the lab environment using virtual machines configured to represent attacker, target, and monitoring systems. Wazuh collects logs and telemetry from endpoints, Suricata analyzes network traffic. All data is captured and reviewed via the Wazuh dashboard and Shuffle automation logs.

6.3 TOOLS AND TECHNOLOGIES

1. Wazuh (SIEM & Host Monitoring) - Used for log collection, threat detection, and alert management.
2. Suricata (Network IDS/IPS) - Analyzes network traffic and triggers alerts for suspicious activity.
3. Shuffle (SOAR Platform) - Automates incident response using predefined playbooks; deployed via Docker.
4. VirusTotal (Threat Intelligence API) - Verifies file hashes received from Wazuh to detect malware.
5. Python (Scripting & Integration) - Supports automation and integration between security components.
6. Auditd (File Integrity Monitoring) - Monitors changes to important files and logs system activity.
7. iptables (Firewall) - Automatically blocks IPs when triggered by critical alerts.
8. Hydra (Brute force Tool) - Performs brute-force SSH login attempts to test detection response.
9. Nmap (Network Scanner) - Simulates port scans.

10. Netcat, SCP, cURL, Wget (Exfiltration Simulation Tools) - Used to simulate unauthorized data transfers.

11. Outlook (Email Notification Platform) - Receives automated alert notifications sent by Shuffle.

12. Ubuntu Server (SOAR Host OS) - Runs Shuffle via Docker; chosen for stability and flexibility.

13. Ubuntu Desktop (Endpoint OS) - Monitored machine running Suricata, Auditd, and Wazuh agent.

14. Kali Linux (Attack Simulator) - Simulates real-world attack scenarios in a controlled environment.

15. VMware Workstation Pro (Virtual Lab) - Hosts all virtual machines within a secure, isolated network.

6.4 SYSTEM/NETWORK ARCHITECTURE

The system operates within a VMware Workstation Pro-based virtual lab, where all machines are connected through a shared NAT network using a virtual Ethernet hub. This setup enables secure, internet-enabled communication for simulating attacks, logging, and automated incident response workflows.

All virtual machines were provisioned locally. The Wazuh manager and Kali Linux attacker machine were deployed using official OVA files, allowing rapid setup with preconfigured settings. The Shuffle server and Ubuntu endpoint were manually installed using the latest Ubuntu ISO from Canonical.

System	Role	Configuration	Purpose
Wazuh Server	Central SIEM/XDR platform; collects logs, applies rules, generates alerts, and forwards them to Shuffle.	3 vCPUs, 8 GB RAM, 50 GB storage	Core detection and alerting engine of the system.
Shuffle Server (Ubuntu)	Runs the SOAR platform using Docker and handles automated response workflows.	3 vCPUs, 8 GB RAM, 50 GB storage	Manages automation, IP blocking, and alert notifications.
Ubuntu Endpoint	Simulates an employee workstation; runs Wazuh Agent, Suricata, Auditd, and iptables for testing detection.	2 vCPUs, 4 GB RAM, 50 GB storage	Acts as the monitored system where activities are logged and analyzed.
Kali Linux	Simulates attacker behavior using tools like Hydra, Nmap, Netcat, SCP, curl, and wget.	1 vCPU, 3 GB RAM, 30 GB storage	Attack simulation VM used to validate detection and response mechanisms.

Table 4: Configurations given the the VM

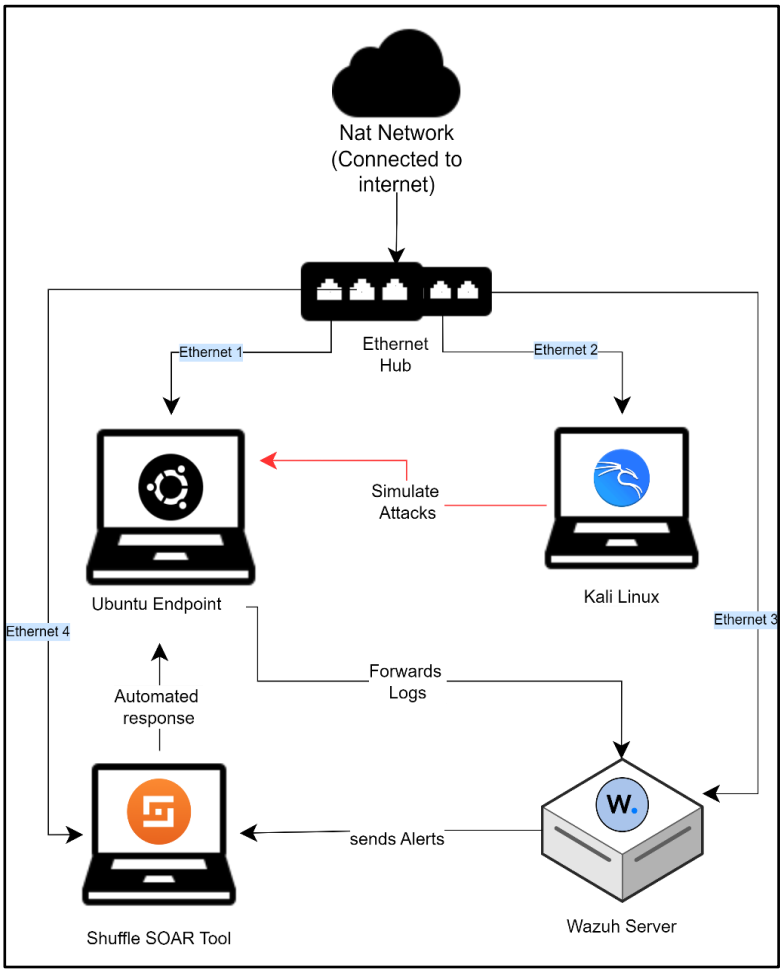


Figure 2: Network Architecture

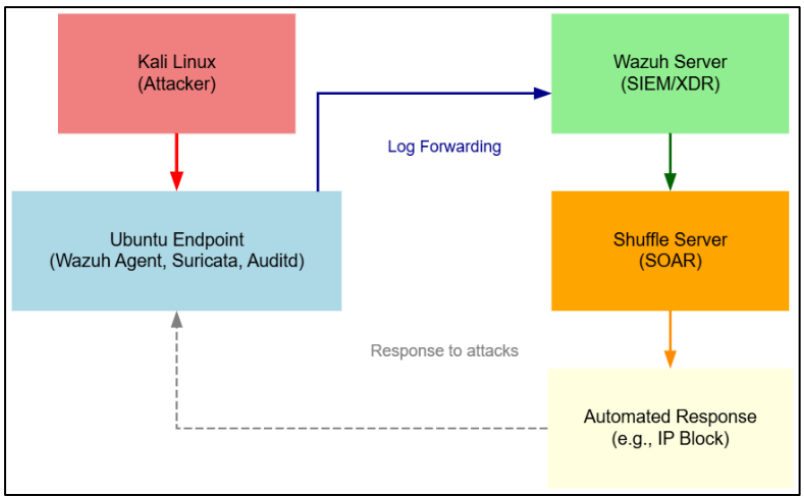


Figure 3: Network Flow

6.5 Implementation Procedures

1. **Setup of Virtual Lab Environment:** Created four VMs (Wazuh Server, Shuffle Server, Ubuntu Endpoint, Kali Linux) connected via NAT for safe and isolated testing.
2. **Installation and Configuration of Wazuh Server and Agent:** Installed Wazuh on the server and agent on the endpoint, enabling log collection and rule-based alerting.
3. **Deployment of Suricata on the Endpoint:** Installed Suricata on the Ubuntu endpoint to monitor network traffic and forward alerts to Wazuh.
4. **Installation of Shuffle (SOAR) via Docker:** Deployed Shuffle using Docker on Ubuntu Server to automate responses to Wazuh alerts.
5. **Integration of Wazuh with Shuffle:** Linked Wazuh to Shuffle by creating webhooks during each use case to trigger automated workflows.
6. **Integration of VirusTotal API:** Configured Wazuh to check file hashes from endpoint with VirusTotal and respond if malicious with Shuffle.
7. **Simulation of Attacks from Kali Linux:** Performed brute-force, scanning, and exfiltration attacks to test detection and response mechanisms.

6.6 Security and Ethical Considerations

This project was conducted entirely within a secure, isolated virtual lab using **VMware Workstation Pro**, ensuring no impact on external networks or systems. All attack simulations and testing activities were performed on internally controlled machines with no connection to live production environments. No real user data or sensitive personal information was used at any stage. Only synthetic logs and simulated behaviours were generated for testing detection, automation, and alert enrichment features.

6.7 Validation and Testing

System validation was performed through simulated attacks, verifying accurate alert generation in Wazuh, correct execution of Shuffle responses, and proper display summaries for enriched event context will also be discussed in later sections.

---

7 IMPLEMENTATION AND DEVELOPMENT

7.1 DEVELOPMENT PROCESS

The project began with basic planning to define goals, select appropriate tools, and outline key requirements such as low cost, modularity, and offline deployment. The focus was on using open-source solutions and building a system suitable for small-to-medium environments.

An **iterative development approach** was followed, where each use case was implemented in its own cycle. The process involved:

- Identifying and planning one use case at a time
- Reading official documentation and setting up detection rules
- Simulating activity from Kali Linux to trigger alerts
- Verifying detection in Wazuh and refining configurations
- Integrating Shuffle only after detection was successful
- Troubleshooting via community forums when needed
- Moving to the next use case after completing one end-to-end

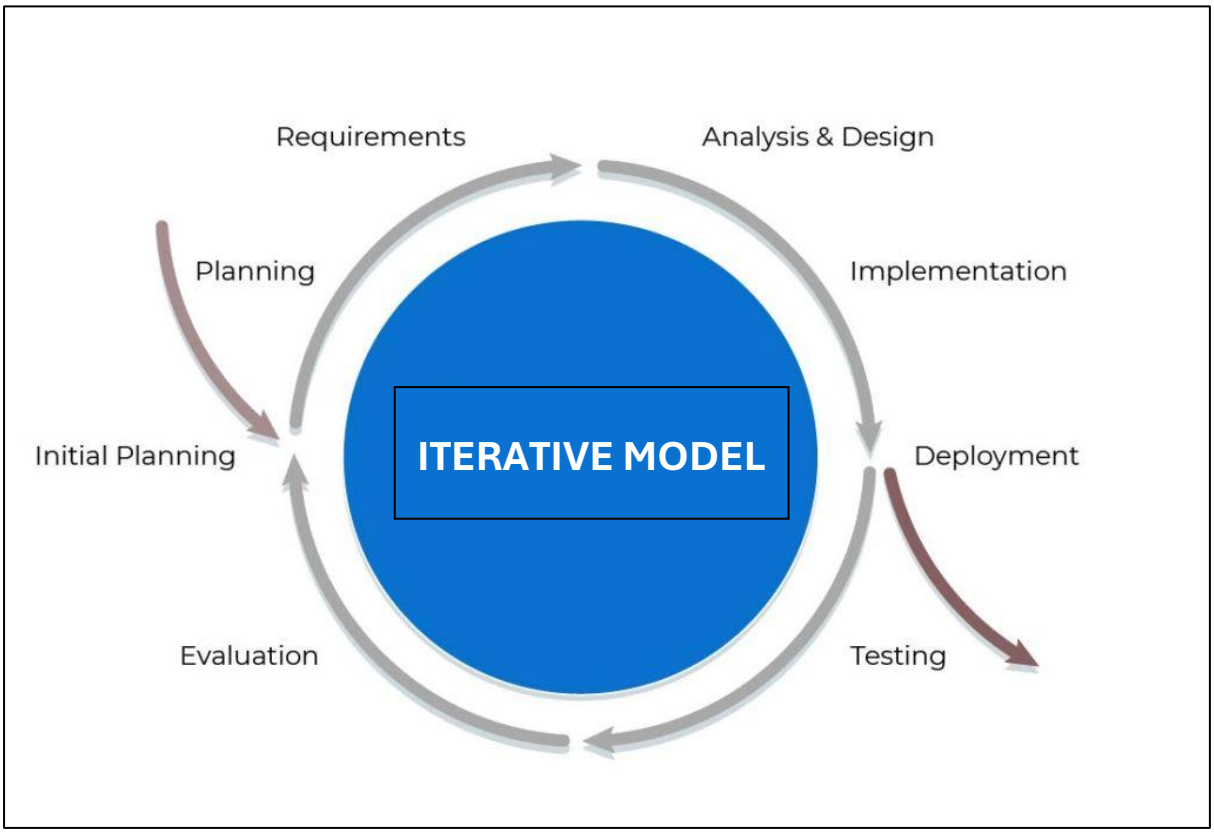


Figure 4: SDLC Model

7.2 PROJECT MILESTONES

Week	Milestone
Week 1–2	Finalized project topic after initial idea exploration
Week 3–4	Researched tools and compared Splunk, QRadar, Wazuh
Week 5–6	Confirmed Wazuh setup; designed VM-based lab architecture
Week 7–8	Documented literature review, problem statement, and prototype
Week 9	Drafted test cases, use cases, and created network diagram
Week 10–11	**Semester Break** – Continued writing first deliverable content
Week 12–13	Finalized and polished first deliverable sections
Week 14	**First Deliverable Submitted – 12th January 2025**
Week 15	Discussed feedback; updated use cases and test cases
Week 16–17	Implemented and tested SSH brute-force detection
Week 18	**Demoed four use cases to supervisor – End of February**
Week 19–22	Developed logon monitoring, privilege abuse, and FIM use cases
Week 23–25	Refined SOAR workflows; added automation and alert response
Week 26	Integrated VirusTotal into Wazuh
Week 27	**Started documentation – 8th April 2025**
Week 28–29	Completed final testing and verified system responses
Week 31	**Final submission deadline – 3rd May 2025**
Week 32	**Viva and Evaluation (TBA)**

Table 5: Project milestones

The project used an iterative approach, with each phase building on the last. Weeks 3–6 focused on researching tools, setting up Wazuh, and designing the VM lab—crucial for defining scope and goals. Regular updates, including use case refinements and testing, ensured steady, focused progress.

ID	Task Name	Start	End	Duration	Color
1	Overall Project Duration	2024-10-12	2025-05-16	217 days	
2	Topic Finalized; Tool Evaluation	2024-10-19	2024-11-08	21 days	
3	Wazuh Confirmed; Lab Setup	2024-11-09	2024-11-22	14 days	
4	Drafted Docs: Review, Prototype, Diagram	2024-11-23	2024-12-13	21 days	
5	Deliverable Work – Semester Break	2024-12-14	2024-12-27	14 days	
6	Finalized First Deliverable Content	2024-12-28	2025-01-10	14 days	
7	First Deliverable Submitted	2025-01-12	2025-01-12	1 day	
8	Planning + Use Case Demo	2025-01-18	2025-02-14	28 days	
9	Developed & Integrated Use Cases	2025-02-15	2025-03-14	28 days	
10	Automation + API Integration	2025-03-15	2025-04-11	28 days	
11	Documentation + Use Case Recording	2025-04-12	2025-05-02	21 days	
12	Final Testing + Report Formatting	2025-05-03	2025-05-09	7 days	
13	Final Submission	2025-05-03	2025-05-03	1 day	
14	Viva + Evaluation	2025-05-17	2025-05-23	7 days	

Figure 5: Project Gantt Chart

7.3 USE CASES

The table below outlines the individual use cases implemented during the project, each targeting a specific aspect of detection, automation, or threat response within the integrated Wazuh–Suricata–Shuffle environment. *Refer to the use case diagram in Appendix figure 70.*

Use Case ID	Use Case Name	Objective
UC01	Detection of Nmap Port Scans	Detect and alert on Nmap scanning activity using custom Suricata rules parsed by Wazuh.
UC02	SSH Brute-Force Detection with Active Response	Detect brute-force SSH login attempts and automatically block attacker IPs using Wazuh and Shuffle.
UC03	Malware Detection and Automated Response with VirusTotal	Detect and automatically delete malicious files confirmed by VirusTotal, while isolating the infected endpoint.
UC04	Logon Monitoring with Time-Based Alerting	Detect and alert on employee logins outside defined working hours or during weekends.
UC05	File Integrity Monitoring and Ransomware Detection	Detect unauthorized file additions, deletions, or modifications, and identify ransomware-like activity through correlation rules.
UC06	Data Exfiltration Detection Using Suricata	Detect unauthorized downloads of sensitive files simulating basic data exfiltration attempts.
UC07	Suspicious Command Monitoring and Data Exfiltration Detection	Monitor and detect suspicious or exfiltration-related command executions using Auditd and Wazuh correlation rules.

Table 6: Use case table

7.4 CONFIGURATION AND IMPLEMENTATION

7.4.1 Virtual Lab Setup

A virtualized lab environment was created using **VMware Workstation Pro** to simulate a secure, isolated network for testing. The lab consisted of four virtual machines:

- **Wazuh Server** (SIEM & alert manager)
- **Ubuntu Endpoint** (monitored system)
- **Kali Linux** (attacker simulation)
- **Shuffle Server** (SOAR platform via Docker)

All machines were connected through a **NAT-based virtual network**, enabling controlled traffic flow between systems while maintaining external internet access for API integrations. The setup allowed full-cycle simulation, detection, enrichment, and response without affecting the host system or live environments.

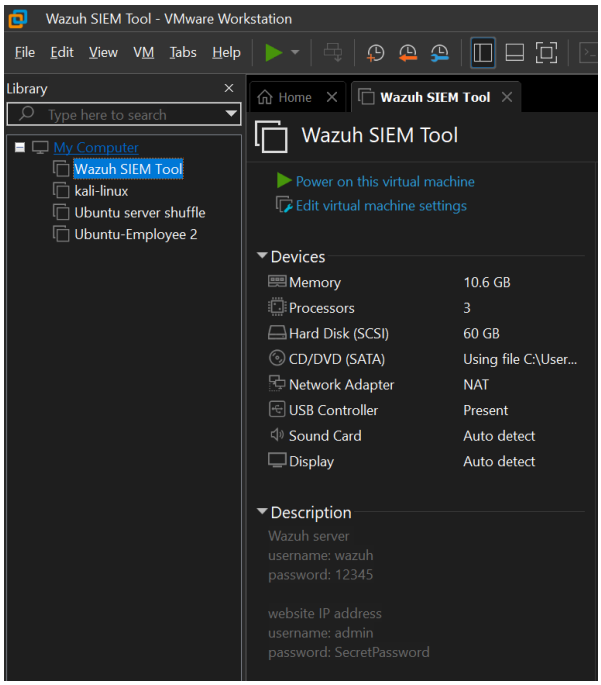


Figure 6: VM configuration

7.4.2 Wazuh Manager Installation and Endpoint Configuration

To enable centralized security monitoring, the Wazuh Manager was deployed using the official ova image provided on the Wazuh website. Importing this file into VMware Workstation Pro automatically provisioned the necessary resources such as CPU, memory, and disk allocation, allowing for a quick and consistent setup.

