Integrated Cyber Defense Environment (ICDE)

DISSERTATION REPORT

SUBMITTED TO

SCHOOL OF CYBER SECURITY AND DIGITAL FORENSICS

NATIONAL FORENSIC SCIENCES UNIVERSITY

FOR THE PARTIAL FULFILMENT

FOR THE DEGREE OF

Master of Science

IN

CYBER SECURITY

Submitted By

SUDARSHAN RANGAPPA

012300300008002019

Under The Supervision Of

Dr. Ramya Shah

Assistant Professor

SCSDF, NFSU

MAY, 2025

Integrated Cyber Defense Environment (ICDE)

A M. Sc. DISSERTATION THESIS

SUBMITTED BY

Sudarshan Rangappa
(012300300008002019)

UNDER THE SUPERVISION OF

DR. RAMYA SHAH
ASSISTANT PROFESSOR
SCHOOL OF CYBER SECURITY & DIGITAL FORENSICS

NATIONAL FORENSIC SCIENCES UNIVERSITY,
GANDHINAGAR, GUJARAT, INDIA
MAY, 2025

SCHOOL OF CYBER SECURITY AND DIGITAL FORENSICS

NATIONAL FORENSIC SCIENCES UNIVERSITY

SECTOR 9, GANDHINAGAR, GUJARAT.

CERTIFICATE

This is to certify **SUDARSHAN RANGAPPA** with enrolment number **012300300008002019** who has worked for the dissertation in **MSc. Cyber Security** is a Bonafide student of the School of Cyber Security, National Forensic Sciences University, Gandhinagar, Gujarat during the fourth semester from January 2025 to April 2025.

This dissertation work was carried out by the student under the supervision of Dr. Ramya Shah, Lecturer, School of Cyber Security & Digital Forensics, National Forensic Sciences University, Gandhinagar.

Date:

Place: Gandhinagar, Gujarat

Name & Signature of Supervisor 1	**Name & Signature of Supervisor 2**

Dean
School of Cyber Security and Digital Forensics
National Forensic Sciences University
Gandhinagar, Gujarat.

SCHOOL OF CYBER SECURITY AND DIGITAL FORENSICS
NATIONAL FORENSIC SCIENCES UNIVERSITY
SECTOR 9, GANDHINAGAR, GUJARAT.

CERTIFICATE FROM GUIDE

I hereby certify that the thesis entitled, ***"Integrated Cyber Defense Environment (ICDE)",*** *embodies the result of bonafide research work done by* ***Sudarshan Rangappa*** (***012300300008002019***) *for the degree of Master of Science in Digital Forensics & Information Security, School of Cyber Security & Digital Forensics by National Forensic Sciences University, Gandhinagar under my guidance and supervision. I further certify that this is an original work and that whatever material obtained and used from other sources has been duly acknowledged in the thesis. This work has not been submitted for any degree or diploma of any university or institute.*

Date:
Place: Gandhinagar, Gujarat

Dr. Ramya Shah
Lecturer, School of Cyber Security & Digital Forensics
National Forensic Sciences University,
Gandhinagar, Gujarat – 382007.

DECLARATION

I "**SUDARSHAN RANGAPPA"** having Enrolment Number **"012300300008002019"** hereby declare that

a. The work contained in the dissertation report entitled **"*Integrated Cyber Defense Environment (ICDE)*"** is being submitted in partial fulfilment for the award of the degree of **"MSc. Cybersecurity"** to **School of Cyber Security & Digital Forensics** is an authentic record of my own work done under the supervision of **Dr. Ramya Shah**.

b. The work has not been submitted to any other Institute/ School / University for any degree or diploma.

c. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the School.

d. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

e. Whenever I have quoted written materials from other sources and due credit is given to the sources by citing them.

f. From the plagiarism test, it is found that the similarity index of whole dissertation is less than 10 % as per the university guidelines.

Date:

Place: Gandhinagar, Gujarat

Sudarshan Rangappa
MSc. Cyber Security (SEM IV)
School of Cyber Security & Digital Forensics
NFSU, Gandhinagar
Enrol. No.: 012300300008002019

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to everyone who has contributed to the successful completion of this dissertation report on the "Integrated Cyber Defense Environment (ICDE)."

First and foremost, I extend my deepest appreciation to my supervisor, Dr. Ramya Shah, Assistant Professor at the School of Cyber Security and Digital Forensics, National Forensic Sciences University (NFSU). Her invaluable guidance, unwavering support, and insightful feedback were instrumental throughout this research process. Her expertise and encouragement inspired me to delve deeper into the subject matter and refine my work to its best potential.

I am also thankful to the faculty and staff of the School of Cyber Security and Digital Forensics at NFSU for providing a stimulating academic environment and for their support during my studies. The knowledge and skills I acquired at NFSU have been crucial for this dissertation.

Furthermore, I acknowledge the contributions of my peers and colleagues who offered valuable discussions, shared their insights, and provided moral support during challenging times.

Lastly, I am profoundly grateful to my family for their unwavering support, encouragement, and understanding. Their love and patience have been my anchor throughout this endeavor.

Thank you all for your invaluable contrbutions and support.

Sudarshan Rangappa

Table of Content

ABSTRACT

This project delivers an advanced, virtualized cybersecurity laboratory meticulously architected within a VMware environment to emulate contemporary Security Operations Center (SOC) operations and significantly enhance defensive strategies. The lab's strength lies in its seamless integration of a diverse, yet synergistic, security ecosystem. Wazuh provides comprehensive Security Information and Event Management (SIEM) and host-based intrusion detection (HIDS) capabilities, feeding critical alerts and logs. Replacing traditional SOAR components, **Shuffle** serves as the powerful open-source Security Orchestration, Automation, and Response engine, enabling the creation of customizable, automated workflows for streamlined incident handling and response actions.

Network vigilance is maintained by Suricata (NIDS), while OpenVAS conducts proactive vulnerability assessments to identify weaknesses. Tactical deception is employed using the Cowrie honeypot to gather threat intelligence, and Splunk underpins the architecture with robust log aggregation and deep analytical capabilities. This integrated stack operates cohesively alongside a realistic Active Directory infrastructure and representative target endpoints, ensuring a high-fidelity training milieu.

Key functionalities include centralized log correlation, real-time network and host threat detection, proactive vulnerability management, strategic deception deployment, and crucially, sophisticated incident response orchestration powered by **Shuffle's** adaptable workflow automation. This immersive environment empowers users to cultivate critical hands-on skills in event correlation, alert investigation, vulnerability lifecycle management, and the execution of efficient, automated response sequences defined within **Shuffle**. Simulated attacks, launched via Kali Linux, rigorously validate the integrated security posture, allowing for iterative refinement of detection logic and response playbooks. Ultimately, this cyber range provides an unparalleled platform for developing demonstrable proficiency in modern threat detection, automated incident response, and building robust cyber resilience.