After Starting the virtual machine, the Wazuh server's internal IP address was identified using:

```
ip a
```

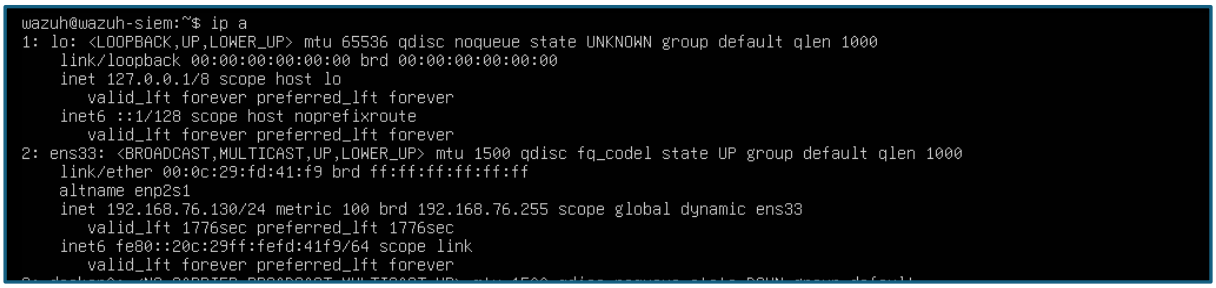


Figure 7: Wazuh server terminal showing IP

The **Wazuh web dashboard** was accessed at: `https://192.168.76.130` using the credentials (Figure 8):

- **Username**: admin
- **Password**: SecretPassword

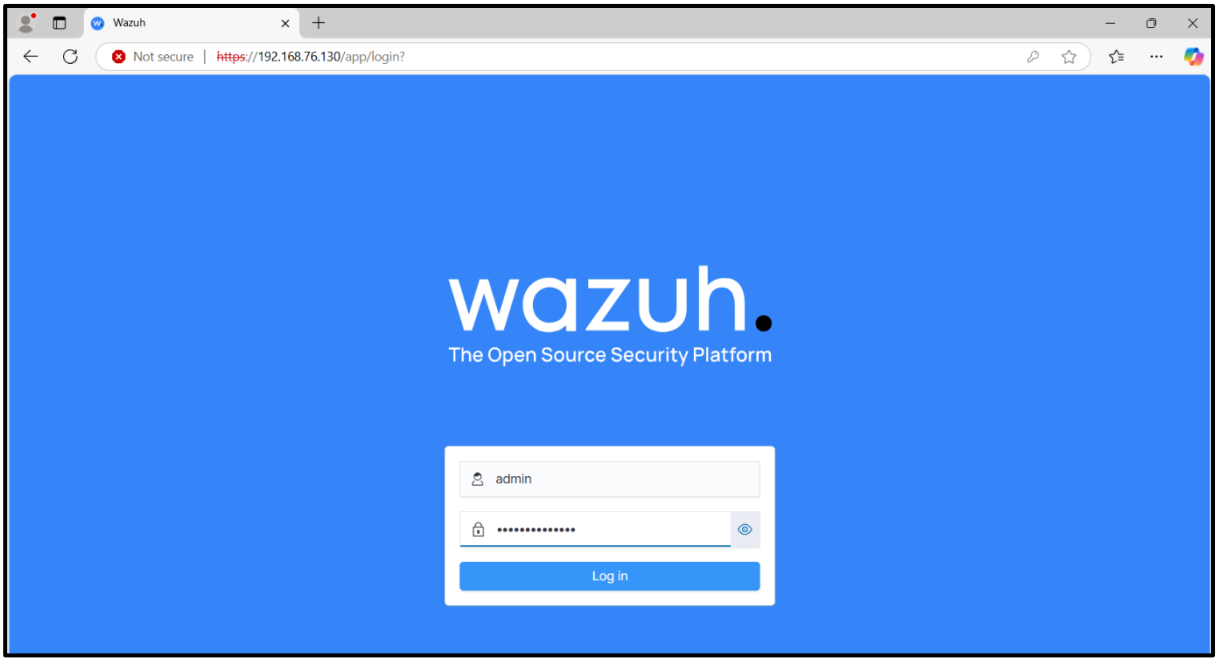


Figure 8: Wazuh dashboard login-screen

Once logged in, the **agent deployment interface** was accessed via:

`https://192.168.76.130/app/endpoints-summary#/agents-preview/deploy`

In the agent deployment section:

- Selected architecture: deb amd64
- Assing a server address Manager IP: 192.168.76.130 (Wazuh Server IP)
- Agent name: Employee-2
- The Auto-generated command was copied (Figure 10)

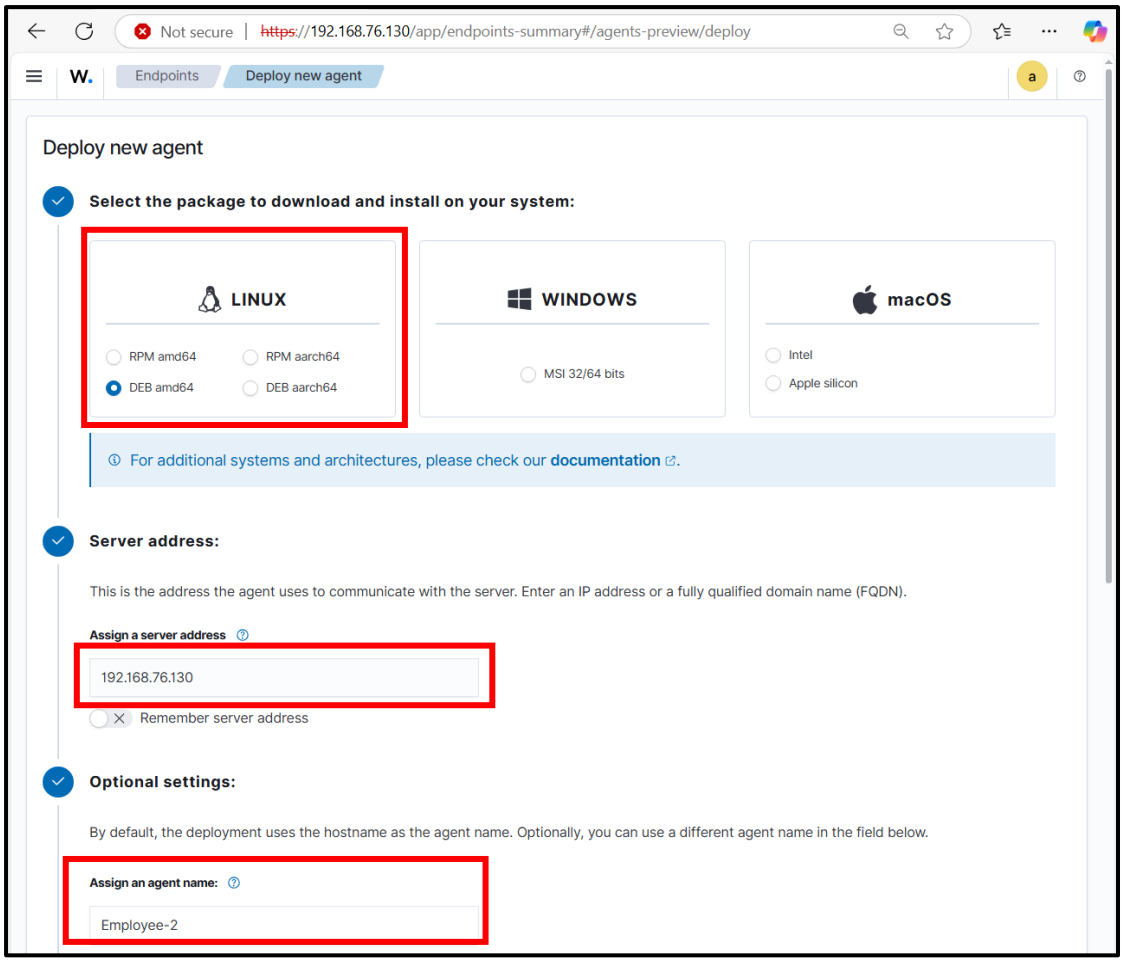


Figure 9: Agent deployment screen with settings filled

Figure 10: Commands on Ubuntu endpoint

Wazuh dashboard showing active agent (Employee-2) Finally, the Wazuh dashboard showed the **Employee-2 agent** in an **active state**, confirming successful installation and connection (Figure 11).

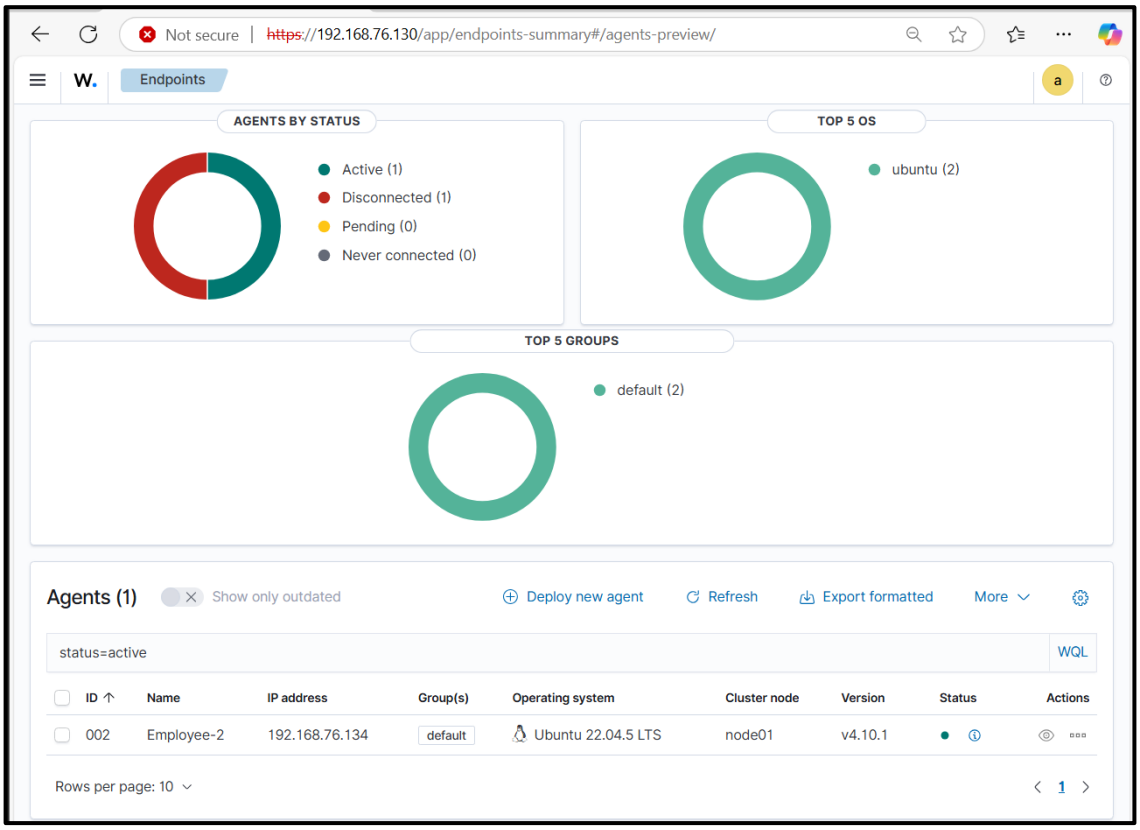


Figure 11: Wazuh dashboard showing active agent (Employee-2)

Purpose:
These steps ensure that the Ubuntu endpoint is actively monitored by the Wazuh manager, enabling real-time log collection, rule-based threat detection, and integration with the broader security automation workflow. (NOTE: Replace the IP address sections with the Wazuh Manager's IP address.)

7.4.3 Suricata Deployment and Integration with Wazuh

Suricata was installed on the Ubuntu endpoint to provide deep network visibility and generate alerts for upcoming use cases. These alerts were then forwarded to the Wazuh manager for centralized correlation and analysis.

7.4.3.1.1 Installation and Ruleset Setup

Suricata was installed via the command (Figure 12):

```
$ sudo add-apt-repository ppa:oisf/suricata-stable
$ sudo apt-get install suricata -y
```

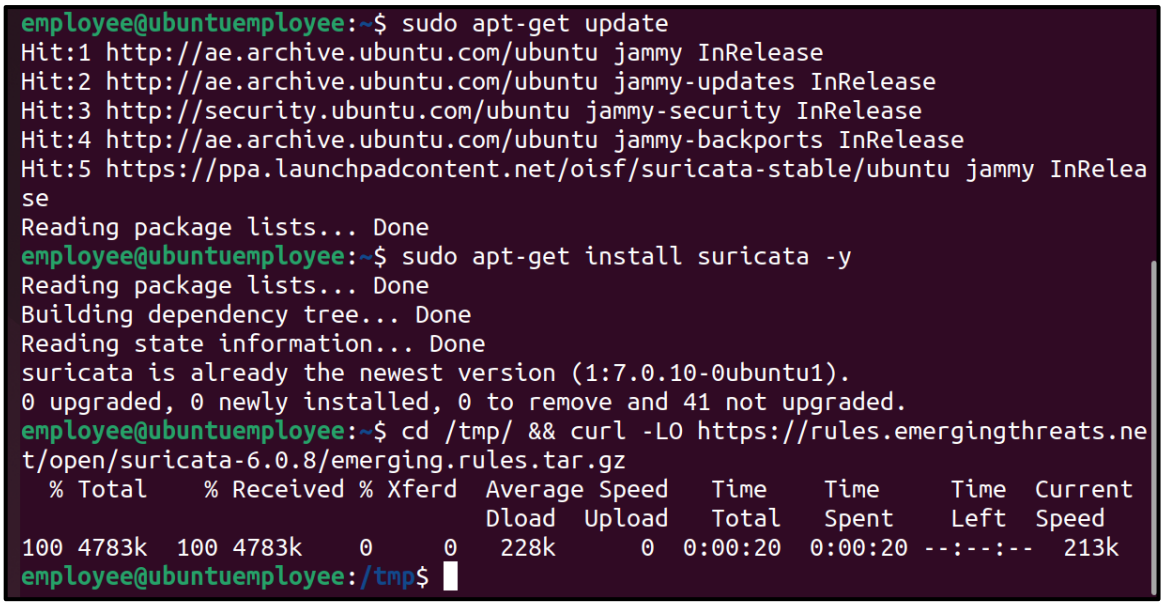


Figure 12: Suricata installation from terminal

The Emerging Threats ruleset was added to Suricata's custom rules to improve detection of scans and suspicious connections, making the IDS more aligned with project use cases.

```
$ cd /tmp/

$ curl -LO https://rules.emergingthreats.net/open/suricata-
6.0.8/emerging.rules.tar.gz

$ sudo tar -xvzf emerging.rules.tar.gz

$ sudo mkdir /etc/suricata/rules

$ sudo mv rules/*.rules /etc/suricata/rules/

$ sudo chmod 640 /etc/suricata/rules/*.rules.
```

7.4.3.2 Suricata Configuration Modifications

Suricata's main configuration file (`sudo nano /etc/suricata/suricata.yaml`) was edited to define detection boundaries and rule behavior:

- HOME_NET was set to the IP address of the monitored Ubuntu endpoint (192.168.76.134)
- EXTERNAL_NET set to "any"
- default-rule-path set to /etc/suricata/rules
- rule -files updated to include *.rules

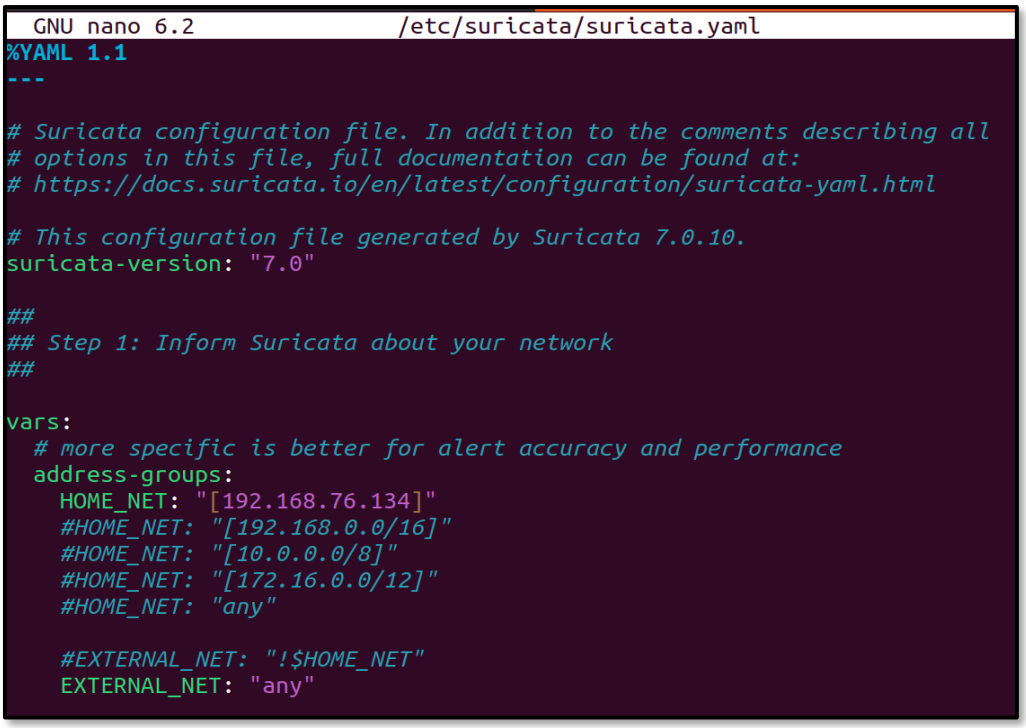


Figure 13: YAML configuration

The EXTERNAL_NET set to 'any' includes all external traffic for analysis. The default-rule-path and *.rules ensure Suricata loads all detection rules from the specified directory (Figure 14).

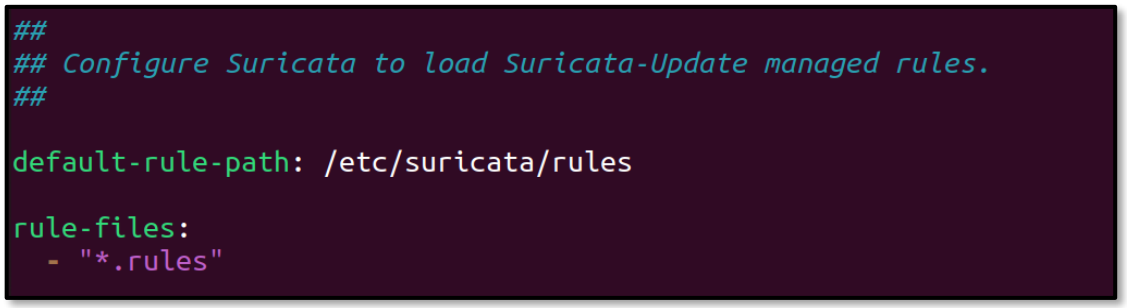


Figure 14: YAML configuration with detection boundaries and rules defined

Under the af-packet section, the active interface **ens33** was defined reason is the Ubuntu employee machine has the NAT interface connected to ens33 (figure 15).

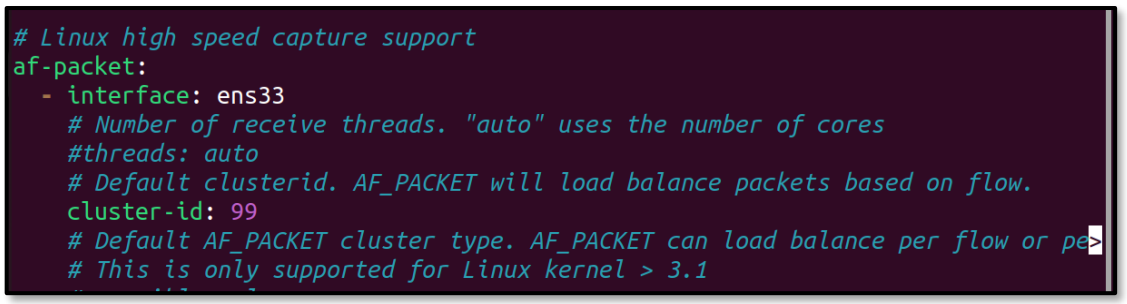


Figure 15: YAML configuration with detection boundaries and rules defined

Suricata was restarted to apply changes by below command.

```
sudo systemctl restart suricata
```

Purpose: These configurations ensure that Suricata is actively monitoring the correct network interface and applying the appropriate rule sets for live traffic inspection.

7.4.3.3 Wazuh Integration

To enable Wazuh to process Suricata alerts, the agent (Ubuntu employee machine) was configured to monitor Suricata's log file (eve.json) by editing the ossec.conf file (Figure 16):

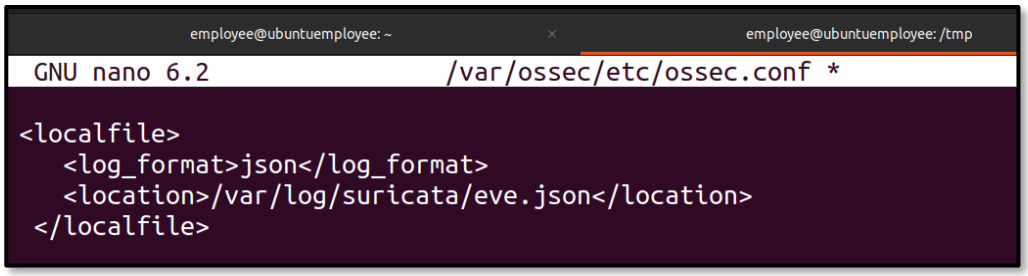


Figure 16: ossec.conf snippet monitoring Suricata logs

The Forwarding Suricata logs to Wazuh enables centralized alert management, real-time correlation with host-based data, and rule-driven responses through Wazuh. The following command was executed to restart the Wazuh agent and apply the configuration:

```
sudo systemctl restart wazuh-agent
```

7.4.3.4 Testing and Validation

To validate the setup, a **ping scan** was executed from another machine to Ubuntu employee machine for simulating an ICMP event. Suricata detected the activity and logged it in eve.json, which was successfully parsed and forwarded by the Wazuh agent (Figure 17).

```
ping -c3 192.168.76.134
```

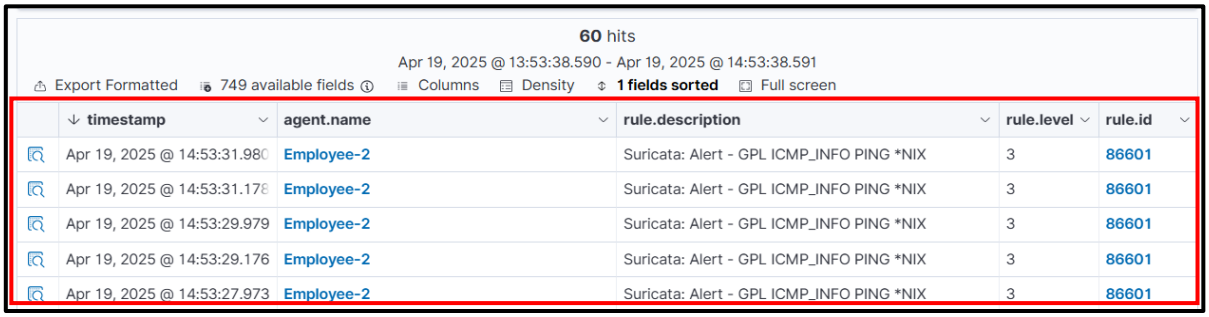


Figure 17: Wazuh dashboard showing Suricata-detected ICMP alert

Purpose:
This confirmed that the entire pipeline—from network threat detection to centralized alert visibility—was functioning as intended.

7.4.4 Configuring Auditd for File Integrity Monitoring (FIM)

To enable detailed file integrity monitoring on the Ubuntu endpoint, **Auditd** was installed and configured. Auditd serves as the underlying audit framework for Wazuh's **who-data monitoring**, allowing not only detection of file modifications, but also attribution — identifying the user and process responsible for each action.

7.4.4.1 Installation and Configuration Overview

Auditd was installed using Ubuntu's package manager. For systems running Audit version 3.1.1 or later, the `audispd-plugins` package was also required. After installation, the service was restarted to apply changes (Figure 18).

```
$ apt-get install auditd

$ apt-get install audispd-plugins

$ systemctl restart auditd
```

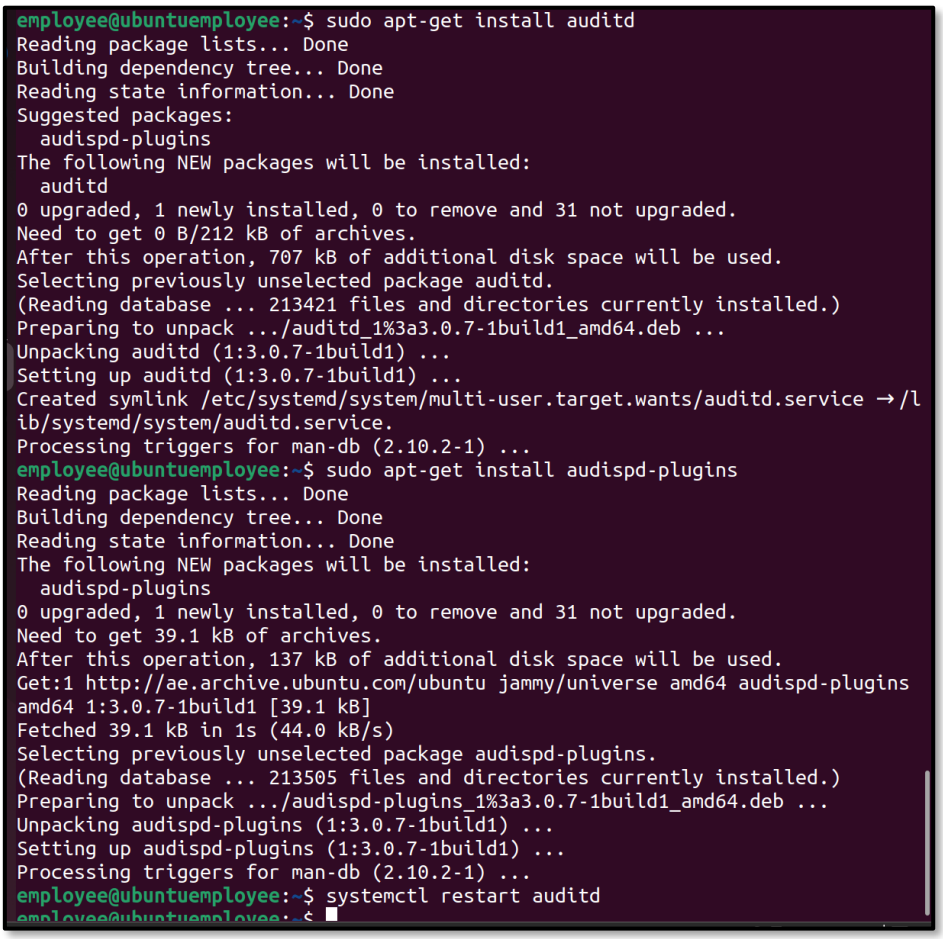


Figure 18: Auditd installed on Ubuntu endpoint

7.4.5 Shuffle SOAR Installation and Configuration

The open-source **Shuffle SOAR** platform was deployed to enable automated security response and workflow orchestration. It was installed on an Ubuntu 22.04 Server running in VMware using Docker and Docker Compose.

7.4.5.1 Preparing the Virtual Machine

An Ubuntu Server 22.04 image was downloaded from linuxvmimages.com and imported into VMware Workstation Pro to serve as the Shuffle host machine. System packages were updated to ensure a clean environment.

7.4.5.2 Installing Docker and Docker Compose

Docker was required to run the Shuffle services and dependencies. The following packages and keys were added to configure the Docker repository:

```
$ sudo apt install -y ca-certificates curl gnupg lsb-release

$ sudo mkdir -p /etc/apt/keyrings

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg

$ echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt update
```

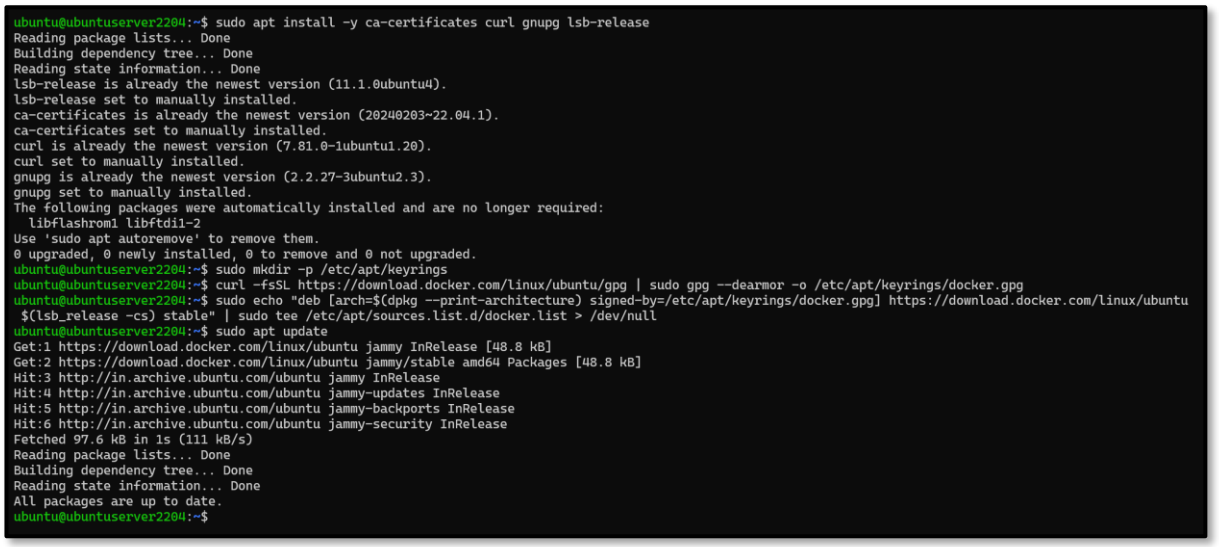


Figure 19: Docker repository configured

Docker and Docker Compose were then installed:

```
$ sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

$ sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose
```

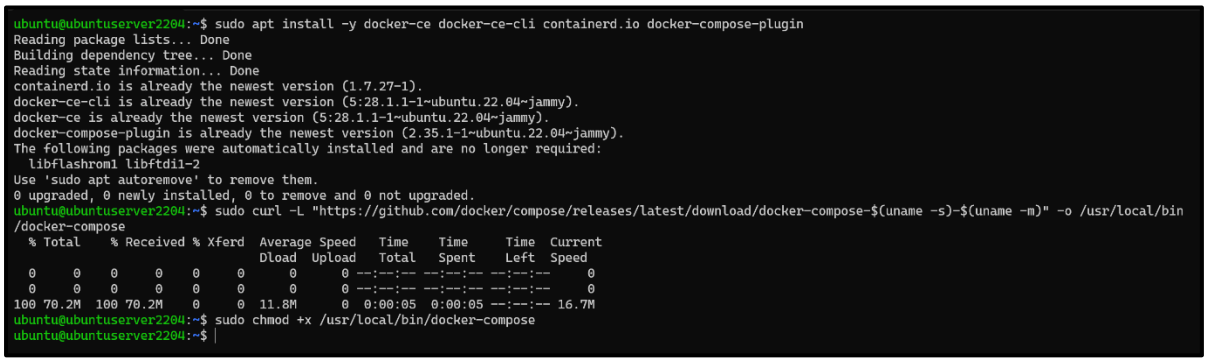


Figure 20: Docker and Docker Compose installation completed

7.4.5.3 Cloning and Launching Shuffle

The Shuffle SOAR repository was cloned from Github:

```
$ git clone https://github.com/Shuffle/Shuffle && cd Shuffle

$ docker-compose up -d

$ sudo sysctl -w vm.max_map_count=262144
```

7.4.5.4 Accessing the Shuffle Dashboard

The dashboard was accessed at: `https://192.168.76.144:3443/workflows` On first access, the user is prompted to register. The default admin account was set as:

- **Username**: admin
- **Password**: admin

Once logged in, the user is presented with the workflow interface where security playbooks can be designed using drag-and-drop nodes.

7.4.6 VirusTotal Integration with Wazuh

VirusTotal was integrated with Wazuh to enhance file-based threat detection. When combined with Wazuh's File Integrity Monitoring (FIM), this allows suspicious files to be automatically checked against VirusTotal's malware database using file hashes.

7.4.6.1 Getting the API Key

An account was created on the official VirusTotal website to access the public API (figure 21).

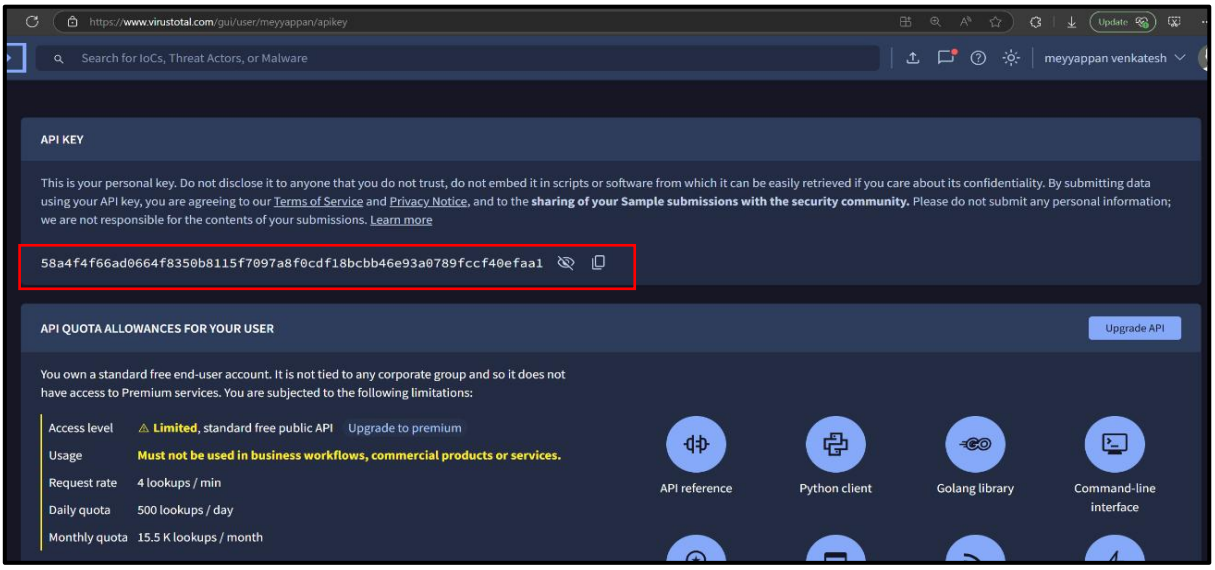


Figure 21: API key retrieved from account dashboard

Purpose: The API key allows Wazuh to query VirusTotal's database and check if a monitored file has been flagged as malicious.

7.4.6.2 Configuring VirusTotal Integration in Wazuh

The following integration Figure 22 block was added to the ossec.conf file of Wazuh server located at: `/var/ossec/etc/ossec.conf`

After saving changes, the Wazuh manager was restarted: `sudo systemctl restart wazuh-manager`

```
 GNU nano 5.8                    /var/ossec/etc/ossec.conf                    Modified

<integration>
  <name>virustotal</name>
  <api_key>58a4f4f66ad0664f8350b8115f7097a8f0cdf18bcbb46e93a0789fccf40efaa1</api_key>
  <!-- Replace with your VirusTotal API key -->
  <group>syscheck</group>
  <alert_format>json</alert_format>
</integration>
```

Figure 22: Configuration added to ossec.conf

7.4.7 Monitoring a Directory for File Changes

The Ubuntu agent was configured to monitor the Downloads folder—a common source of infections via email or browsers. Any new or modified file is tracked by Wazuh, and its hash is automatically sent to VirusTotal for malware analysis. In the agent's (Employee) /var/ossec/etc/ossec.conf this is added (Figure 23).

```
<directories check_all="yes" whodata="yes" report_changes="yes"
realtime="yes">/home/employee/Downloads</directories>
```

Figure 23: Directory monitoring rule for the Downloads folder

---

7.4.8 Retrieving the API Key from Shuffler.io

For automated Wazuh-to-Shuffle alerting, a user-specific API key is needed. Unlike local instances, the cloud-hosted Shuffler.io provides this key to securely trigger workflow email alerts based on Wazuh events. After logging in, the API key was retrieved by navigating to: https://shuffler.io/settings (Figure 24).

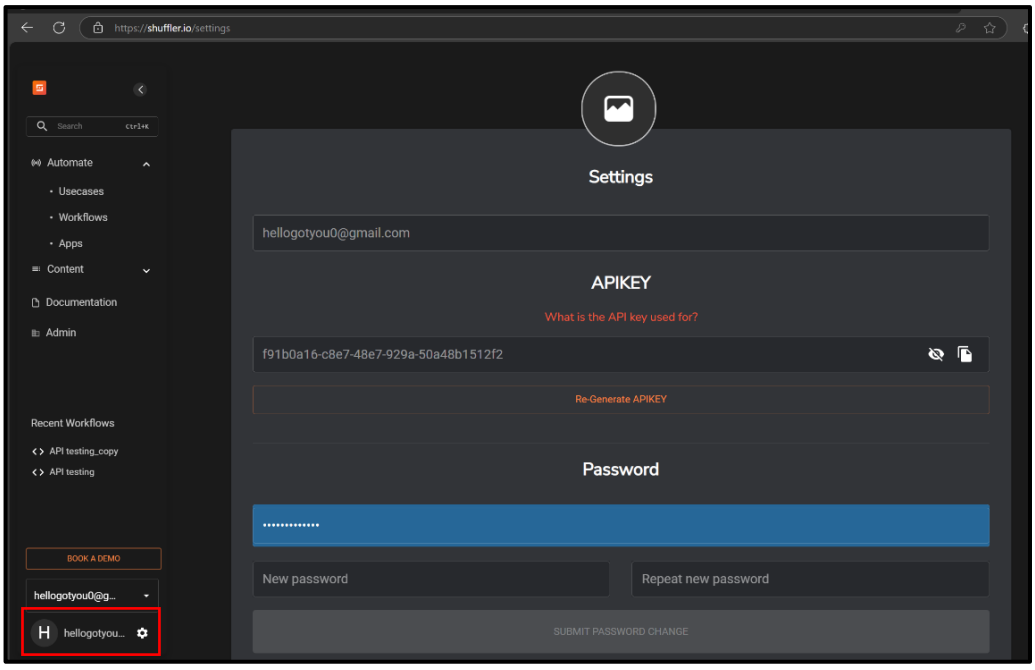


Figure 24: Shuffler.io dashboard showing location of API key

7.4.9 Wazuh integration with Shuffle

A new **Webhook trigger node** was added in the Shuffle workflow builder (Figure 25). Upon activation, it generated a unique Webhook URI:

```
https://192.168.76.133:3443/api/v1/hooks/webhook_fe1b816b-c53e-49df-8d06-1e157ea93976
```

This URL is responsible for receiving JSON-formatted alerts from Wazuh.

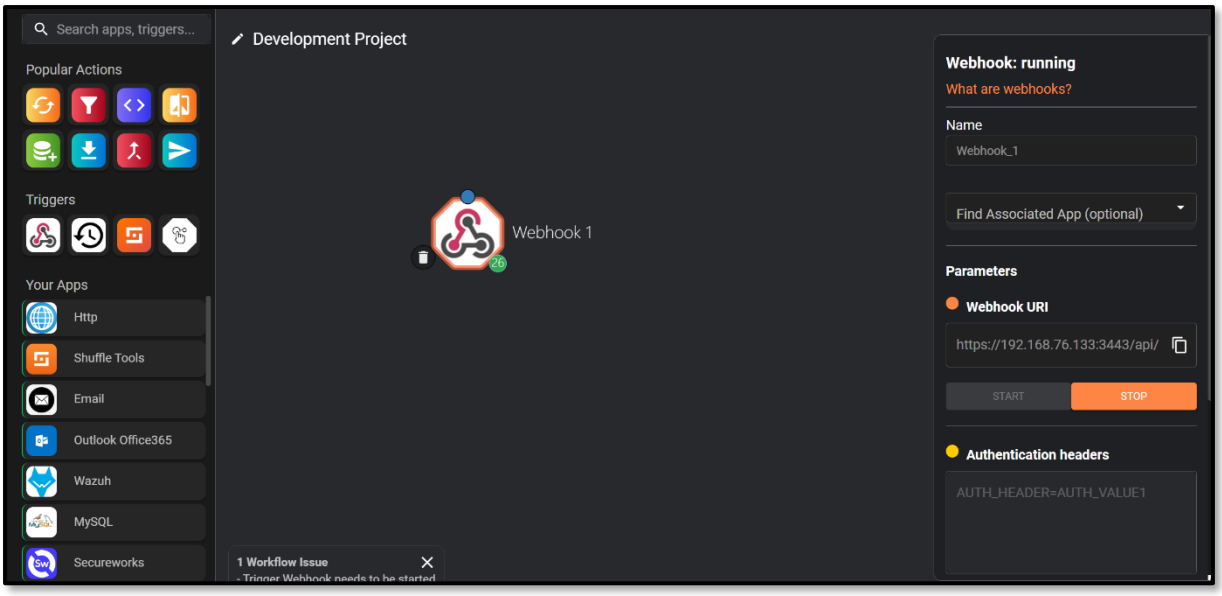


Figure 25: Webhook creation interface in Shuffle

Wazuh Integration with Centralized Webhook

As shown in **Figure 26**, the Wazuh manager's configuration file (/var/ossec/etc/ossec.conf) was updated with multiple <integration> blocks. Each block is linked to a specific use case by specifying its corresponding **rule ID** and **alert level**, while all forward alerts to the same Shuffle webhook endpoint.

The integrations cover the following use cases:

1. **SSH brute-force detection** (Rule ID 5763, Level 10)
2. **Malware detection via VirusTotal** (Rule ID 87105, Level 12)
3. **Logon monitoring** for off-hours or weekends (Rule IDs 17101, 17102, Level 9)

These webhook triggers are only activated **when the configured rule ID and level are matched** during runtime.