Keywords: *Cybersecurity Laboratory, SOAR (Security Orchestration, Automation, Response), Incident Response, Threat Detection, Virtualization*

LIST OF ABBREVIATIONS

Abbreviation	Description
ICDE	Integrated Cyber Defense Environment
SOC	Security Operations Center
EYFS	Early Years Foundation Stage
SIEM	Security Information and Event Management
SOAR	Security Orchestration, Automation, and Response
NIDS	Network Intrusion Detection System
HIDS	Host-based Intrusion Detection System
APT	Advanced Persistent Threat
AV	Antivirus
VM	Virtual Machine
GVM	Greenbone Vulnerability Management
GSA	Greenbone Security Assistant
NVTs	Network Vulnerability Tests
SCAP	Security Content Automation Protocol
CERT	Computer Emergency Response Team
TTPs	Tactics, Techniques, and Procedures
AD	Active Directory
DNS	Domain Name System
TLS	Transport Layer Security
UUIDs	Universally Unique Identifiers
RPC	Remote Procedure Call
DC	Domain Controller
OU	Organizational Unit
GPO	Group Policy Object
AD DS	Active Directory Domain Services
EPS	Events Per Second

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION AND BACKGROUND STUDY

1.1 Introduction

1.1.1 Overview of Modern Security Operations Centers (SOC)

Modern cybersecurity threats are increasingly sophisticated, persistent, and automated, demanding proactive and intelligent defense mechanisms. Security Operations Centers (SOCs) serve as the central command hub for organizations, tasked with continuously monitoring, detecting, analyzing, and responding to cybersecurity incidents. Effective SOCs rely on a combination of skilled personnel, well-defined processes, and integrated technologies to maintain situational awareness and protect critical assets against a diverse range of threats.

1.1.2 The Need for Integrated Cyber Defense Environments

The effectiveness of a SOC heavily depends on the seamless integration of various security tools. Siloed systems often lead to alert fatigue, delayed responses, and missed correlations. An Integrated Cyber Defense Environment (ICDE) aims to overcome these challenges by creating a cohesive ecosystem where different security technologies (like SIEM, SOAR, NIDS, HIDS, Vulnerability Scanners, Honeypots, and Log Management) work together. Such environments provide a holistic view of the security posture and enable more efficient and effective threat detection and response. Furthermore, realistic simulation environments are crucial for training cybersecurity professionals and validating defense strategies without risking production systems.

1.1.3 Goals and Objectives of the ICDE Project

The primary goal of this project is to design, implement, and validate a comprehensive, virtualized Integrated Cyber Defense Environment (ICDE) that simulates key functions of a modern SOC.

Objectives:

- To establish a scalable virtualized infrastructure using VMware capable of hosting multiple security tools and target systems.
- To deploy and configure a suite of essential open-source and standard security tools: Wazuh (SIEM/HIDS), Shuffle (SOAR), Suricata (NIDS), OpenVAS (Vulnerability Scanner), Cowrie (Honeypot), and Splunk (Log Aggregation/Analysis).
- To integrate these tools to ensure effective data flow (logs, alerts) between components.
- To set up a realistic target environment including an Active Directory domain and Windows/Linux endpoints.
- To configure the Shuffle SOAR platform for alert ingestion and prepare it for workflow automation.
- To demonstrate the ICDE's capabilities in detecting simulated attacks and managing security events.
- To validate the effectiveness of the integrated security stack through controlled attack scenarios executed via Kali Linux.

1.2 Background Study

1.2.1 Existing Cyber Defense Mechanisms and Their Limitations

Traditional cyber defences, such as basic firewalls and signature-based antivirus (AV), proved insufficient against modern threats. Firewalls often lack visibility into encrypted traffic or malicious payloads within allowed protocols, necessitating advanced inspection like that offered by NIDS (**Suricata**). Signature-based AV is ineffective against zero-day exploits, polymorphic malware, or fileless attacks, highlighting the need for behavioral analysis and integrity monitoring provided by HIDS (**Wazuh** agents). Furthermore, early security tools operated in silos, generating numerous uncorrelated alerts (alert fatigue) and hindering the detection of complex attacks like APTs. This lack of integration and context underscored the need for centralized log aggregation (**Splunk**), event correlation (**Wazuh** SIEM), and ultimately, automated orchestration (**Shuffle**) to enable faster, more effective investigation and response.

1.2.2 Evolution of Security Technologies (SIEM, SOAR, NIDS, HIDS, etc.)

To address the limitations of traditional defenses, security technologies have evolved significantly. Security Information and Event Management (SIEM) systems, like Wazuh, emerged to aggregate logs and events from diverse sources (network devices, servers, endpoints, applications) into a central platform. SIEMs provide correlation capabilities, allowing analysts to connect seemingly disparate events, identify patterns indicative of attacks, and generate more context-rich alerts. Network Intrusion Detection Systems (NIDS), such as Suricata, advanced beyond simple signature matching to include protocol analysis and anomaly detection, monitoring network traffic for suspicious activities. Host-based Intrusion Detection Systems (HIDS), often integrated within SIEM agents like Wazuh's, monitor activities directly on endpoints, detecting unauthorized file modifications, rootkit installations, or policy violations. Vulnerability Scanners (e.g., OpenVAS) became crucial for proactively identifying system weaknesses before they could be exploited. Honeypots (e.g., Cowrie) were developed as deception tools to lure attackers, gather intelligence on their methods, and provide early warnings. Most recently, Security Orchestration, Automation, and Response (SOAR) platforms, exemplified by Shuffle in this project, have gained prominence. SOAR aims to connect the entire security stack, automating repetitive tasks, orchestrating complex response workflows (playbooks), and enriching alerts with external threat intelligence, thereby reducing analyst workload, minimizing response times, and improving overall SOC efficiency.

1.2.3 Virtualization in Cybersecurity Training and Simulation

Virtualization technology, particularly hypervisors like VMware, has revolutionized cybersecurity training and research. It allows for the creation of isolated, self-contained virtual networks and systems ('labs' or 'cyber ranges') on shared physical hardware. This offers numerous advantages:

- **Isolation:** Virtual labs can be completely segmented from production networks, allowing students and researchers to safely execute malware, practice penetration testing techniques, and simulate cyber-attacks without risking real-world systems.
- **Scalability & Flexibility:** VMs can be easily created, cloned, modified, and destroyed, allowing for rapid setup and teardown of complex network topologies and diverse operating system environments tailored to specific training scenarios.
- **Repeatability:** Virtual environments can be snapshotted and reverted to known-good states, ensuring consistent and repeatable conditions for training exercises, experiments, and tool validation.

- **Cost-Effectiveness:** Virtualization reduces the need for extensive physical hardware, making sophisticated lab environments more accessible and affordable for educational institutions and organizations. This project leverages VMware to build such an ICDE, providing a practical, hands-on platform for developing skills in deploying, configuring, integrating, and utilizing modern security tools within a simulated SOC context.

1.3 Problem Statement and Research Questions

Many organizations and educational institutions face challenges in setting up realistic, affordable, and comprehensive cybersecurity training and testing environments. Commercial solutions can be expensive, and integrating disparate open-source tools requires significant expertise. This project addresses the need for a well-documented, integrated, and functional virtual lab environment that leverages readily available tools.

Key Questions:

- How can diverse security tools (SIEM, SOAR, NIDS, etc.) be effectively integrated within a virtualized environment to simulate a functional SOC workflow?
- Can Shuffle be successfully integrated as a SOAR platform to ingest alerts from tools like Wazuh within this custom environment?
- How effectively can this integrated environment detect and facilitate the analysis of common cyber-attack scenarios?

1.4 Scope and Limitations of the Project

This project encompasses the design, deployment, configuration, and basic integration of the specified security tools within a VMware environment. It includes setting up the Active Directory domain, endpoints, and the attacker machine. Validation involves demonstrating alert generation, log collection, vulnerability scanning, honeypot interaction, and alert ingestion into the Shuffle platform.