This ensures that Shuffle only receives alerts relevant to each specific automation workflow, keeping processing efficient and event handling precise.

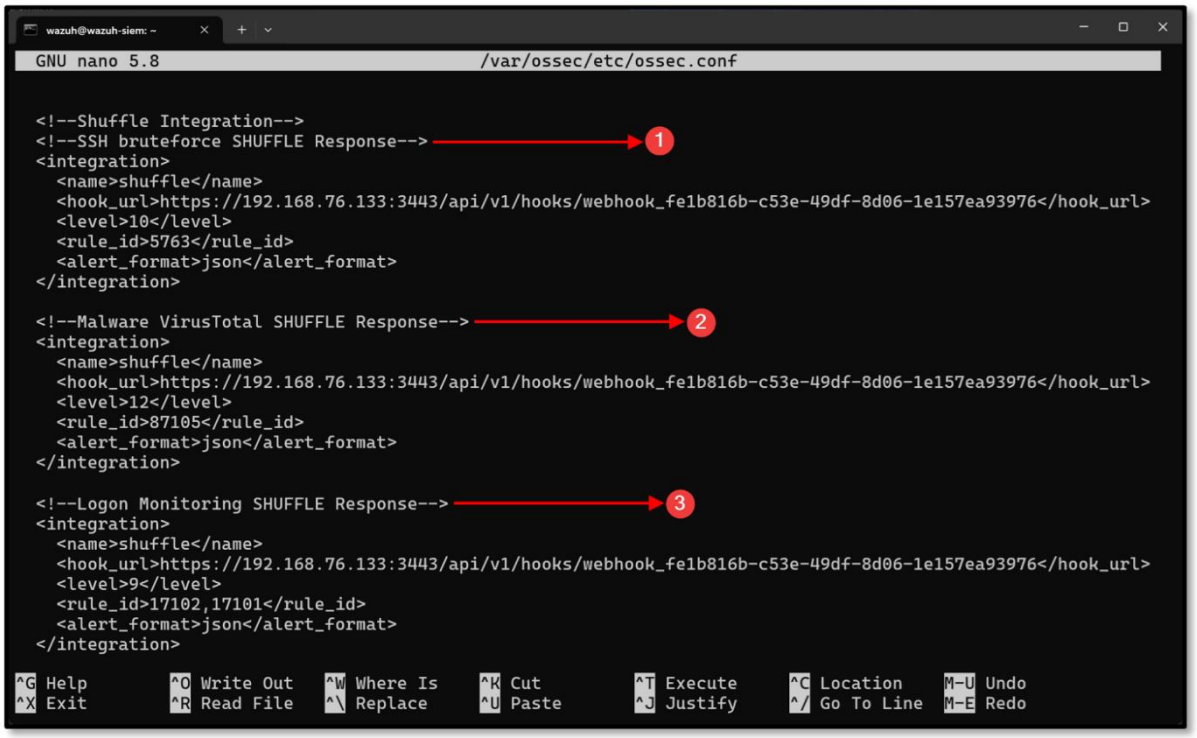


Figure 26: Wazuh ossec.conf file with numbered webhook integration blocks

Enabling Automated IP Blocking (For SSH Brute-Force)

For active response, the <active-response> block in ossec.conf links Rule ID 5763 to Wazuh's built-in firewall-drop command, which uses `iptables` to block the attacker IP for 180 seconds (Figure 27).

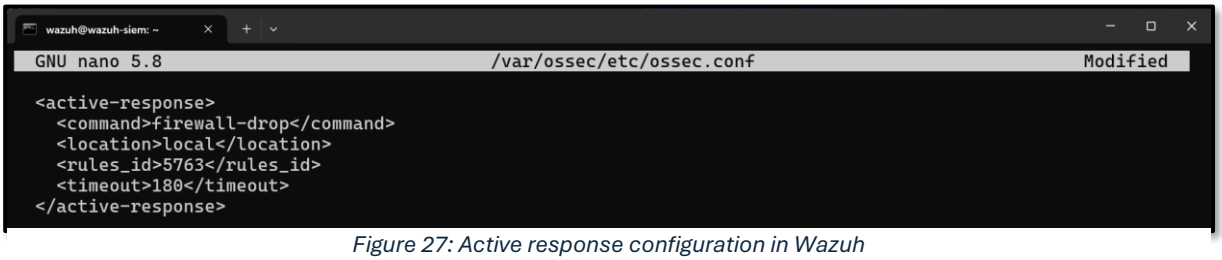


Figure 27: Active response configuration in Wazuh

7.4.10 UC01: Nmap Port Scan Detection

This use case focuses on detecting reconnaissance activity—specifically **Nmap SYN scans**—using **Suricata**, with alerts forwarded to **Wazuh** for correlation and analysis. No active response is triggered; it is strictly for alerting and analyst review.

7.4.10.1 Configuration Summary

Suricata was already integrated with Wazuh as outlined in Section 7.4.3. To detect Nmap SYN scans, a custom rule was added to /etc/suricata/rules/local.rules.

Custom Rule Highlights (See Figure 28):

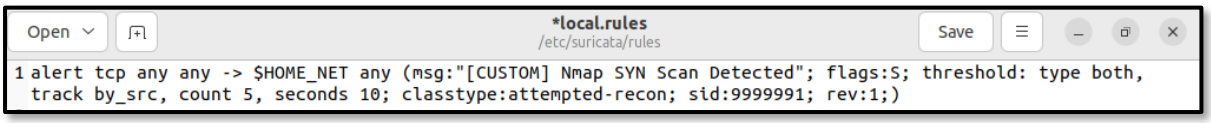


Figure 28: Custom Suricata rule to detect Nmap SYN scans.

- flags:S: Matches SYN packets, commonly used in stealth scans.
- threshold: Triggers an alert if 5 SYN packets are seen within 10 seconds from the same IP.
- msg: Displays a custom message in the alert.
- sid: Unique signature ID for identification.
- classtype: Classifies the activity as reconnaissance.

7.4.10.2 Attack Simulation

An Nmap service version scan was initiated from the **Kali attacker machine** (IP: `192.168.76.131`) using the following command:

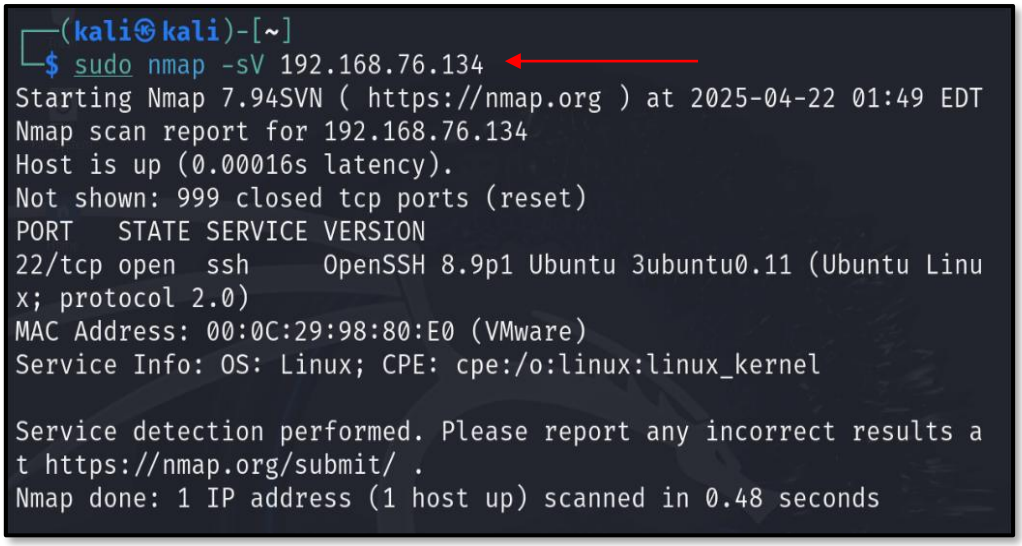
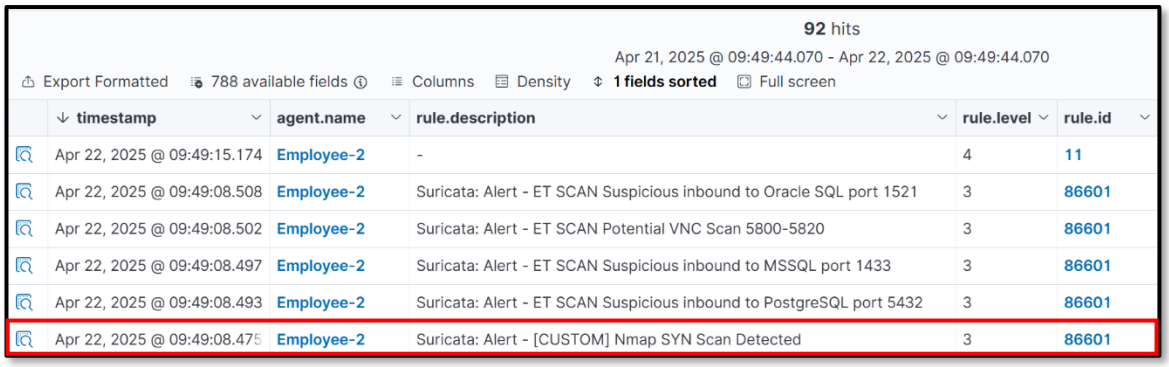


Figure 29: Nmap scan from 192.168.76.131

- -sV: Scans ports and identifies running services and versions—commonly used in the vulnerability mapping phase of an attack.

7.4.10.3 Alert Detection and Validation

Wazuh successfully received and parsed Suricata logs, displaying alerts triggered by the custom rule. This confirms end-to-end functionality: Suricata detects the scan, Wazuh logs the event, and the analyst sees the alert with rich context (see Figure 30 & 31).



92 hits

Apr 21, 2025 @ 09:49:44.070 - Apr 22, 2025 @ 09:49:44.070

Export Formatted | 788 available fields | Columns | Density | 1 fields sorted | Full screen

timestamp	agent.name	rule.description	rule.level	rule.id
Apr 22, 2025 @ 09:49:15.174	Employee-2	-	4	11
Apr 22, 2025 @ 09:49:08.508	Employee-2	Suricata: Alert - ET SCAN Suspicious inbound to Oracle SQL port 1521	3	86601
Apr 22, 2025 @ 09:49:08.502	Employee-2	Suricata: Alert - ET SCAN Potential VNC Scan 5800-5820	3	86601
Apr 22, 2025 @ 09:49:08.497	Employee-2	Suricata: Alert - ET SCAN Suspicious inbound to MSSQL port 1433	3	86601
Apr 22, 2025 @ 09:49:08.493	Employee-2	Suricata: Alert - ET SCAN Suspicious inbound to PostgreSQL port 5432	3	86601
Apr 22, 2025 @ 09:49:08.475	Employee-2	Suricata: Alert - [CUSTOM] Nmap SYN Scan Detected	3	86601

Figure 30: Wazuh Alert of custom nmap scan rule

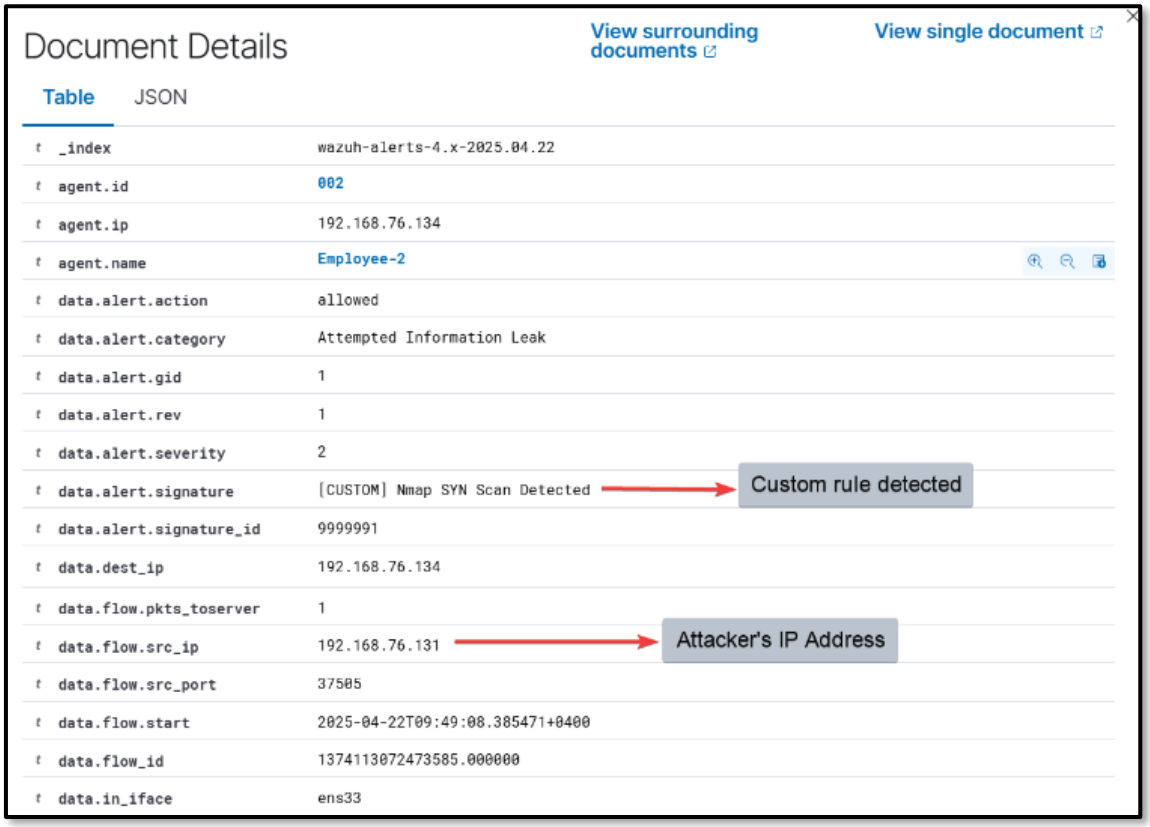


Figure 31: Detailed Log of Nmap scan

7.4.11 UC02: SSH Brute-Force Detection with Active Response

This use case demonstrates how SSH brute-force attempts are detected by Wazuh (Rule ID 5763, Level 10) and automatically mitigated using iptables, with alerts forwarded to Shuffle via the centralized webhook configured in Section 7.4.9. Once received, Shuffle processes the alert and triggers a real-time automated response.

Inside the Shuffle SOAR platform, a Webhook node first receives the incoming Wazuh alert. This is followed by a conditional filter that checks whether the **alert title** equals "**Multiple authentication failure**", confirming a brute-force attempt.

7.4.11.1 Automated Response

After configuring the webhook and log parser, a response action was added to block the attacker's IP using iptables. As shown in Figure 32, the "Block Bruteforce" module used the "Run ssh command" action from the "Popular Actions" menu. The host IP (e.g., 192.168.76.134) was entered, the Username set to root, password is also set.

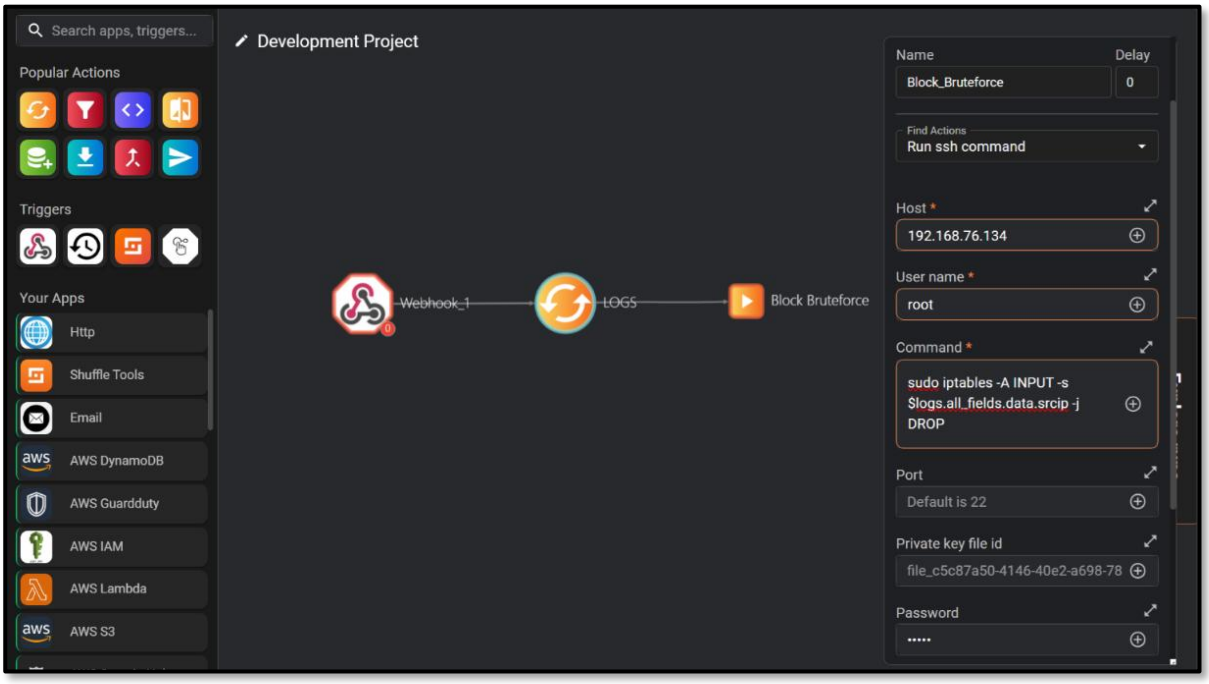


Figure 32: Configuration of the "Block Bruteforce" action

The **Command** field was critical

```
sudo iptables -A INPUT -s $logs.all_fields.data.srcip -j DROP
```

This command dynamically pulls the attacker's IP address (srcip) from the Wazuh alert and injects a rule into the firewall (iptables) to immediately block incoming connections from that source. The use of $logs.all_fields.data.srcip ensures the rule is tailored to the IP triggering the alert. A condition block was added after the Logs node to filter alerts. It checks if the alert title equals **"Multiple authentication failure"**. If true, the workflow continues. If false, it stops (Figure 33).

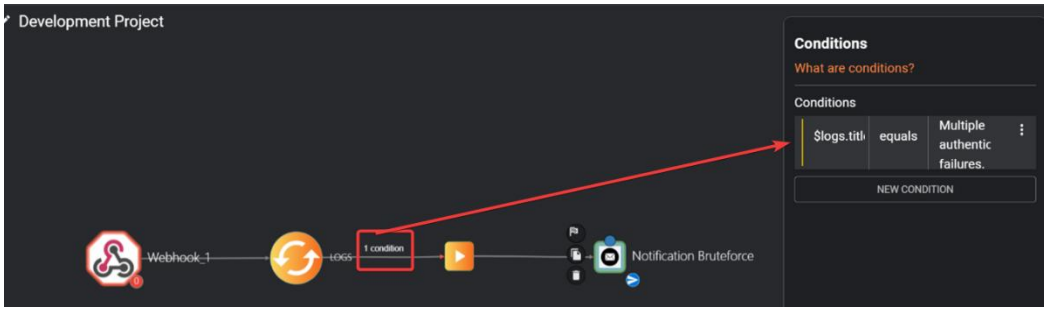


Figure 33: Condition node matching the alert title for SSH brute-force.

7.4.11.2 Email Notification Configuration in Shuffle

To notify analysts of detected brute-force attempts, an email action was added in Shuffle using the "Send email shuffle" node.

- **API Key:** Retrieved from Shuffler.io to authorize the workflow.
- **Recipients:** Directed to the analyst inbox (e.g., socanalystdmu2911@outlook.com).
- **Subject:** Populated dynamically via $logs.title (e.g., "sshd: brute force trying to get access to the system").
- The **Body Preview** feature confirms that fields like $logs.title and $logs.all_fields.data.srcip are correctly filled, ensuring a readable and actionable alert.