Limitations:

- The project focuses on the foundational setup and integration. The development of complex, fully automated end-to-end response playbooks within Shuffle is beyond the current scope but established as immediate future work.
- The scale of the environment (number of endpoints, complexity of AD) is limited by available hardware resources.
- Performance benchmarking under heavy load is not a primary objective.

2.0 Literature Study of Existing Work

Title	Key Findings	Limitation
Stallings, W. (2020). *Cybersecurity Technologies for Network Defense: SIEM Solutions*	SIEM plays a critical role in real-time threat detection through log aggregation and correlation.	SIEM solutions generate high volumes of alerts, often leading to alert fatigue.
Shackleford, D. (2019). *SOAR: The Future of Automated Incident Response.*	SOAR improves response times by automating incident workflows.	Integration challenges with legacy security infrastructure limit its effectiveness.
Spitzner, L. (2018). *Honeypots: Tracking Hackers.*	Honeypots are effective in gathering intelligence on attacker behavior and tactics.	They require continuous monitoring and maintenance to avoid detection by attackers.
Paxson, V. (2021). *Network Monitoring with Zeek: A Deep Dive into Intrusion Detection*	Zeek provides deep packet inspection and enriches threat intelligence.	High-performance overhead and storage requirements for large network environments.
Herzog, A. (2022). *OpenVAS and Vulnerability Management in Modern Cybersecurity Frameworks.*	OpenVAS efficiently scans for vulnerabilities and provides remediation suggestions.	High false-positive rates lead to unnecessary resource allocation.

Table 1 Literature review

3.0 Methodology and Implementation

3.1 System Architecture and Design of the ICDE

3.1.1 Network Topology and Virtual Machine Setup (VMware)

The ICDE is hosted within a VMware Workstation environment. All virtual machines are connected to a custom VMware network named AD-project (NAT). This network utilizes the 192.168.33.0/24 IP address range, with the default gateway configured at 192.168.33.2. This setup facilitates communication between the VMs while providing internet connectivity via NAT for necessary updates and feed synchronization.

Role / Primary Tool	Operating System	IP Address	RAM	vCPUs	Hard Disk	Notes
Attacker Machine	Kali Linux	192.168.33.143/24	4gb	2	80gb	Used for launching simulated attacks.
Log Aggregation / Analysis	Ubuntu Server 22.04 LTS	192.168.33.128/24	4gb	2	80gb	Hosts Splunk Enterprise (Free License).
Domain Controller / AD	Windows Server 2025	192.168.33.129/24	4gb	4	60gb	Hosts Active Directory Domain Services.
Target Endpoint	Windows 11	192.168.33.130/24	4gb	2	80gb	Represents a typical user workstation.
SIEM / HIDS Manager	Ubuntu Server 22.04 LTS	192.168.33.131/24	4gb	2	80gb	Hosts Wazuh Manager.
SOAR Platform	Ubuntu Server 22.04 LTS	192.168.33.149/24	8gb	2	60gb	Hosts Shuffle.
Vulnerability Scanner	Debian 12	192.168.33.147/24	8gb	4	60gb	Hosts OpenVAS (GVM).
Honeypot	Debian 12	192.168.33.142/24	4gb	2	60gb	Hosts Cowrie.
NIDS Sensor	Debian 12	192.168.33.144/24	2gb	2	60gb	Hosts Suricata. (May need promiscuous mode)

Table 2 Network topologies and configuration of each virtual machines

3.1.2 ICDE Component Roles and Integration Strategy

The ICDE operates through the coordinated interaction of its various components, each fulfilling a specific role in the simulated SOC workflow. Data flows primarily through log forwarding (Syslog) and API interactions:

- **Kali Linux (Attacker Machine):** This VM serves as the launch point for simulated attacks, penetration testing activities, and network scanning against the target environment (AD Server, Windows 11 Workstation). Its actions generate network traffic and endpoint events that the defensive tools aim to detect.

- **Windows Server 2025 (Active Directory Domain Controller):** Acts as the central authentication and management server for the simulated domain. It is a critical target for attacks (e.g., credential theft, privilege escalation). It generates Windows Event Logs detailing authentication successes/failures, policy changes, etc., which are forwarded via the Wazuh agent to the Wazuh Manager and optionally directly to Splunk.

- **Windows 11 (Target Endpoint):** Represents a typical user workstation within the domain. It is a primary target for simulated malware execution, phishing attempts, and lateral movement originating from Kali or potentially compromised accounts. It runs a Wazuh agent that monitors host activity (file changes, process execution, registry modifications) and forwards logs/alerts to the Wazuh Manager. Its Windows Event Logs can also be sent to Splunk.

- **Wazuh (SIEM / HIDS Manager):** This Ubuntu server hosts the Wazuh Manager, which collects and analyzes logs from the Wazuh agents deployed on the Windows Server and Windows 11 endpoints. It performs log analysis, file integrity monitoring, configuration assessment, and generates alerts based on its ruleset. These alerts and relevant logs are forwarded via syslog to Splunk for long-term storage and broader correlation. Crucially, its API is configured to allow the Shuffle SOAR platform to query for new alerts, initiating potential investigation or response workflows.

- **Suricata (NIDS Sensor):** This Debian VM monitors network traffic flowing through the icde-project(NAT) network (requires appropriate network configuration like a SPAN port or promiscuous mode on its interface). It uses rulesets (e.g., ET Open) to detect suspicious network activity, intrusions, and potential malware communications. Detected events are logged in Eve JSON format and forwarded via syslog to Splunk and potentially Wazuh for correlation.

- **Cowrie (Honeypot):** This Debian VM acts as an SSH honeypot, mimicking a vulnerable system to attract and log attacker interactions (login attempts, commands executed). These logs provide valuable intelligence on attacker TTPs (Tactics, Techniques, and Procedures) and are forwarded via syslog to Splunk and potentially Wazuh.

- **OpenVAS (Vulnerability Scanner):** Hosted on Debian, OpenVAS is used to proactively scan the target systems (Windows Server, Windows 11, potentially other servers) for known vulnerabilities. Scan results are managed within the OpenVAS Greenbone Vulnerability Management (GVM) interface and are typically reviewed manually by the analyst in this phase. While not directly integrated for automated log flow in this iteration, reports can be exported.

- **Splunk (Log Aggregation / Analysis):** This Ubuntu server runs Splunk Enterprise (Free License) and serves as the central repository for logs from multiple sources (Wazuh, Suricata, Cowrie, Windows Event Logs). It receives data via syslog over configured TCP/UDP ports, indexes it, and provides a powerful search interface and visualization capabilities for the analyst to perform deep-dive investigations and correlate events across the entire environment. Its API allows Shuffle to perform enrichment lookups.

- **Shuffle (SOAR Platform):** Running on Ubuntu Server, Shuffle acts as the orchestration hub. It connects to the Wazuh API to ingest security alerts. Based on these triggers, it can (or will, in future work) execute predefined workflows. These workflows can involve querying Splunk or Wazuh for additional context (enrichment), interacting with external services (like sending notifications via the Telegram Bot API), and potentially executing response actions (like running SSH commands or triggering API calls on other systems) once fully implemented. The analyst interacts with Shuffle to view ingested alerts, monitor workflow execution (when implemented), and manage cases.