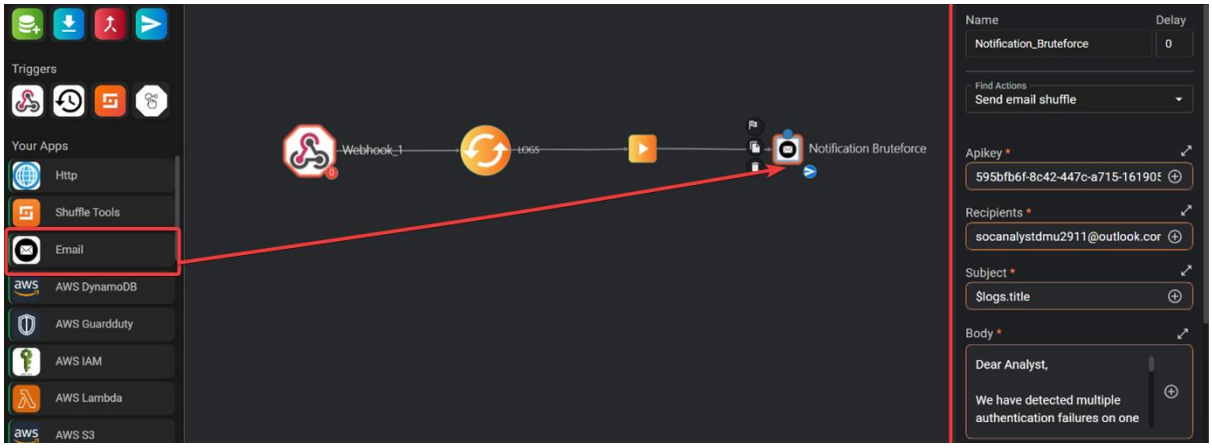


Figure 34: Email Node

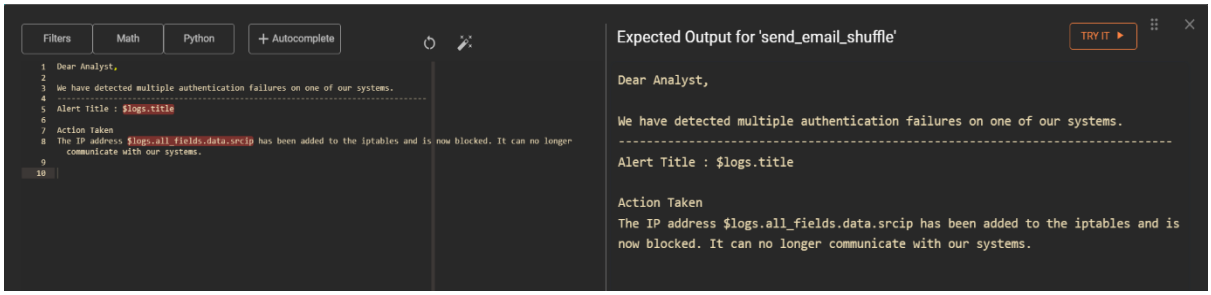


Figure 35: Template of email

7.4.11.3 Attack simulation and verification

A brute-force SSH attack was simulated from the Kali machine using Hydra and a custom passwords.txt file containing common weak credentials, targeting the Ubuntu employee endpoint (Figure 36).

Explanation:

```
hydra -t4 -l employee -P passwords.txt ssh://192.168.76.134
```

- **-t4**: Runs 4 parallel attack threads (increases speed).
- **-l employee**: Sets the target username (employee).
- **-P passwords.txt**: Specifies the password list file to use for attempts.
- **ssh://192.168.76.134**: Targets the SSH service on the victim machine at IP 192.168.76.134

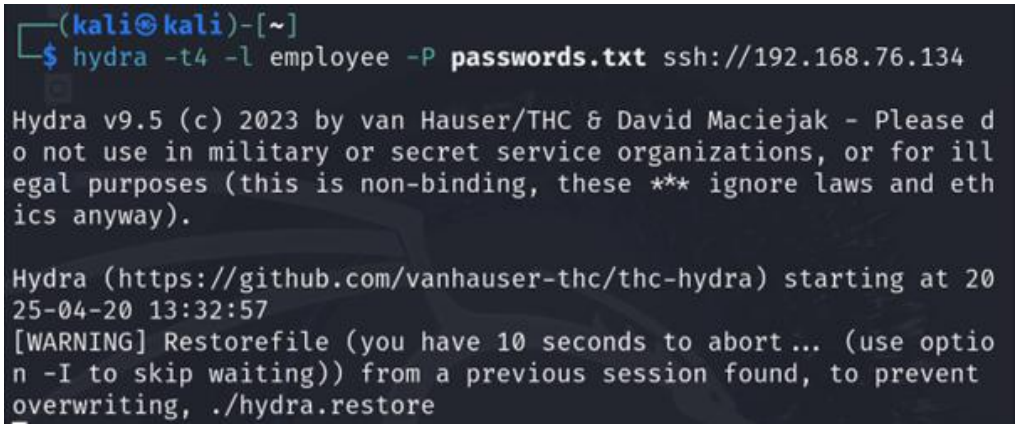


Figure 36: Hydra SSH Attack Command

Wazuh successfully detected multiple authentication failures, triggering Rule Level 5763 level 10. On April 20, 2025, time 21:39 this is triggered after 4 -5 wrong attempts (Figure 37).

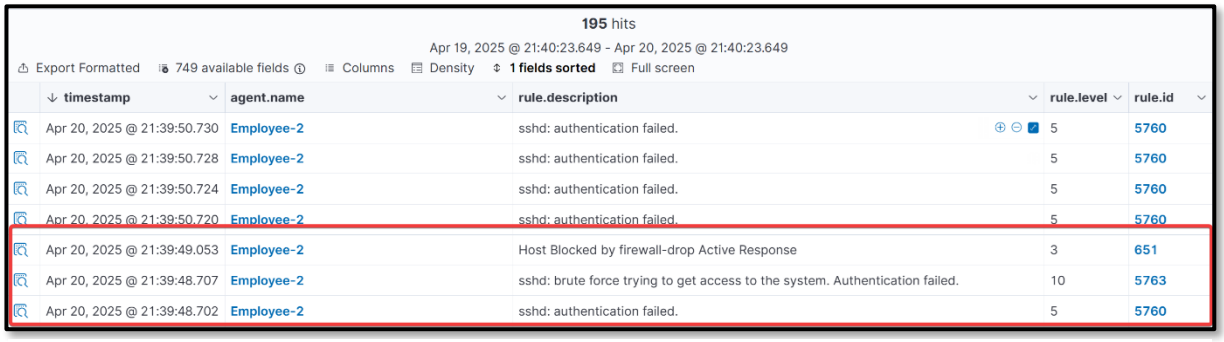


Figure 37: Wazuh Alert Events in Dashboard

The analyst received an alert email, and iptables confirmed the attacker's IP was added to the DROP list, completing the automated response (see Figure 38 and 39).

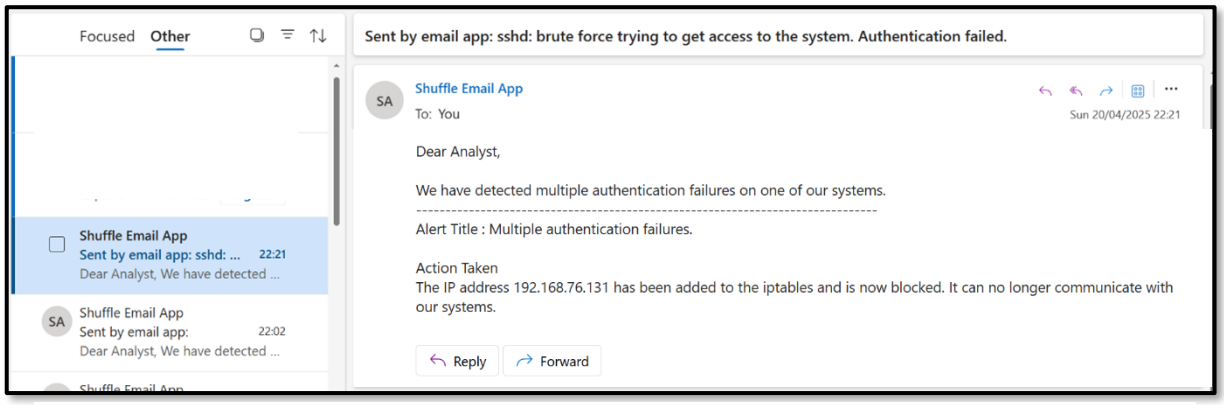


Figure 38: Email Notification Screenshot

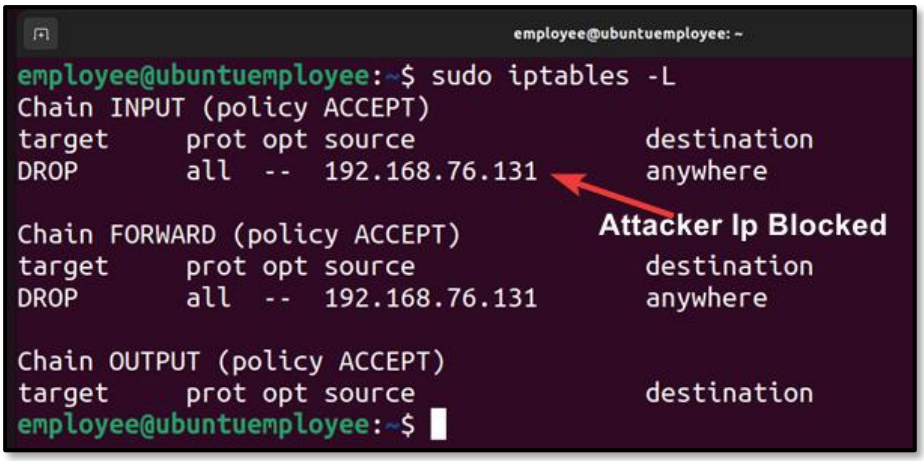


Figure 39: IP Blocked via Iptables

7.4.12 UC03: Malware Detection and Automated Response with VirusTotal and Shuffle

This use case demonstrates automatic malware detection and mitigation using Wazuh, VirusTotal, and Shuffle. When a suspicious file is detected on the endpoint, its hash is submitted to VirusTotal. If flagged as malicious, Wazuh triggers Rule ID 87105 (Level 12), which is forwarded to Shuffle via the centralized webhook (see Section 7.4.9). Suspicious but unconfirmed files trigger separate rules that do not activate this workflow.

Within Shuffle, a new branch was added to the existing workflow. When triggered, it activates a **Run SSH Command** node that:

- Deletes the malicious file using $logs.all_fields.data.virustotal.source.file
- Disables the ens33 interface to contain potential malware spread (see Figure 41)

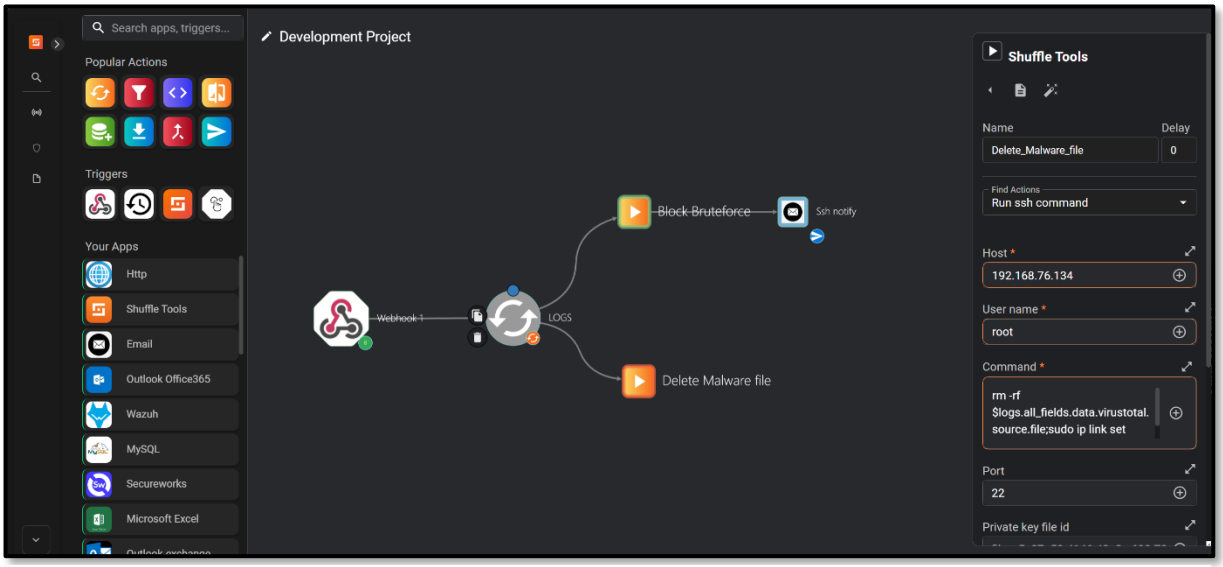


Figure 40: Extended Shuffle workflow

The action uses Run ssh command with root credentials to remove the malware and stop the spread.

Figure 41: Shuffle command deletes malicious file and disables the network interface

A condition block ensures this response only runs when $logs.all_fields.data.virustotal.malicious == 1, confirming the file is verified as malicious (Figure 42).

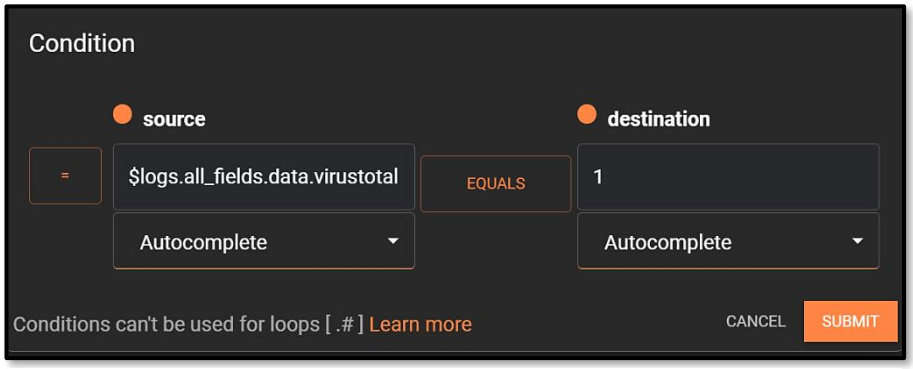


Figure 42: Logical Condition

We add the email node to automate email notifications and a template in which the email is sent.

Figure 43: Email Node Config

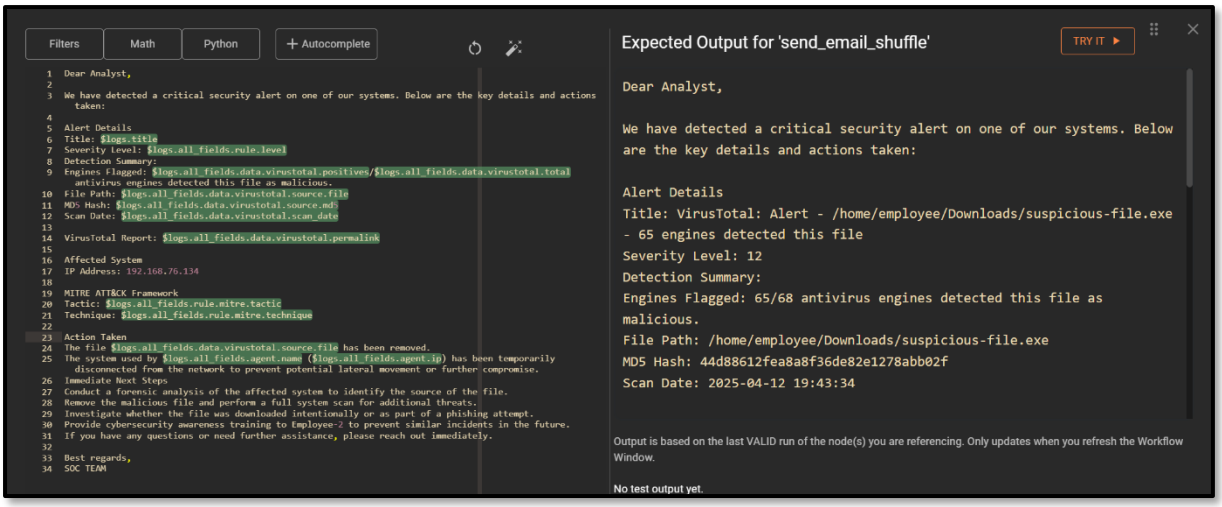


Figure 44: Email Template

7.4.12.1 Attack Simulation and Validation

To simulate this use case:

1. A known EICAR test file was downloaded and saved as *suspicious-file.exe* in the */home/employee/Downloads/* directory (Figure 45).

```
sudo curl -Lo /home/employee/Downloads/suspicious-file.exe
https://secure.eicar.org/eicar.com
```

```
employee@ubuntuemployee:~/Downloads$ sudo curl -Lo /home/employee
/Downloads/suspicious-file.exe https://secure.eicar.org/eicar.com
  % Total    % Received % Xferd  Average Speed   Time    Time
 Time  Current
                                 Dload  Upload   Total   Spent
 Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- -
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- -
100    68  100    68    0     0    119      0 --:--:-- --:--:-- -
--:--:--   119
employee@ubuntuemployee:~/Downloads$ ls
suspicious-file.exe
```

Figure 45: Malicious file downloaded for simulation

Wazuh, integrated with VirusTotal, scanned the file and matched it to **Rule ID 87105** when 66 out of 68 antivirus engines flagged it. On April 21, 2025, at 20:48, Rule ID 87105 was triggered after 66 of 68 engines flagged the file. Since Level 12 indicates a confirmed threat, the automated response was executed immediately.

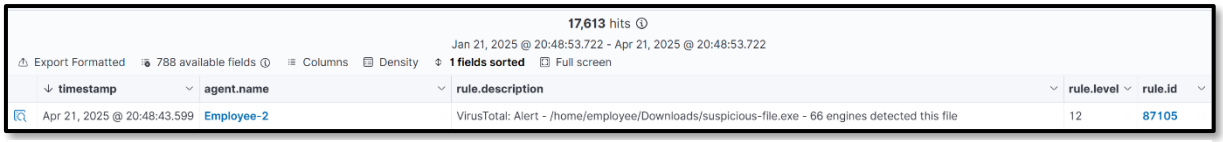


Figure 46: VirusTotal Alert with rule ID 87105 and level 12

Shuffle received the alert and executed the SSH command (Figure 46 and 47):

```
employee@ubuntuemployee:~/Downloads$ ls
suspicious-file.exe
employee@ubuntuemployee:~/Downloads$ ls
employee@ubuntuemployee:~/Downloads$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNO
WN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOW
N group default qlen 1000
    link/ether 00:0c:29:98:80:e0 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
employee@ubuntuemployee:~/Downloads$
```

Figure 47: File deleted and Ip down

The final shuffle playbook for use case 02 and 03

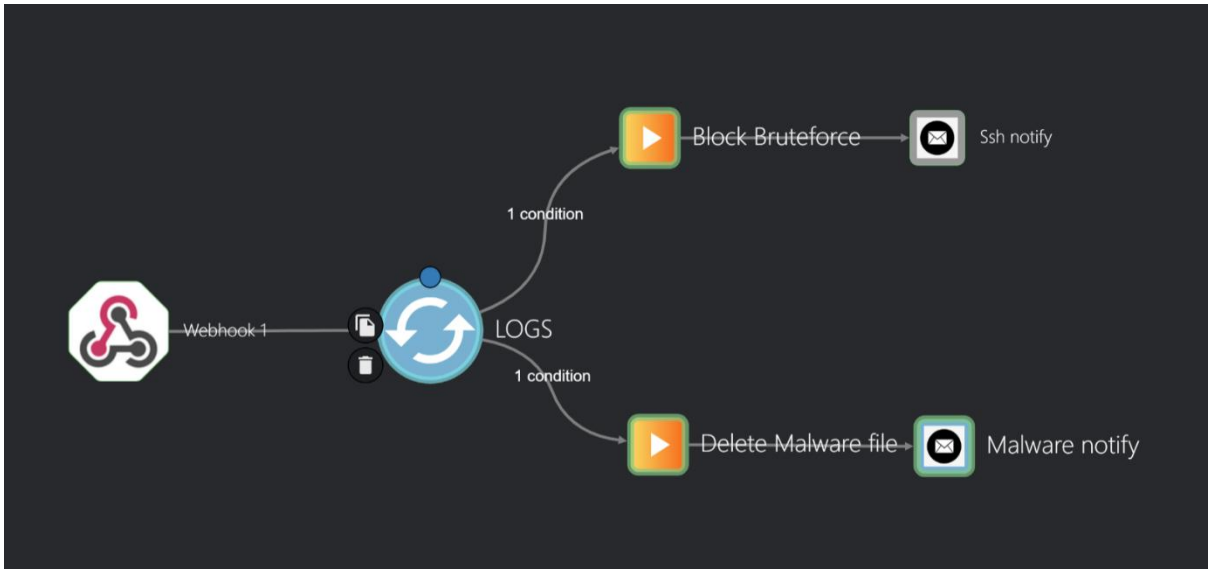


Figure 48: Workflow for UC 02 and UC 03

7.4.12.2 Analyst Notification

Simultaneously, the analyst received a rich notification summarizing (Figure 49):