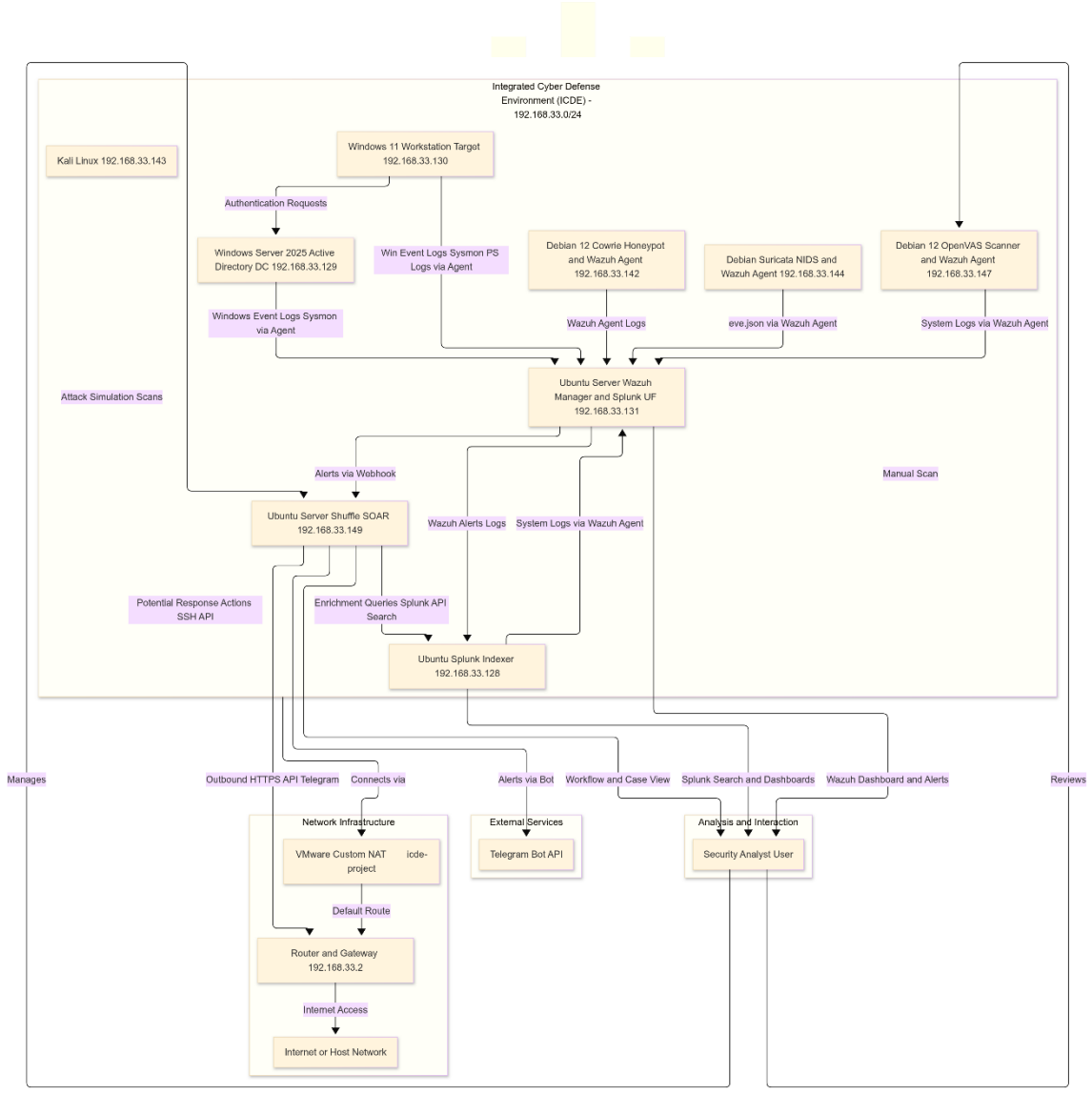

Figure 1 Planned workflow of the whole project

3.2 Configuration and Deployment of Components

The Wazuh platform, comprising the Wazuh server, indexer, and dashboard, was installed on the designated Ubuntu Server 22.04 LTS VM (192.168.33.131).

3.2.1 Wazuh Manager Installation:

The installation followed the official Wazuh documentation, specifically utilizing the "assisted installation" method outlined in the Quickstart guide. This involved downloading and executing the provided installation script, which handles the deployment of all necessary Wazuh components.

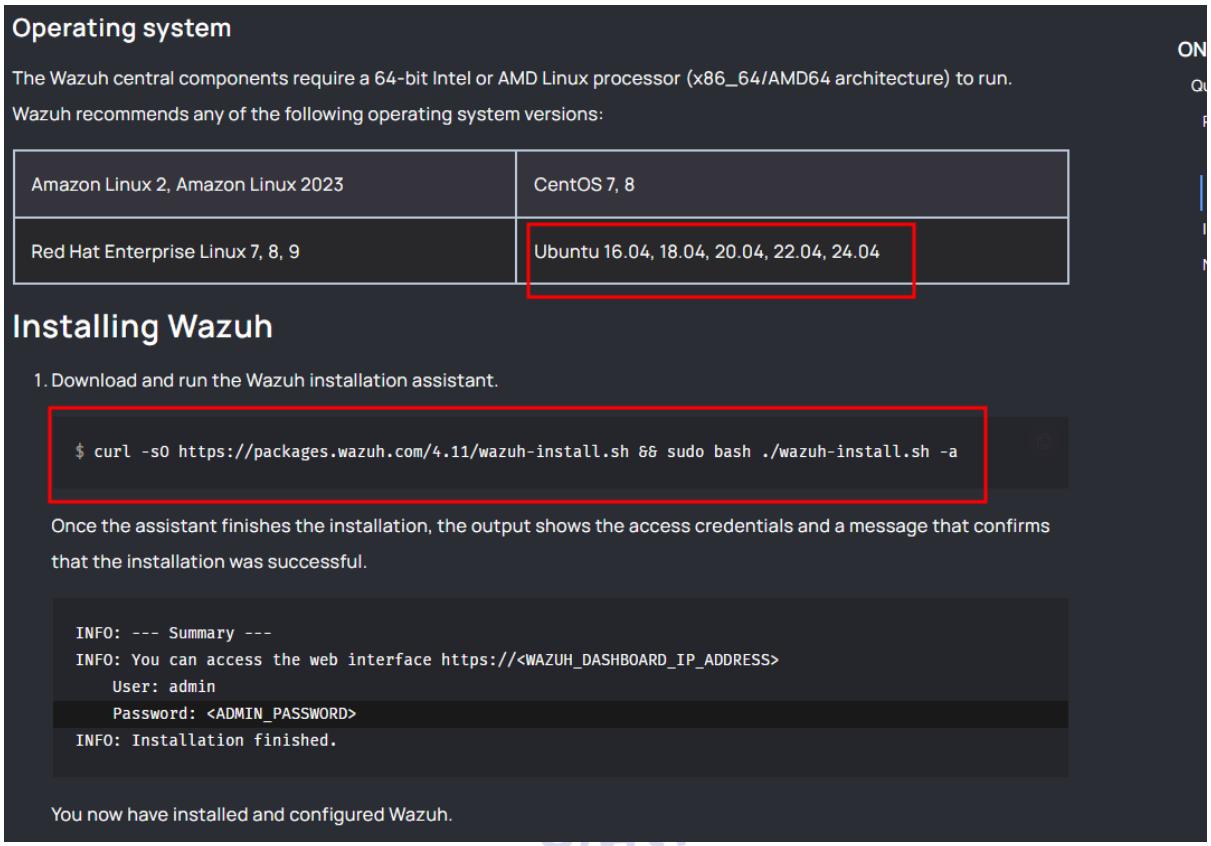

Figure 2 Wazuh Quickstart Documentation Reference

Figure 3 Assisted Installation Command Example

- Upon successful completion of the script, the Wazuh dashboard became accessible via HTTPS (https://192.168.33.131).

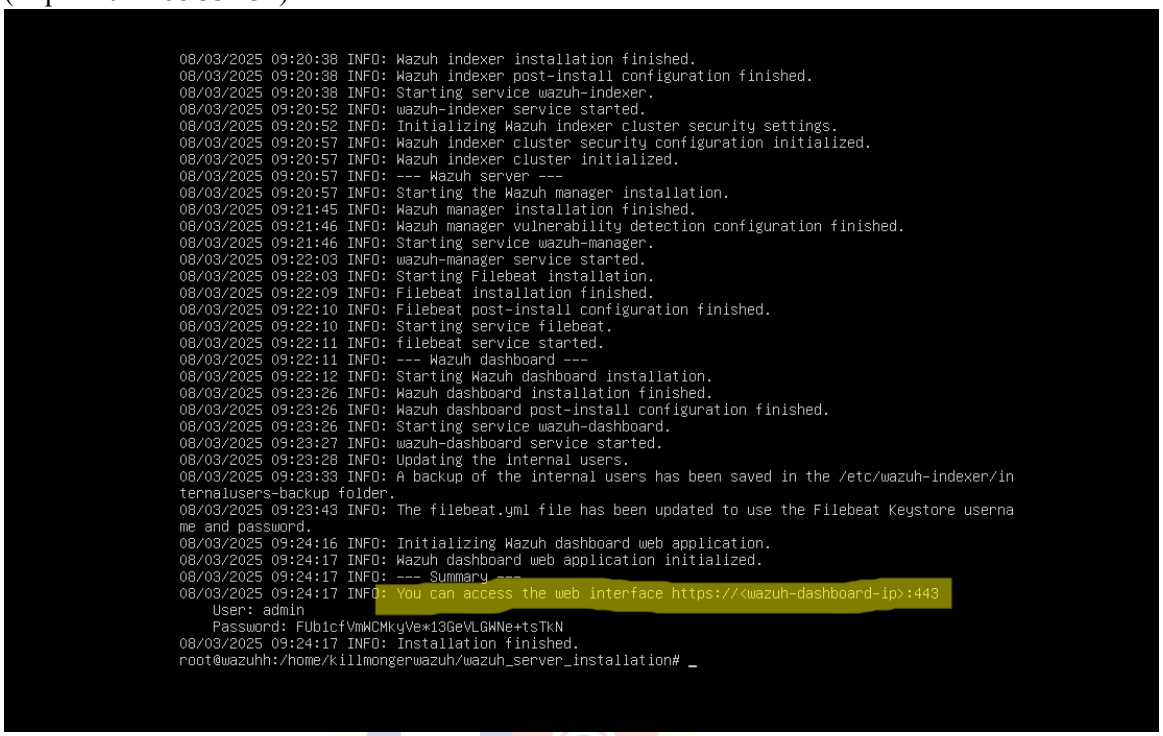

Figure 4 Wazuh Installation Success Message

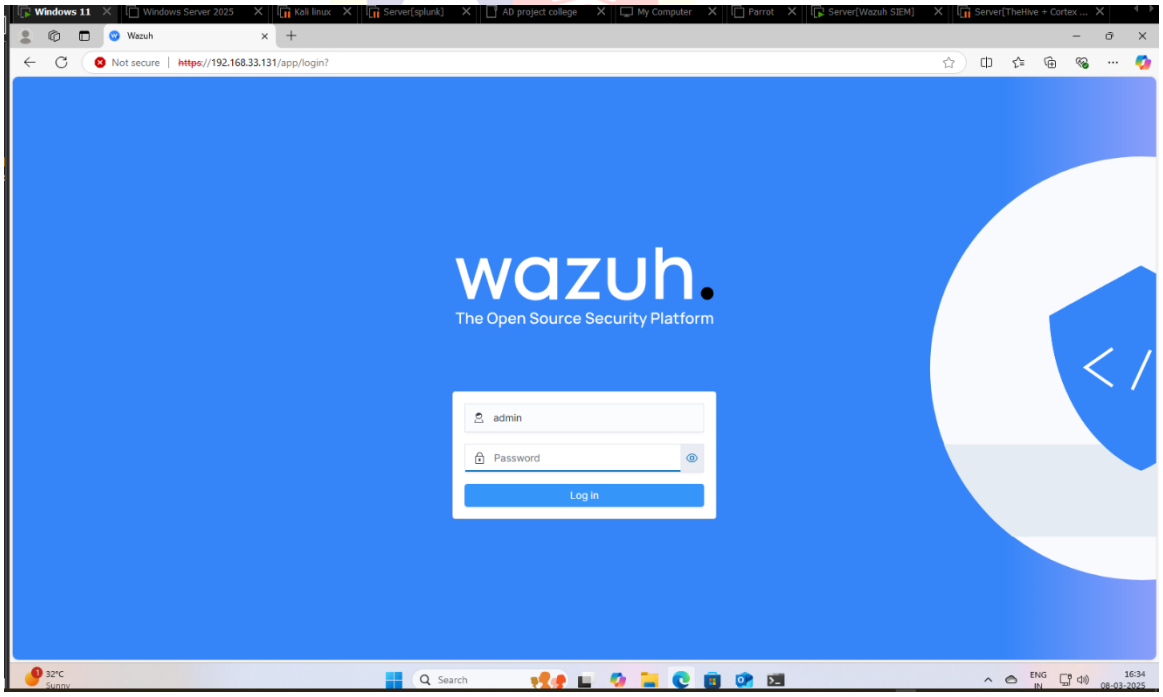

Figure 5 Wazuh Dashboard Login Page

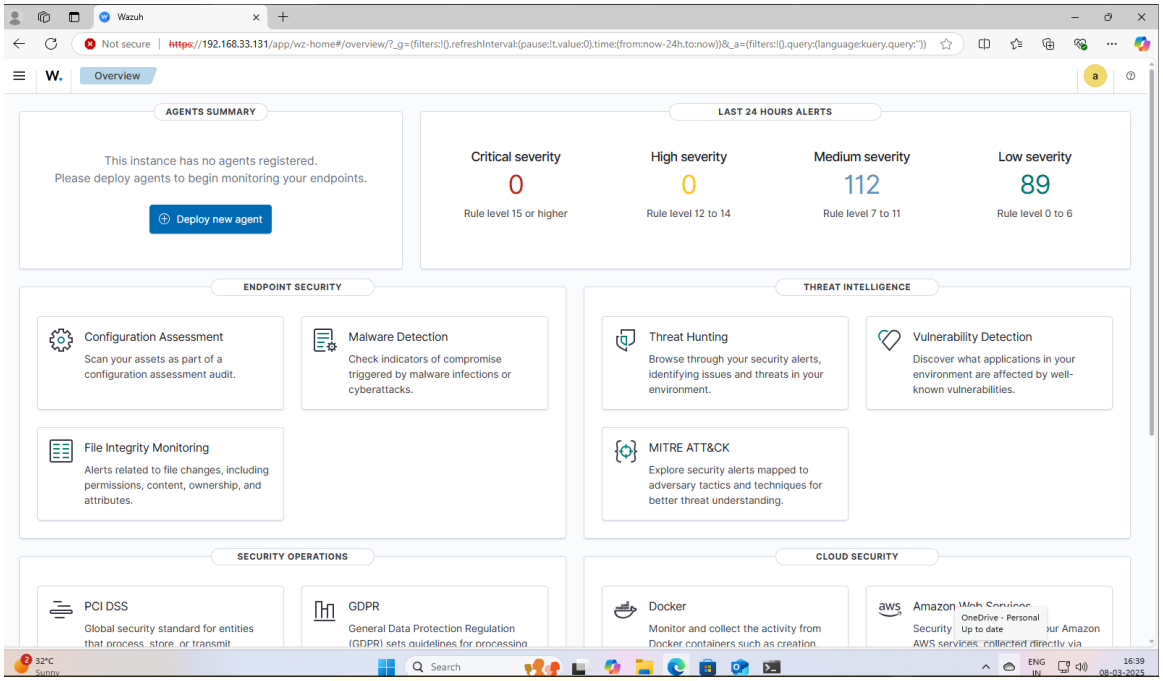

Figure 6 Wazuh Dashboard Main View after Login

3.2.2 Wazuh Agent Deployment

Wazuh agents were deployed to monitor the target Windows systems and the Linux servers hosting other security tools.

Windows Agents (Windows 11 & Windows Server 2025)

The deployment process utilized the "Deploy new agent" feature within the Wazuh dashboard. The appropriate operating system (Windows) was selected, and the dashboard provided PowerShell commands specific to the Wazuh manager's configuration. These commands were executed in an administrative PowerShell prompt on the respective Windows machines (192.168.33.130 and 192.168.33.129). This process downloaded the agent, configured it with the manager's IP address, and registered the agent.