- File name
- MD5 hash
- VirusTotal permalink
- Detection score
- Affected system and IP
- MITRE ATT&CK tags
- Confirmation of file deletion and network isolation

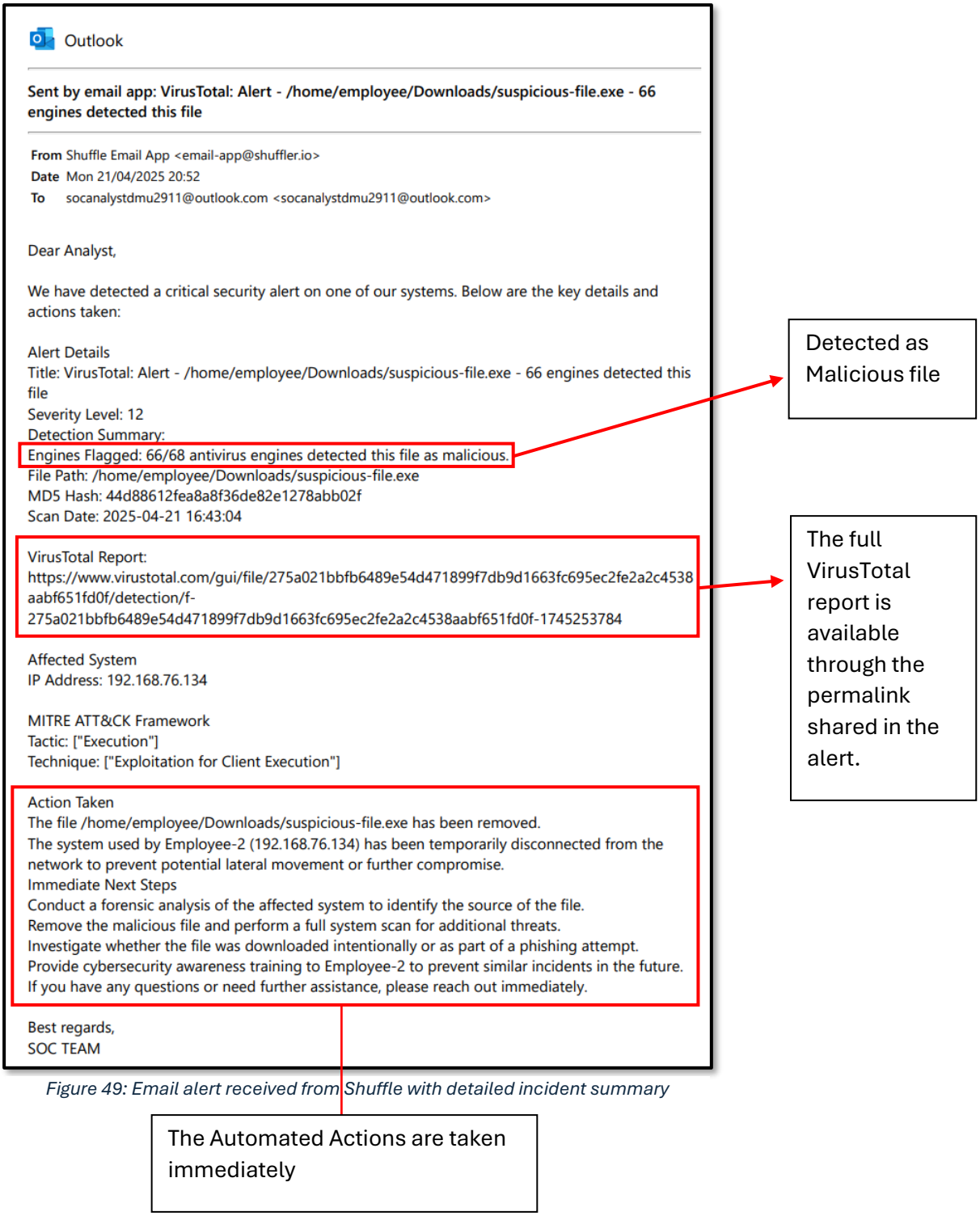


Figure 49: Email alert received from Shuffle with detailed incident summary

7.4.13 UC04: Logon monitoring with time-based alerting

This use case detects employee logins outside of defined business hours (6:00 PM – 8:00 AM) or during weekends. Instead of triggering an active response, an email alert is sent to the analyst for review. Alerts are forwarded to Shuffle via the **centralized webhook configuration detailed in Section 7.4.9**.

7.4.13.1 Wazuh Rule Configuration

Two custom rules were defined in `/var/ossec/etc/rules/local_rules.xml` (Figure 50):

- **Rule ID 17101** – Triggers if a login occurs between **6:00 PM and 8:00 AM**, using <time>6 pm - 8:00 am</time>. Any login during this time — even at 7:00 AM — will generate an alert.
- **Rule ID 17102** – Triggers on **weekends**, defined via <weekday>weekends</weekday>.

```
 GNU nano 5.8                  /var/ossec/etc/rules/local_rules.xml

<group name="custom_login_rules">
  <rule id="17101" level="9">
    <if_sid>5715</if_sid>
    <time>6 pm - 8:00 am</time>
    <description>Successful login during non-business hours.</description>
    <group>login_time,pci_dss_10.2.5,pci_dss_10.6.1,gpg13_7.1,gpg13_7.2,gdpr_IV_35.7.d,gdpr_>
  </rule>

  <rule id="17102" level="9">
    <if_group>authentication_success</if_group>
    <weekday>weekends</weekday>
    <description>Successful login during weekends.</description>
    <group>login_day,pci_dss_10.2.5,pci_dss_10.6.1,gpg13_7.1,gpg13_7.2,gdpr_IV_35.7.d,gdpr_I>
  </rule>
</group>
```

Figure 50: Wazuh rule configuration for off-hours and weekend detection.

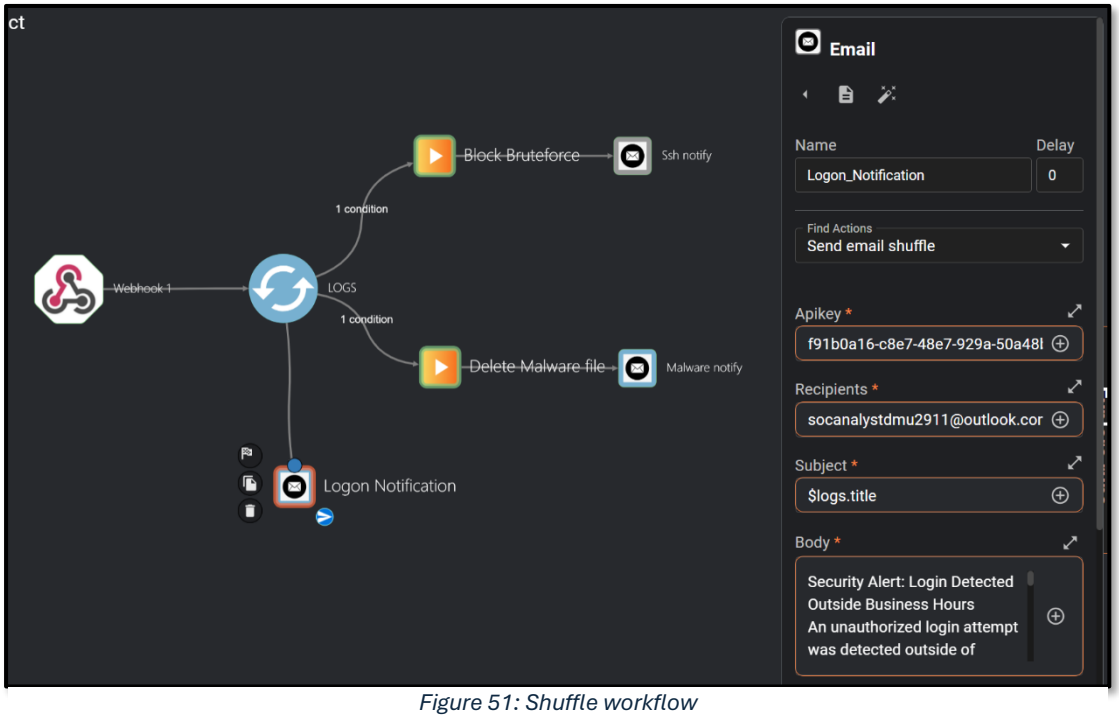


Figure 51: Shuffle workflow

Email Template in Shuffle: The **"Send email shuffle"** action is used. The template dynamically inserts alert fields like hostname, user, timestamp, and IP. A **Python block** is embedded to convert UTC to **Dubai time** for local relevance (Figure 52 and 53).

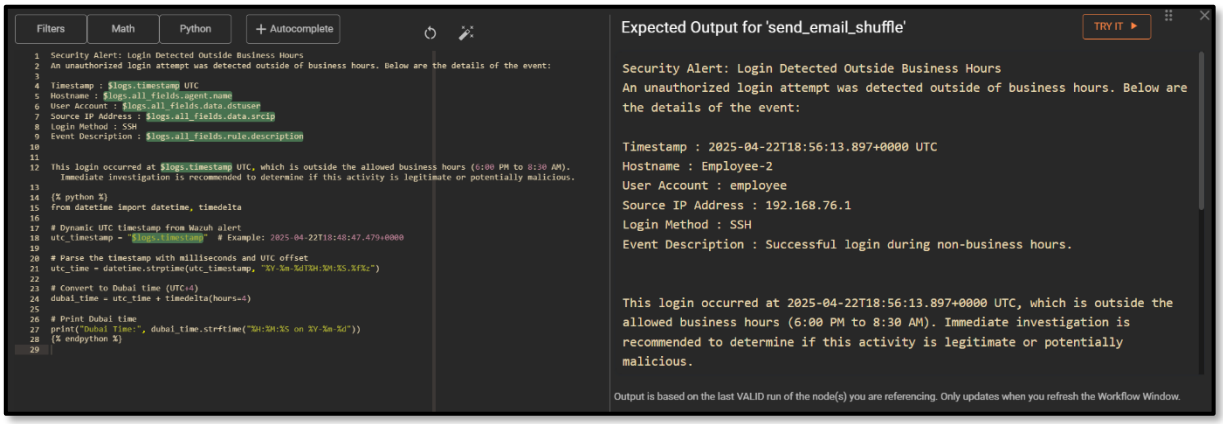


Figure 52: Email Template

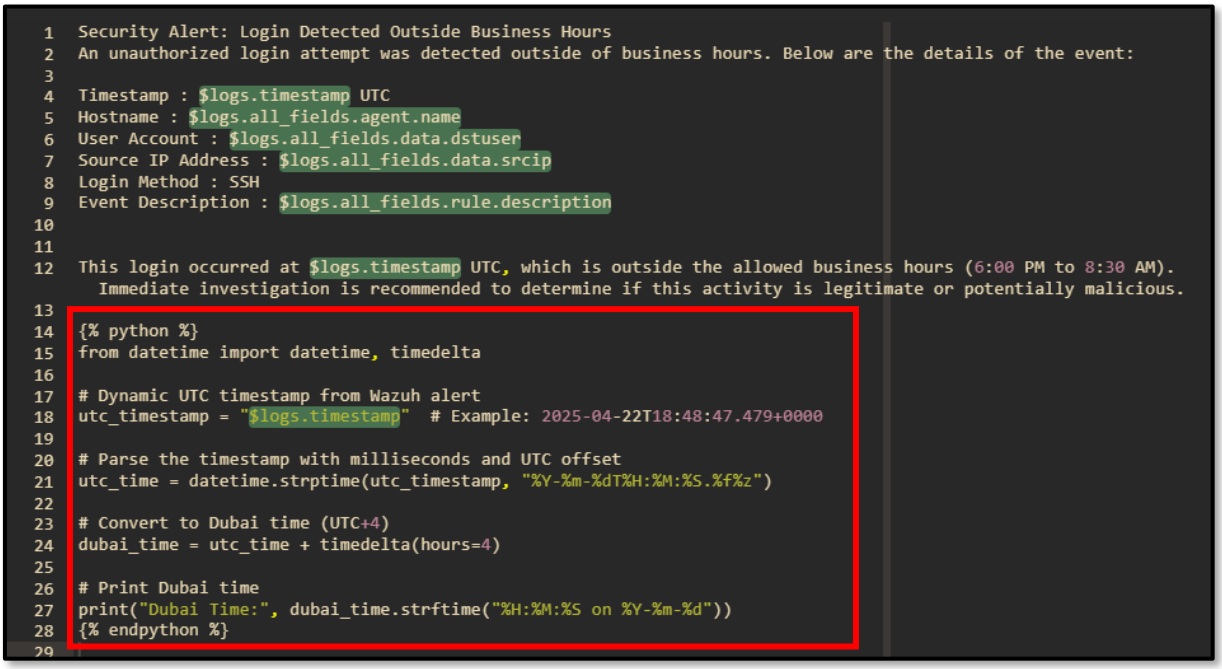


Figure 53: Python script with email

7.4.13.2 Attack Simulation and Alert Verification

To validate this setup, an SSH login was initiated from a personal laptop to the monitored Ubuntu endpoint at **10:55 PM**, which falls outside business hours.

Wazuh successfully detected the event and triggered **Rule ID 17101**. The alert was forwarded to Shuffle, and the analyst received a timestamped email with key details and both UTC and Dubai time clearly included Refer to Figures 54, 55, and 56.

```
PS C:\Users\Meyya> ssh employee@192.168.76.134
employee@192.168.76.134's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

31 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

Figure 54: Ssh attempt

330 hits

Apr 21, 2025 @ 22:55:17.512 - Apr 22, 2025 @ 22:55:17.512

Export Formatted | 1042 available fields | Columns | Density | **1 fields sorted** | Full screen

	timestamp	agent.name	rule.description	rule.level	rule.id
	Apr 22, 2025 @ 22:55:09.883	Employee-2	PAM: Login session opened.	3	5501
	Apr 22, 2025 @ 22:55:09.860	Employee-2	PAM: Login session opened.	3	5501
	Apr 22, 2025 @ 22:55:09.850	Employee-2	Successful login during non-business hours.	9	17101

Figure 55: Wazuh Alert Matching Rule 17101

Sent by email app: Successful login during non-business hours.

SA **Shuffle Email App**

To: You

Tue 22/04/2025 22:57

Security Alert: Login Detected Outside Business Hours
An unauthorized login attempt was detected outside of business hours. Below are the details of the event:

Timestamp : 2025-04-22T18:57:09.951+0000 UTC
Hostname : Employee-2
User Account : employee
Source IP Address : 192.168.76.1
Login Method : SSH
Event Description : Successful login during non-business hours.

This login occurred at 2025-04-22T18:57:09.951+0000 UTC, which is outside the allowed business hours (6:00 PM to 8:30 AM). Immediate investigation is recommended to determine if this activity is legitimate or potentially malicious.

Dubai Time: 22:57:09 on 2025-04-22

Reply Forward

Figure 56: Email Alert with Time Conversion

7.4.14 UC05: File Integrity Monitoring and Ransomware Detection

This use case demonstrates how Wazuh's File Integrity Monitoring (FIM) module is used to detect unauthorized file additions, deletions, and modifications in real time. It not only highlights regular file integrity monitoring but also extends into identifying high-risk patterns such as those exhibited during ransomware attacks. To simulate such behaviour, a ransomware scenario was executed using the wazuh-ransomware-poc.py script available on GitHub (tech-corewin, 2023).

7.4.14.1 Rule IDs and Base Logic

This use case builds on the FIM rules already discussed in **Section 7.4.7**,:

- **550** – File Modified, **553** – File deleted, **554** – File added

These base events serve as input for correlation logic that can detect abnormal file activity.

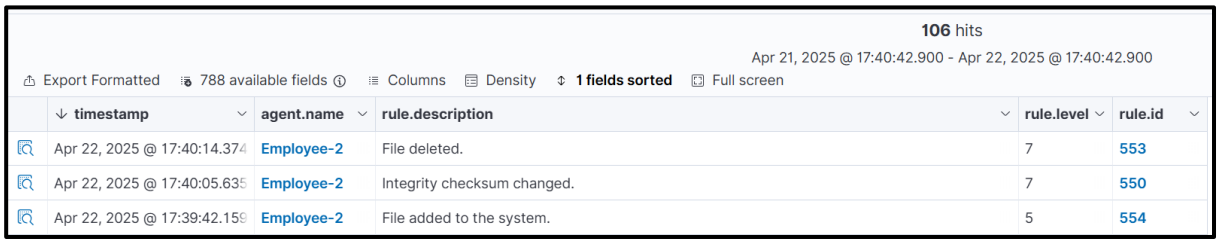


Figure 57: Alerts showing rules 550, 553, and 554 in action

7.4.14.2 Custom Correlation Rule Configuration

To detect potentially malicious file activity, custom rules were added to /var/ossec/etc/rules/local_rules.xml. These rules are designed to correlate rapid file operations that often occur during ransomware attacks. Correlating rules are important because they link multiple related events together, allowing Wazuh to detect complex attack patterns like ransomware, which may not be obvious from individual file changes alone (Figure 58).

- **Rule ID 199910** is triggered if **Rule 553** (file deletion) is matched **20 or more times** within a 30-second window. This helps identify abnormal mass file deletion events, which are often a sign of wiper malware or ransomware cleanup activity.
- **Rule ID 199911** is triggered if **Rule 554** (file creation) is matched **20 or more times** within the same 30-second period. This detects a sudden surge of file additions, which typically occurs when ransomware encrypts existing files and replaces them with .encrypted copies.
- **Rule ID 199912** is a correlation rule that only fires if **both Rules 199910 and 199911 are matched** within the defined time frame. This combination — large numbers of files being deleted and new encrypted files being created — is considered a strong indicator of ransomware behaviour.

By layering these rules, Wazuh moves beyond basic event detection and applies logic that mimics real-world threat patterns. This enables faster and more accurate identification of coordinated or automated attacks such as ransomware.

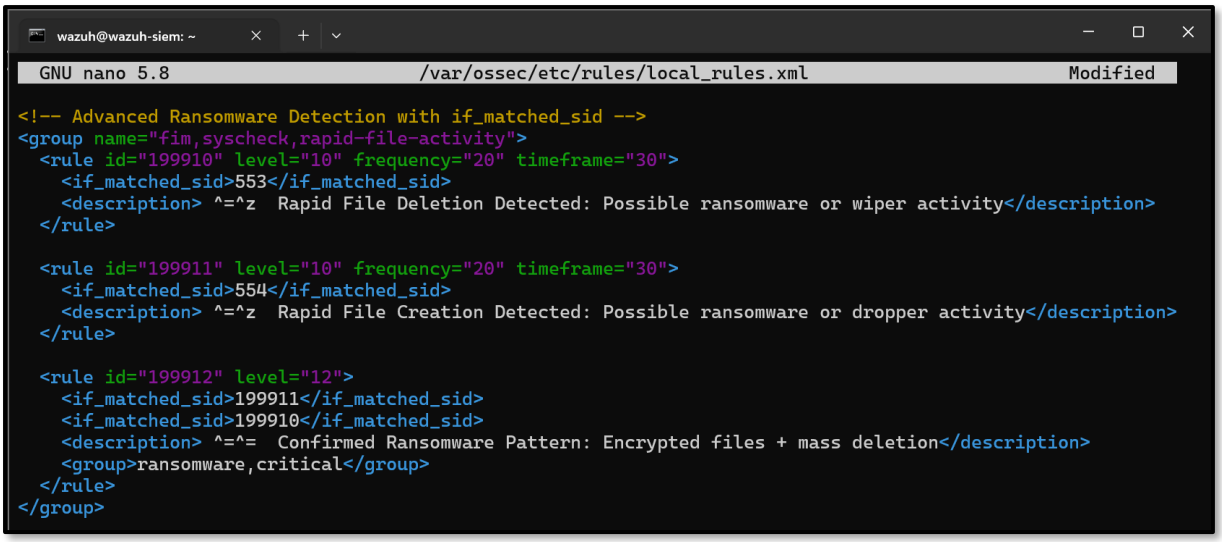


Figure 58: Custom rules with frequency and timeframe logic

7.4.14.3 Real-Time FIM Configuration