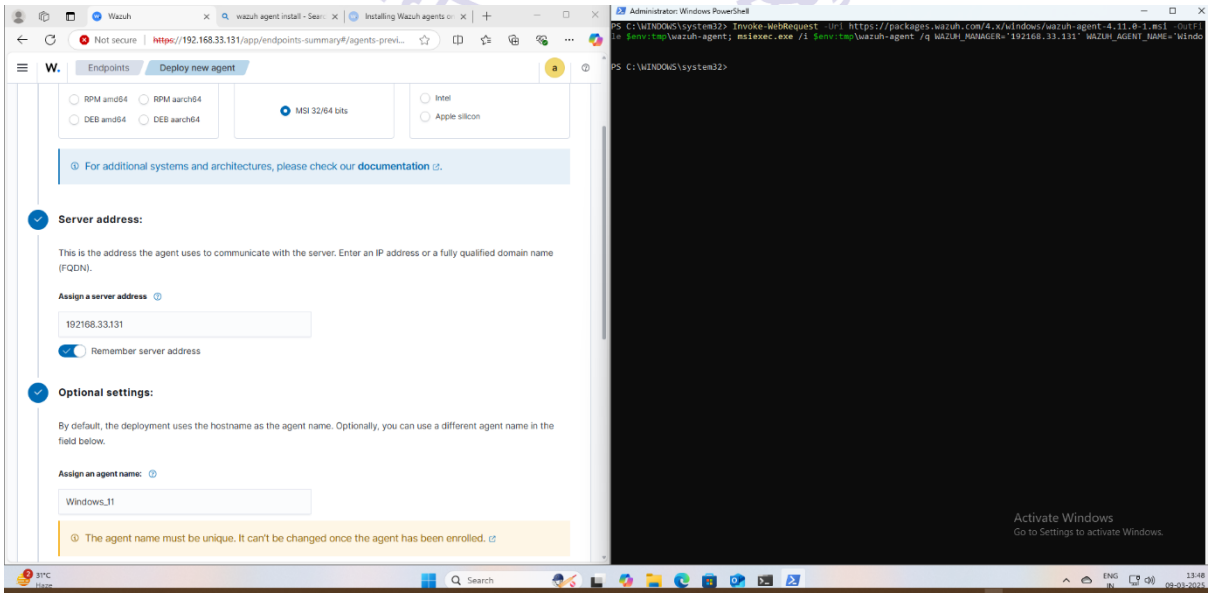

Figure 7 Wazuh Deploy Agent UI - Windows Selection

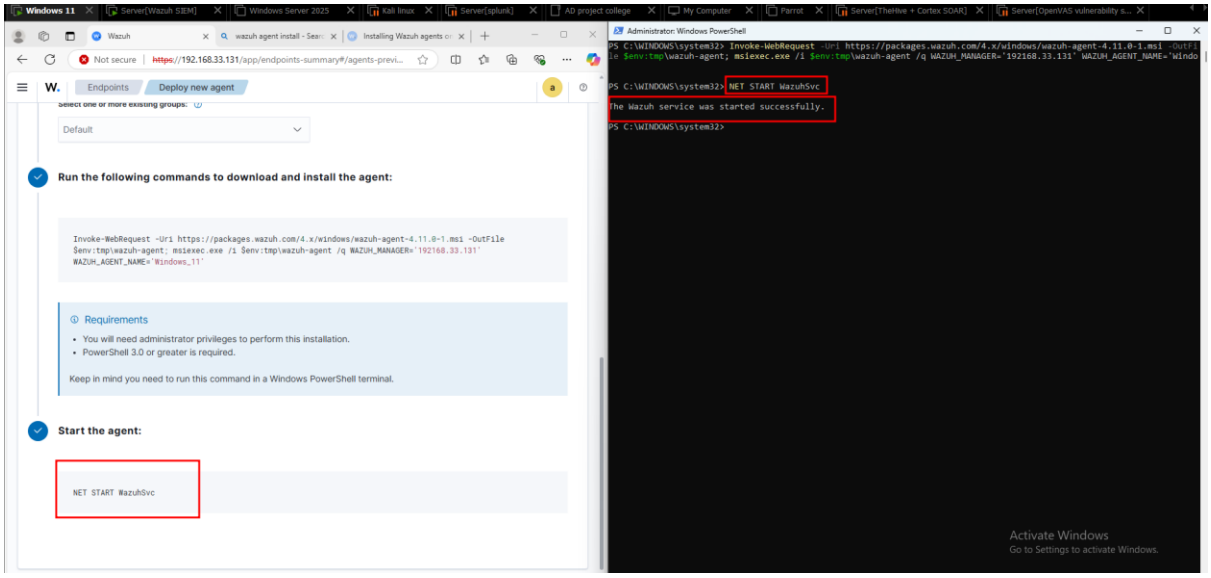

Figure 8 Example PowerShell Command for Windows Agent

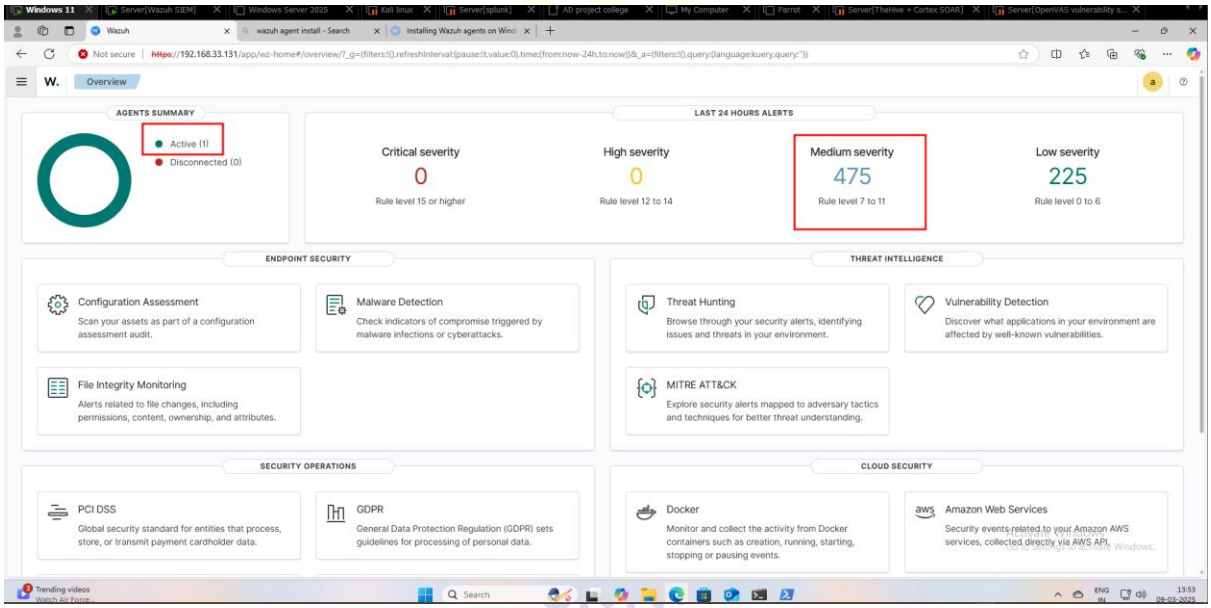

Figure 9 Successful Agent Installation Message on Windows

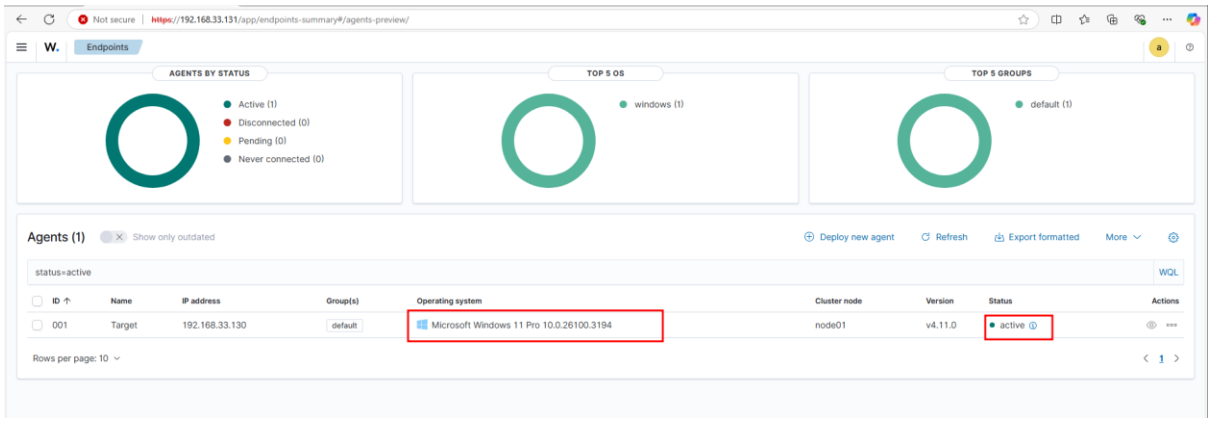

Figure 10 Windows 11 successfully added