```
<directories check_all="yes" whodata="yes" report_changes="yes"
realtime="yes">/home/employee/Downloads/ransomware-
simulation</directories>
```

To monitor file changes, the following directory was added in Agent ossec.conf (Figure 59):

- **check_all**: Monitors all file attributes (size, ownership, permissions, etc.)
- **whodata**: Identifies the user or process responsible for changes
- **report_changes**: Logs what was modified
- **realtime**: Enables immediate detection
- **Target Directory**: This folder was created to simulate a realistic environment where ransomware encrypts files across multiple subdirectories.

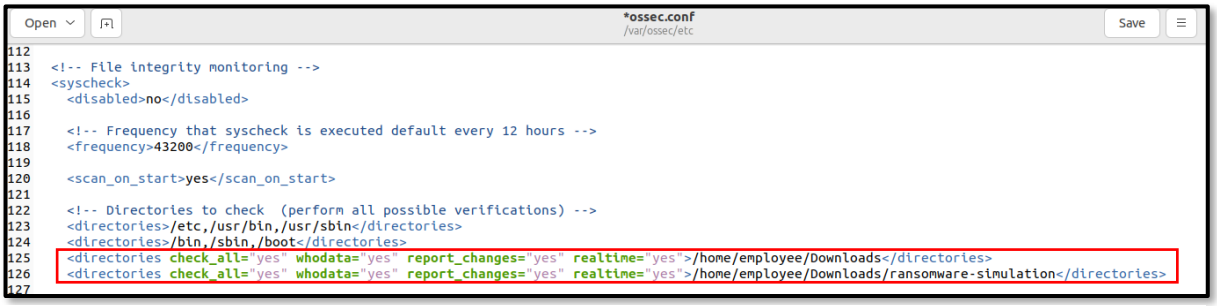


Figure 59: FIM configuration in Agent ossec.conf

7.4.14.4 Ransomware Simulation

To simulate a realistic ransomware scenario, the test environment was first prepared using:

```
python3 wazuh-ransomware-poc.py prepare
```

This script generated multiple `.txt` files across several nested directories inside the monitored folder. The simulated attack was then launched with:

```
python3 wazuh-ransomware-poc.py attack
```

Once executed, the script recursively encrypted all test files and renamed them with the .encrypted extension, mimicking real-world ransomware behaviour.

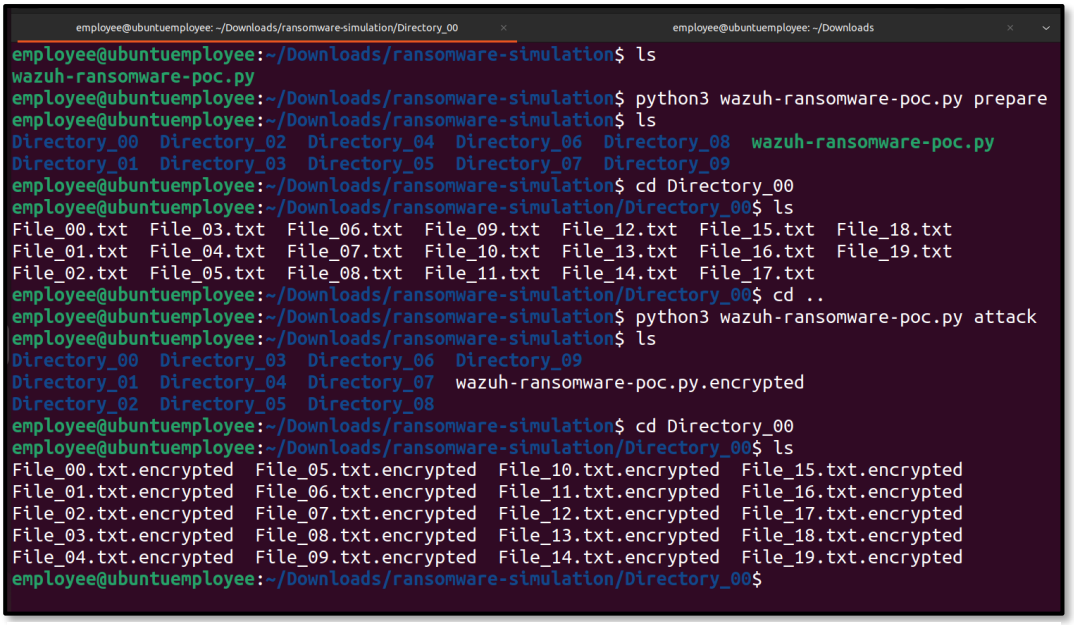


Figure 60: Directory view before and after the simulated ransomware attack

7.4.14.5 Alerting and Detection

Wazuh detected a surge of file creation and deletion events, triggering:

- **199910** – Mass deletions
- **199911** – Mass file creations
- **199912** – Correlated pattern resembling ransomware

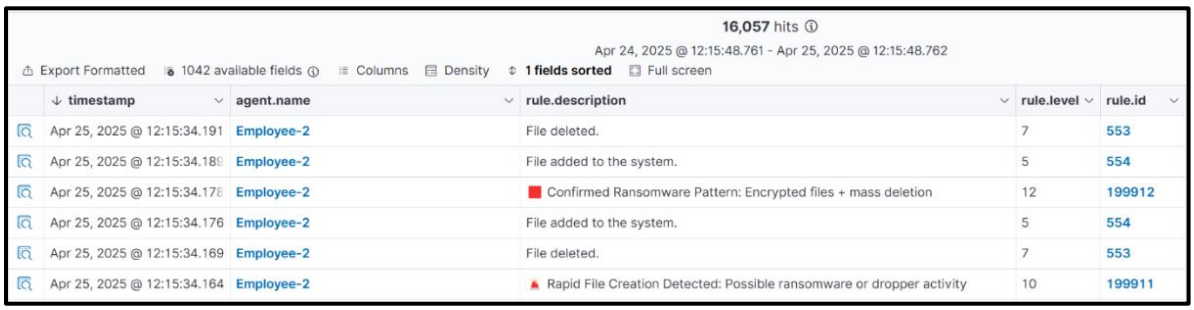


Figure 61: Wazuh dashboard showing correlated alert events

7.4.15 UC06: Data Exfiltration Detection Using Suricata

This use case demonstrates how custom Suricata rules were implemented to detect the unauthorized download of confidential files from an internal endpoint, simulating a basic data exfiltration attempt Suricata was already integrated with Wazuh as outlined in **Section 7.4.3**.

7.4.15.1 Suricata Custom Rules Configuration

Two custom rules were added to /etc/suricata/rules/local.rules to monitor for unauthorized access to sensitive files (Figure 62):

- Rule 1 detects HTTP requests for /confidential_report.txt, a file containing confidential employee data.
    - If any system inside the monitored network requests this specific file over HTTP, an alert is generated.
    - This rule is important for catching the unauthorized transfer of sensitive plain-text information.
- Rule 2 detects HTTP requests for /confidential_archive.zip, representing a staged archive of confidential documents.
    - The goal of this rule is to identify bulk data staging, where attackers collect multiple sensitive files into a compressed format for easier exfiltration.

Both rules are triggered when an HTTP GET request contains the specific file name in the URL path. The activity is classified as a policy violation, indicating potential insider threat, accidental leak, or active data exfiltration.

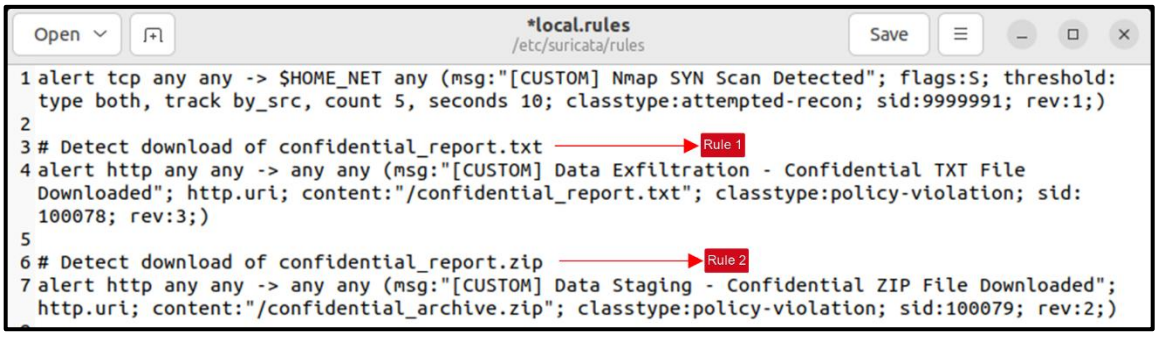


Figure 62: Suricata rules config

7.4.15.2 Environment Setup

A lightweight Python HTTP server was started on the Ubuntu employee endpoint to serve the confidential files (Figure 63):

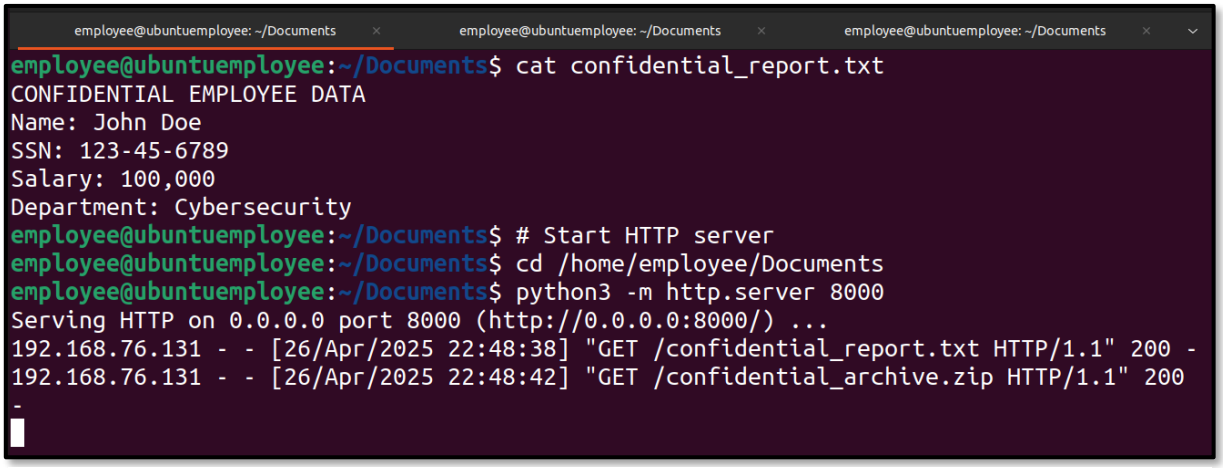


Figure 63: Python HTTP server running on port 8000

7.4.15.3 Attack Simulation

The Kali attacker machine simulated a data exfiltration attempt by downloading the files using curl:

```
$ curl http://192.168.76.134:8000/confidential_report.txt
$ curl http://192.168.76.134:8000/confidential_archive.zip
```

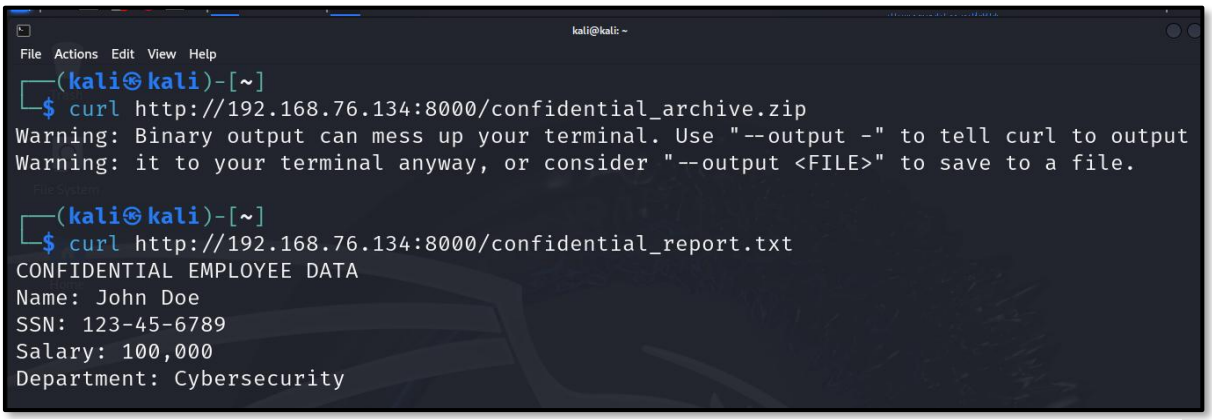
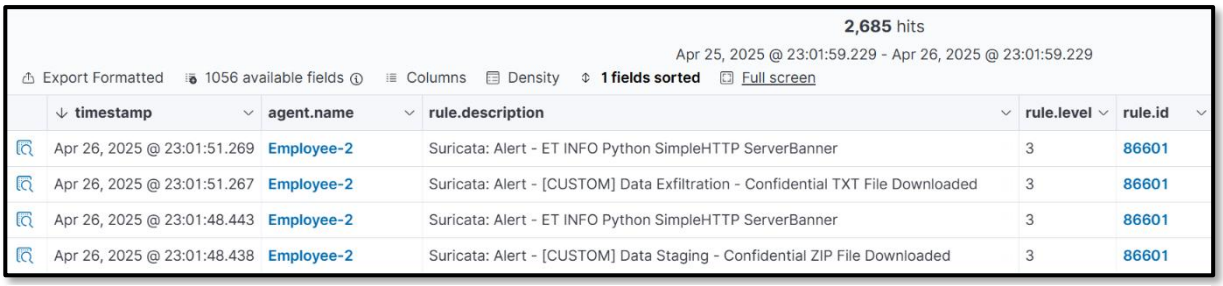


Figure 64: Kali Linux downloading the confidential text and archive files

7.4.15.4 Detection and Alert Verification

Suricata captured the file transfer events and triggered alerts for both Rule 1 and Rule 2. These were parsed and forwarded to Wazuh, where they were visible in the dashboard under the associated Suricata rules.



2,685 hits

Apr 25, 2025 @ 23:01:59.229 - Apr 26, 2025 @ 23:01:59.229

Export Formatted | 1056 available fields | Columns | Density | 1 fields sorted | Full screen

timestamp	agent.name	rule.description	rule.level	rule.id
Apr 26, 2025 @ 23:01:51.269	Employee-2	Suricata: Alert - ET INFO Python SimpleHTTP ServerBanner	3	86601
Apr 26, 2025 @ 23:01:51.267	Employee-2	Suricata: Alert - [CUSTOM] Data Exfiltration - Confidential TXT File Downloaded	3	86601
Apr 26, 2025 @ 23:01:48.443	Employee-2	Suricata: Alert - ET INFO Python SimpleHTTP ServerBanner	3	86601
Apr 26, 2025 @ 23:01:48.438	Employee-2	Suricata: Alert - [CUSTOM] Data Staging - Confidential ZIP File Downloaded	3	86601

Figure 65: Wazuh dashboard showing detection of data exfiltration activity

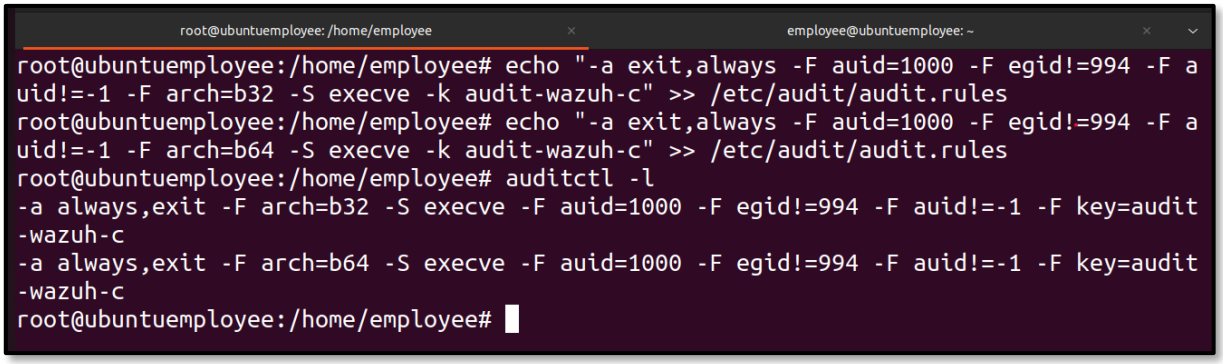
---

7.4.16 UC07: Suspicious Command Monitoring and Data Exfiltration Detection

This use case demonstrates how Wazuh, combined with Auditd, was configured to monitor command executions on the endpoint, detect suspicious activities, and raise alerts for potential data exfiltration attempts mostly **(configured in section 7.4.4)**.

7.4.16.1 Audit Rules Configuration

Audit rules were appended to `/etc/audit/audit.rules` to monitor the `execve` system call for command execution activities:



root@ubuntuemployee: /home/employee | employee@ubuntuemployee: ~

```
root@ubuntuemployee:/home/employee# echo "-a exit,always -F auid=1000 -F egid!=994 -F a
uid!=-1 -F arch=b32 -S execve -k audit-wazuh-c" >> /etc/audit/audit.rules
root@ubuntuemployee:/home/employee# echo "-a exit,always -F auid=1000 -F egid!=994 -F a
uid!=-1 -F arch=b64 -S execve -k audit-wazuh-c" >> /etc/audit/audit.rules
root@ubuntuemployee:/home/employee# auditctl -l
-a always,exit -F arch=b32 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit
-wazuh-c
-a always,exit -F arch=b64 -S execve -F auid=1000 -F egid!=994 -F auid!=-1 -F key=audit
-wazuh-c
root@ubuntuemployee:/home/employee#
```

Figure 66: Audit rules added for command monitoring

The ossec.conf on the Wazuh agent was updated to parse the audit logs. This allows Wazuh to ingest audit events in real time.

Open | *ossec.conf /var/ossec/etc | Save

```
185
186   <localfile>
187     <log_format>audit</log_format>
188     <location>/var/log/audit/audit.log</location>
189   </localfile>
```

Figure 67: Wazuh Agent ossec.conf configuration for Auditd log collection

7.4.16.2 Custom Wazuh Rules for Suspicious Commands

Custom correlation rules were created in /var/ossec/etc/rules/local_rules.xml (Figure 68):

- **Rule 100210** (Level 12): Detects execution of suspicious commands such as nmap, nc, hydra, sqlmap, tcpdump, and others typically associated with network scanning or reconnaissance.
- **Rule 100215** (Level 10): Detects execution of potential data exfiltration tools like scp, curl, wget, ftp, and ssh.