Follow the same steps for Windows server 2025

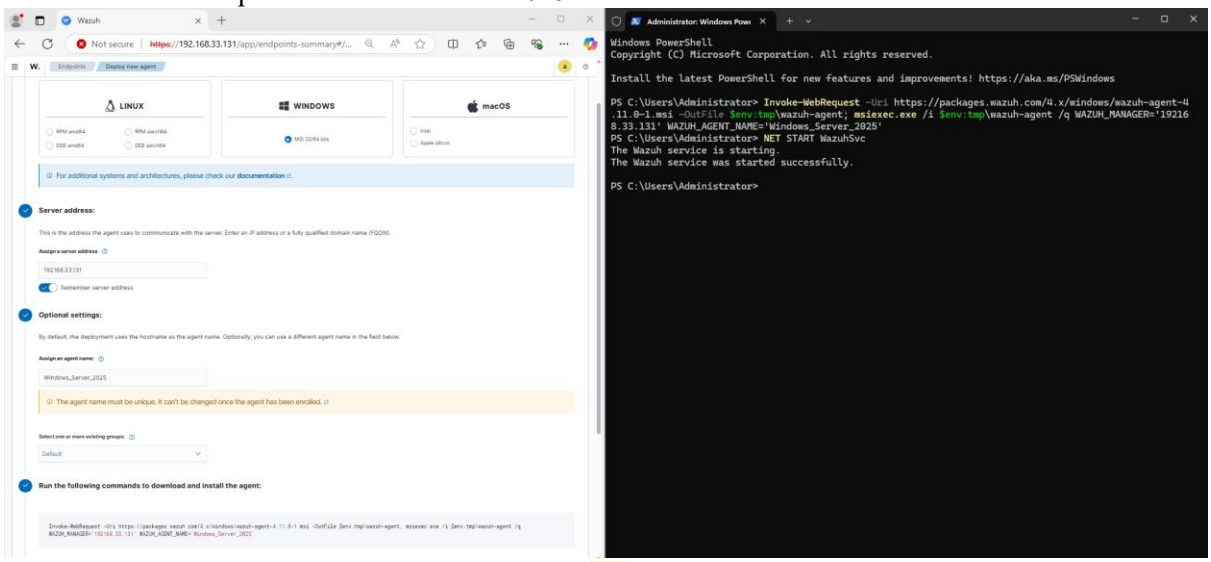

Figure 11 Following all commands to install agent on windows server 2025

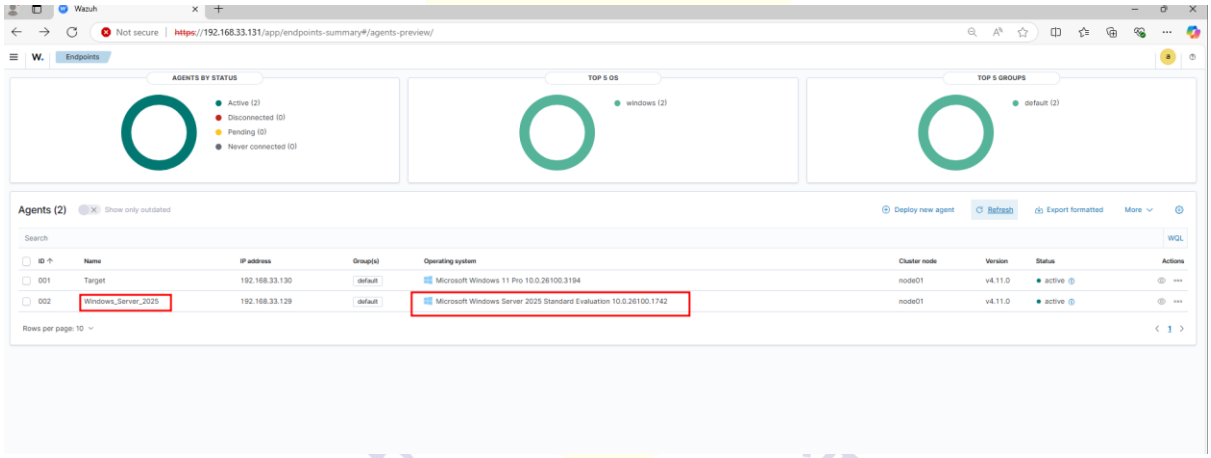

Figure 12 windows server 2025 also added and showing up in wazuh dashboard

Linux Agents (Suricata, Cowrie, Splunk, OpenVAS Servers)