```
 GNU nano 5.8                                  /var/ossec/etc/rules/local_rules.xml

<group name="audit,suspicious_command">
  <rule id="100210" level="12">
    <if_sid>80792</if_sid>
    <field name="audit.command">nmap|nc|hydra|ncat|netcat|sqlmap|tcpdump|nikto</field>
    <description> ^=^z  Suspicious Command Detected: $(audit.command)</description>
  </rule>
</group>

<group name="audit,lolbins_detection">
  <rule id="100215" level="10">
    <if_sid>80792</if_sid>
    <field name="audit.command">scp|curl|wget|ftp|ssh</field>
    <description> ^=^z  Potential Data Exfiltration $(audit.command)</description>
  </rule>
</group>
```

Figure 68: Custom Wazuh rules for suspicious and exfiltration command detection

Each rule triggers when a matching command is found in the audit.command field.

7.4.16.3 Attack Simulation and Detection

To test the setup:

- Commands like nc and ssh were executed from the Ubuntu endpoint.
- Auditd captured the executions, Wazuh parsed the logs, and custom rules triggered alerts.

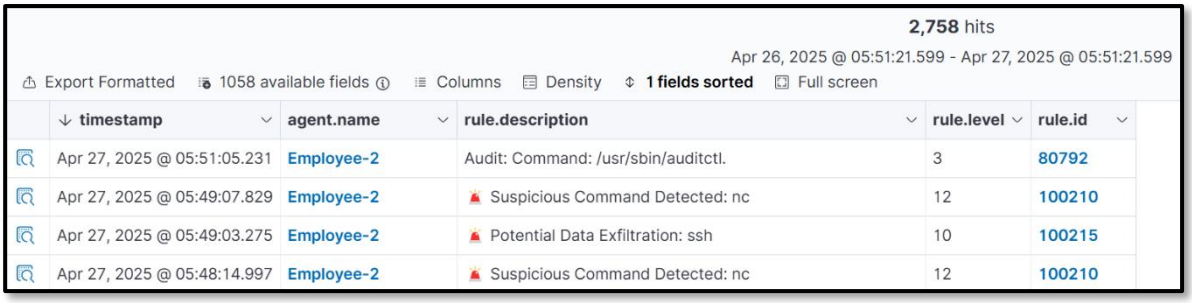


Figure 69: Wazuh dashboard showing detection of suspicious and data exfiltration commands

---

7.5 CHALLENGES ENCOUNTERED

During project development, some technical issues were faced:

- **Wazuh Manager Configuration Issues:** Certain custom rules and active responses initially did not trigger as expected. These were resolved by consulting the **official Wazuh Discord community**, where real-time troubleshooting support was provided.

- **Shuffle Workflow Errors:** In Shuffle, API connection issues and incorrect data parsing between webhook and logs node were frequent. These were resolved using **official Shuffle Discord support** and reviewing webhook payloads carefully.
- **Timezone Conversion Challenges:** Converting UTC login timestamps to Dubai time inside Shuffle's workflow required building a Python block manually, after standard templates failed.

8 TEST PLAN

The project underwent extensive testing to validate that each implemented security use case operated as expected under simulated real-world conditions. Attack scenarios were carefully crafted to match the objective of each use case, and both detection and response actions were evaluated.

- For **UC01: Nmap Port Scan Detection**, an Nmap service version scan (nmap -sV) was launched from the Kali machine against the monitored Ubuntu endpoint. This tested Suricata's and Wazuh ability to detect and alert on reconnaissance activity.
- In **UC02: SSH Brute-Force Detection with Active Response**, Hydra was used to simulate multiple login attempts against SSH. The system was evaluated for its detection capability and automatic IP blocking via iptables.
- **UC03: Malware Detection and Automated Response** involved downloading a flagged file onto the endpoint. The workflow was tested to ensure that VirusTotal correctly identified the file, the file was deleted, and the endpoint's network interface was disabled.
- **UC04: Logon Monitoring with Time-Based Alerting** was tested by performing SSH logins outside allowed business hours. Wazuh successfully detected the off-hours access, and Shuffle generated an alert email with the login time converted into Dubai local time.
- In **UC05: File Integrity Monitoring and Ransomware Detection**, rapid creation and deletion of files were simulated within a monitored directory. The system's ability to detect abnormal file system changes without generating excessive alerts was verified.
- **UC06: Data Exfiltration Detection** was evaluated by using scp, curl, and wget commands to transfer sensitive files externally. Suricata rules were tested to ensure outbound traffic violations were caught.
- For **UC07: Suspicious Command Monitoring with AI Enrichment**, suspicious commands like nmap, nc, and scp were executed on the endpoint. Wazuh successfully detected the executions, and Shuffle generated enriched, human-readable alert explanations.

Each attack was manually triggered, observed through Wazuh's dashboard, and verified through alert logs, Shuffle workflows, and automated email notifications where applicable. The test cases are explained clearly in the below section.

8.1 TEST CASES

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Outcome
TC 01	Wazuh Agent Enrollment and Manager Setup	Install Wazuh agent. Enroll endpoint. Verify dashboard connectivity.	Agent enrolled and active in Wazuh dashboard.	Agent enrolled successfully and active.	Pass
TC 02	Suricata Deployment and Wazuh Integration	Install Suricata. Configure EVE-JSON. Simulate network scans.	Suricata alerts appear in Wazuh.	Suricata alerts parsed and displayed.	Pass
TC 03	Auditd Configuration for FIM	Configure Auditd. Monitor sensitive directories. Modify monitored files.	Unauthorized changes trigger Wazuh alerts.	File modification detected by Wazuh.	Pass
TC 04	Shuffle SOAR Deployment	Install Shuffle via Docker. Create test workflow. Verify accessibility.	Shuffle is accessible and workflows are operable.	Shuffle deployed successfully.	Pass
TC 05	VirusTotal Integration with Wazuh	Configure VirusTotal API. Drop suspicious file. Observe enriched alerts.	File hashes submitted to VirusTotal, scan results shown.	Enrichment successful, VirusTotal reports attached.	Pass
TC 06	API Key Retrieval and Email Notification in Shuffle	Retrieve API key from Shuffler.io. Configure email node. Test email sending.	Email alerts triggered automatically using API key.	Email alerts sent correctly.	Pass
TC 07	Webhook Integration from Wazuh to Shuffle	Set up webhook for critical alerts. Trigger alert. Verify reception in Shuffle.	Alerts automatically received and trigger Shuffle workflows.	Workflow triggers successfully on critical alerts.	Pass
TC 08	Nmap Port Scan Detection (UC01)	Run Nmap scan (-sV) from Kali. Monitor Wazuh alerts.	Wazuh detects port scanning activity.	Scan attempt detected accurately.	Pass
TC 09	SSH Brute-Force Detection and Active Response (UC02)	Launch SSH brute-force via Hydra. Verify Wazuh alert. Confirm attacker IP block.	Brute-force detected, and attacker's IP automatically blocked via Shuffle.	Attacker IP blocked, alert generated.	Pass

TC 10	Malware Detection and Response (UC03)	Upload malicious file. Confirm VirusTotal scan. Observe file deletion and network isolation.	Malicious file deleted; network disabled to contain threat.	Automated deletion and isolation performed.	Pass
TC 11	Logon Monitoring Outside Business Hours (UC04)	SSH outside allowed hours (6 PM to 8:30 AM). Monitor email alert with Dubai timestamp.	Login alert sent with converted Dubai time.	Alert generated correctly; Dubai time included.	Pass
TC 12	File Integrity Monitoring and Ransomware Detection (UC05)	Simulate rapid file creation/deletion. Verify Wazuh FIM alert and ransomware correlation.	Wazuh triggers ransomware detection based on rapid file events.	Rapid changes flagged as ransomware behavior.	Pass
TC 13	Data Exfiltration Detection with Suricata (UC06)	Perform file transfer via SCP/Netcat. Monitor outbound traffic alerts.	Unauthorized exfiltration attempt detected by Suricata/Wazuh.	Data theft detected successfully.	Pass
TC 14	Suspicious Command Monitoring and Detection (UC07)	Execute Netcat, SCP, Curl, Wget. Observe Wazuh alert generation.	Suspicious commands flagged and alerts generated.	Suspicious activity detected and alerted.	Pass
TC 15	FIM Alerting Enhancement	Create small file modifications. Validate correlation of single alert for mass changes.	Alert generated summarizing rapid changes.	Consolidated alert triggered.	Pass
TC 16	Shuffle Conditional Filtering	Create non-relevant alert. Test Shuffle conditional node stopping irrelevant workflows.	Non-matching alerts filtered without triggering workflow.	Workflow correctly filtered and not executed.	Pass
TC 17	Email Notification Format Validation	Trigger login anomaly. Verify structured email report with alert fields.	Email body contains timestamp, IP, user, and Dubai time.	Email structured as expected.	Pass
TC 18	Multiple Use Case Workflow Handling	Trigger Nmap scan. Trigger SSH brute-force. Upload malware sequentially. Validate Shuffle branching.	Each type of attack triggers correct branch in workflow.	Parallel workflows executed accurately.	Pass

Table 7: Test cases and outcomes

8.2 RESULTS

All implemented use cases were successfully tested under simulated threat conditions in a controlled virtual lab environment. Each test case passed as expected, demonstrating the system's ability to detect and respond to various attack scenarios, including reconnaissance, brute-force attacks, malware detection, file integrity violations, and data exfiltration attempts. While the data exfiltration scenario was effective in a lab setting, its detection approach may require further refinement for reliable deployment in complex real-world environments. Overall, the results confirmed that both detection and automated responses were executed correctly without manual intervention.

8.3 ANALYSIS

During testing, the system's actual performance closely matched the expected outcomes for all major use cases. Detection mechanisms consistently triggered appropriate alerts, and automated workflows executed correct response actions such as IP blocking, file deletion, and email notifications.

A minor observation was noted in the Data Exfiltration Detection use case (UC06). While basic file transfer attempts using SCP, Wget, and Netcat were detected, the system may be less effective against more sophisticated and real-world exfiltration methods involving encryption or tunnelling. This reflects a practical limitation inherent in network-based detection models without full deep packet inspection.

Aside from minor alert generation delays during bulk file operations, no critical anomalies or system failures were encountered. Overall, the solution behaved reliably and demonstrated strong real-world applicability within the scope of the project.

8.4 RISK ASSESSMENT

While the testing phase demonstrated high system reliability, several risks remain. False positives may occur, especially in file integrity and login monitoring use cases, requiring periodic rule tuning. Shuffle workflows depend on external services like VirusTotal and Shuffler.io; any downtime could affect alert enrichment or email notification delivery.

In the case of data exfiltration detection, the current Suricata rules are effective for basic methods but may miss sophisticated or encrypted exfiltration techniques. Additional layers, such as deep packet inspection or behavioural analytics, would be required for enhanced protection. Finally, large-scale deployment would require scalability planning to prevent alert flooding or resource exhaustion. Regular system audits, backup workflows, and rule optimization strategies are recommended to mitigate these risks.

9 DISCUSSION

9.1 INTERPRETATION OF FINDINGS

- The integrated system combining Wazuh, Suricata, and Shuffle successfully achieved real-time threat detection and automated response without human intervention.
- Real-world threats such as reconnaissance, brute-force attacks, malware infections, unauthorized file changes, and suspicious command executions were detected and mitigated effectively.
- Automation workflows reduced the time between detection and response, validating the objective of minimizing manual analyst workload.
- System logs and Shuffle workflows confirmed that security incidents were handled appropriately according to predefined policies.
- Overall, the results demonstrate that affordable, open-source tools can be effectively combined to meet cybersecurity needs for small to medium-sized enterprises.

9.2 IMPLICATIONS

The outcomes of this project demonstrate several critical implications for cybersecurity practice and research:

- The project validates that organizations operating with constrained cybersecurity budgets can still deploy a robust and comprehensive threat detection and response framework using open-source tools, without dependence on costly commercial software solutions.
- Automation of key incident response actions—such as IP blocking, malware quarantine, and real-time alerting—significantly enhances response efficiency while minimizing the likelihood of human error during high-pressure scenarios.
- By integrating Security Information and Event Management (SIEM), Intrusion Detection System (IDS), and Security Orchestration, Automation, and Response (SOAR) functionalities, the system exemplifies a layered security architecture that supports the defense-in-depth model. This approach ensures that threats are addressed at multiple levels, enhancing overall resilience.
- The implementation of Shuffle-based workflows streamlines alert triage and response, effectively reducing alert fatigue and enabling security analysts to focus on high-priority and actionable threats.
- The findings reinforce the practical viability of adopting open-source cybersecurity platforms in operational environments. They align with current research advocating for modular, adaptable, and community-driven security architectures, encouraging broader adoption across industry and academia.

9.3 LIMITATIONS

- Data exfiltration detection was effective for basic file transfer methods but may be less reliable against encrypted or stealthy exfiltration techniques without deeper network inspection.
- System scalability was not evaluated at enterprise scale; in high-load environments, resource contention and performance bottlenecks could arise.
- Dependency on third-party services like VirusTotal and Shuffler.io introduces risk if these services become unavailable, impacting enrichment workflows.
- Despite tuning, the system could generate false positives, especially during noisy network activities or legitimate bulk file operations.
- Workflow configurations must be continuously monitored and updated to adapt to the evolving threat landscape, ensuring that automated responses remain effective against emerging attack vectors.
- Cloud environments were not included in this implementation, limiting applicability in hybrid or fully cloud-based infrastructures.

9.4 RECOMMENDATIONS FOR FUTURE WORK

- Enhancing detection mechanisms for encrypted and covert data exfiltration by incorporating deep packet inspection and behavioral anomaly analysis.
- Testing the system under high-volume and prolonged operational conditions to evaluate scalability and resource optimization needs.
- Expanding Shuffle workflows to include additional real-world attack scenarios, such as phishing detection, privilege escalation monitoring, and insider threat detection.
- Developing backup strategies for API-dependent services to maintain critical alerting and response functions during external service outages.
- Integrating external threat intelligence feeds to enable proactive detection based on emerging global threat data.
- Exploring the use of lightweight machine learning algorithms to detect anomalies and enhance proactive threat identification without heavy manual tuning.
- Extend the solution to support cloud-native environments for broader deployment flexibility and scalability.
- Enhance AI integration to provide deeper, context-aware analysis of security events and reduce alert fatigue.

10 APPENDIX

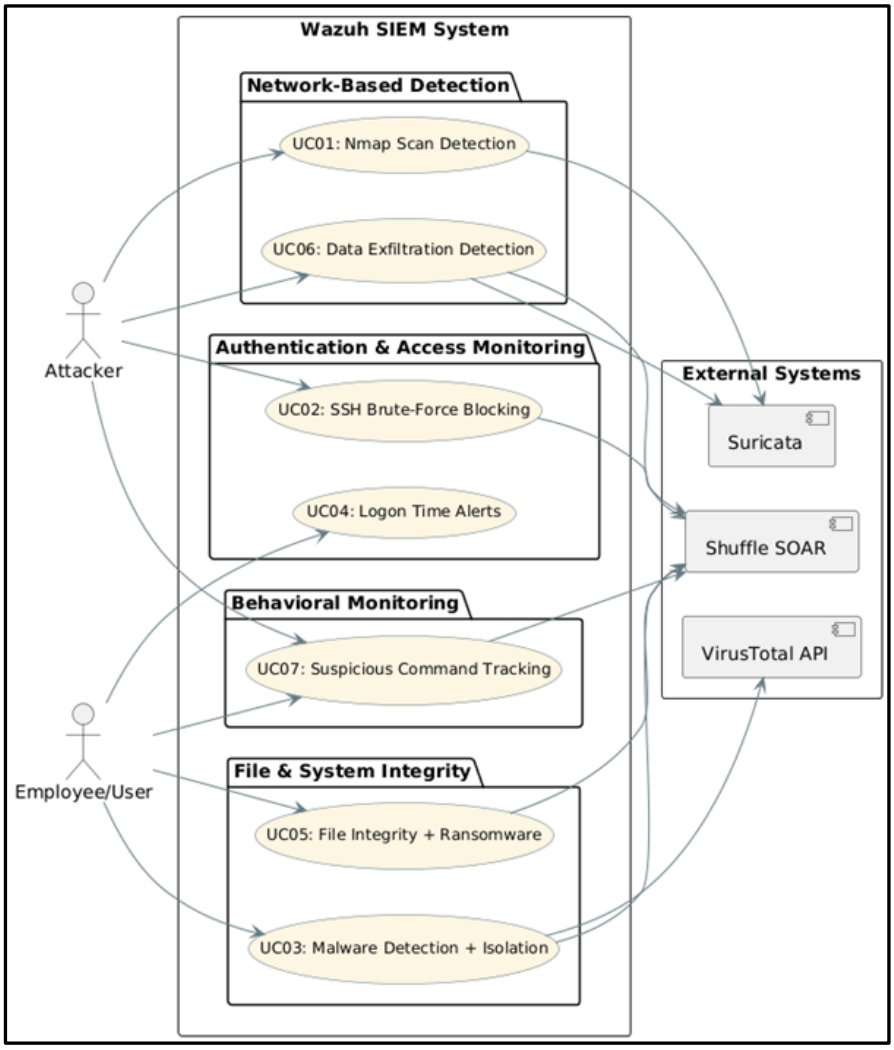


Figure 70: Use case diagram

11 REFERENCES AND BIBLIOGRAPHY

Ali, M.L. et al., 2024. The Rise of Artificial Intelligence: Industry Insights and Applications in Security Information and Event Management (SIEM). In: 2024 IEEE 15th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Yorktown Heights, NY, USA, pp. 477–482. Available at: https://doi.org/10.1109/UEMCON62879.2024.10754705 (Accessed: 9 December 2024).

Amami, R., Charfeddine, M. and Masmoudi, S., 2024. Exploration of Open Source SIEM Tools and Deployment of an Appropriate Wazuh-Based Solution for Strengthening Cyberdefense. In: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT), Vallette, Malta, pp. 1–7. Available at: https://doi.org/10.1109/CoDIT62066.2024.10708476 (Accessed: 9 December 2024).

Bassey, C. et al., 2024. Building a Scalable Security Operations Center: A Focus on Open-source Tools. Journal of Engineering Research and Reports, 26(7), pp. 196–209. Available at: https://doi.org/10.9734/jerr/2024/v26i71203 (Accessed: 10 December 2024).

Brandao, P.R. and Nunes, J., 2021. Extended Detection and Response: Importance of Events Context. Kriativ Tech, 9. Available at: https://doi.org/10.31112/kriativ-tech-2021-10-58 (Accessed: 10 December 2024).

Davies, J., 2019. 206 days: IBM's estimate on how long it takes to find a security breach. Telecoms.com, 23 July. Available at: https://www.telecoms.com/security/206-days-ibm-s-estimate-on-how-long-it-takes-to-find-a-security-breach (Accessed: 11 February 2025).

George, A.S. et al., 2021. XDR: The Evolution of Endpoint Security Solutions - Superior Extensibility and Analytics to Satisfy the Organizational Needs of the Future. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 8(1). Available at: https://doi.org/10.5281/zenodo.7028219 (Accessed: 11 December 2024).

Gupta, R., 2024. Security Monitoring with Wazuh. Packt Publishing. ISBN: 9781837632152.

Kapko, M., 2025. AT&T, Verizon say they evicted Salt Typhoon from their networks. Cybersecurity Dive, 7 January. Available at: https://www.cybersecuritydive.com/news/att-verizon-salt-typhoon/736680/ (Accessed: 11 February 2025).

Kasturi, S. et al., 2024. On the Benefits of Vulnerability Data Consolidation in Application Security. International Conference on Cyber Warfare and Security (ICCWS), 19(1), p. 2086. Available at: https://doi.org/10.34190/iccws.19.1.2086 (Accessed: 12 December 2024).

Linares, J., 2019. Preventing and detecting ransomware with Wazuh. [blog] Wazuh, 12 December. Available at: https://wazuh.com/blog/preventing-and-detecting-ransomware-with-wazuh/ (Accessed: 12 December 2024).

Nguyen, M.-D. et al., 2024. AI4SOAR: A Security Intelligence Tool for Automated Incident Response. In: Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES '24). Association for Computing Machinery, New York, NY, USA, Article 170, pp. 1–8. Available at: https://doi.org/10.1145/3664476.3670450 (Accessed: 13 December 2024).

Podzins, O. and Romanovs, A., 2019. Why SIEM is Irreplaceable in a Secure IT Environment? In: 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, pp. 1–5. Available at: https://doi.org/10.1109/eStream.2019.8732173 (Accessed: 13 December 2024).

Roche, D. and Dowling, S., 2023. Elevating Cybersecurity Posture by Implementing SOAR. In: 2023 Cyber Research Conference - Ireland (Cyber-RCI), Letterkenny, Ireland, pp. 1–7. Available at: https://doi.org/10.1109/Cyber-RCI59474.2023.10671437 (Accessed: 14 December 2024).

Sridharan, A. and Kanchana, V., 2022. SIEM integration with SOAR. In: 2022 International Conference on Futuristic Technologies (INCOFT), Belgaum, India, pp. 1–6. Available at: https://doi.org/10.1109/INCOFT55651.2022.10094537 (Accessed: 14 December 2024).

tech-corewin (2023) *Wazuh Files: SIEM Setup and Configuration for Wazuh*. GitHub repository. Available at: https://github.com/tech-corewin/Wazuh-files/blob/main/README.md (Accessed: 6 April 2025).

Wazuh, n.d. Audit commands run by user. [online] Available at: https://documentation.wazuh.com/current/proof-of-concept-guide/audit-commands-run-by-user.html (Accessed: 10 April 2025).

Wazuh, n.d. Blocking SSH brute-force attacks with Active Response. [online] Available at: https://documentation.wazuh.com/current/user-manual/capabilities/active-response/ar-use-cases/blocking-ssh-brute-force.html (Accessed: 15 March 2025).

Wazuh, n.d. Detect and remove malware using VirusTotal integration. [online] Available at: https://documentation.wazuh.com/current/proof-of-concept-guide/detect-remove-malware-virustotal.html (Accessed: 26 March 2025).

Wazuh, n.d. Integrate Network IDS (Suricata). [online] Available at: https://documentation.wazuh.com/current/proof-of-concept-guide/integrate-network-ids-suricata.html (Accessed: 17 March 2025).

Wazuh, n.d. PoC: File Integrity Monitoring. [online] Available at: https://documentation.wazuh.com/current/proof-of-concept-guide/poc-file-integrity-monitoring.html (Accessed: 12 February 2025).