A similar process was followed for the Debian and Ubuntu servers. The "Deploy new agent" feature was used, selecting the appropriate Linux distribution. The provided command-line instructions were executed on each respective server (192.168.33.144, 192.168.33.142, 192.168.33.128, 192.168.33.147). This registered the agents with the manager.

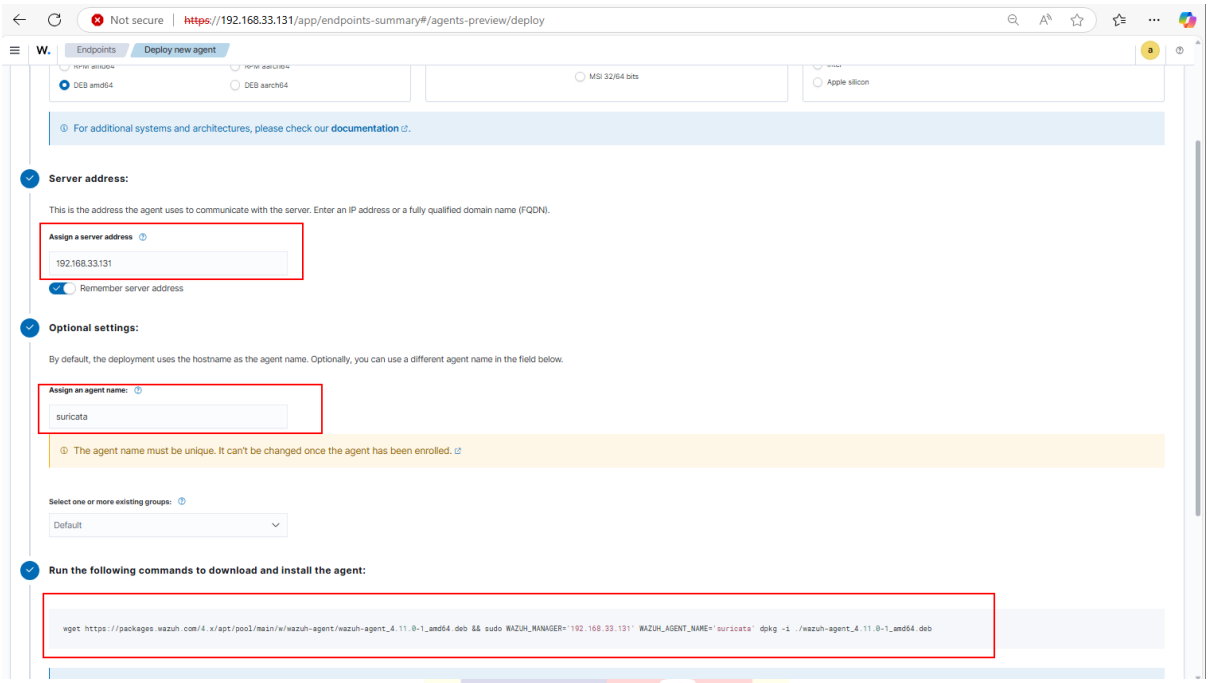

Figure 13 setting up and taking commands for suricata agent installation

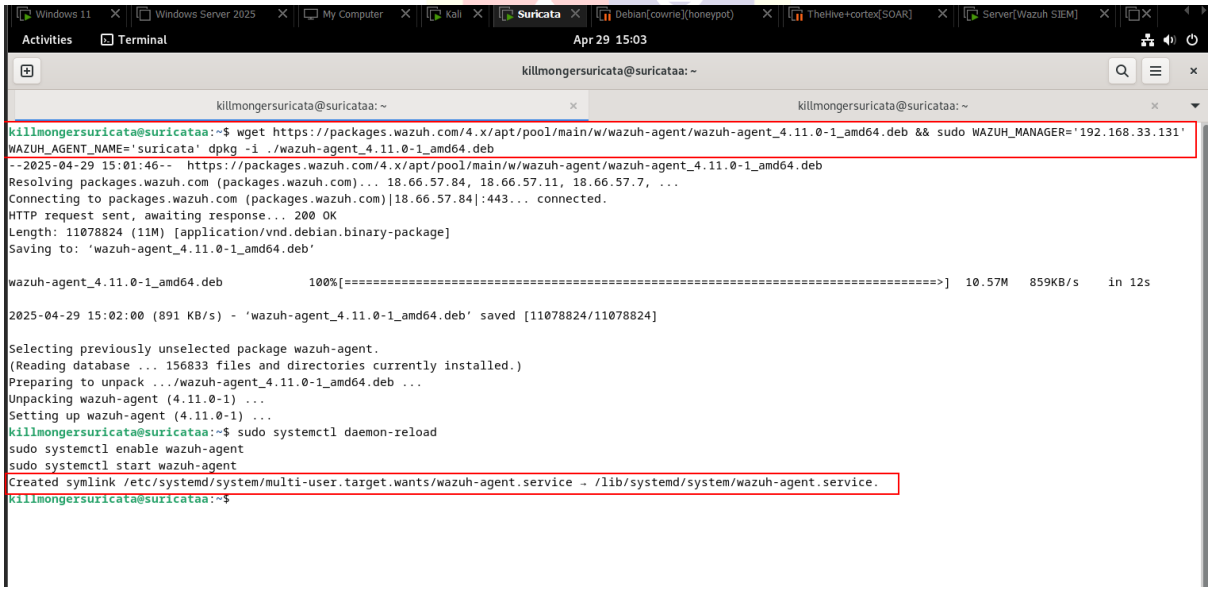

Figure 14 running and successfully installing the wazuh agent on suricata

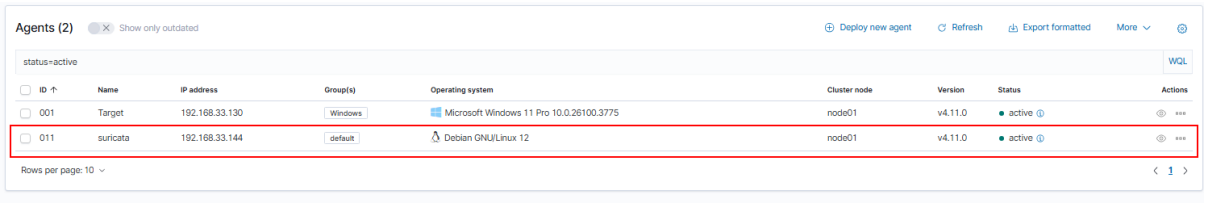

Figure 15 suricata agent showing up in wazuh dashboard

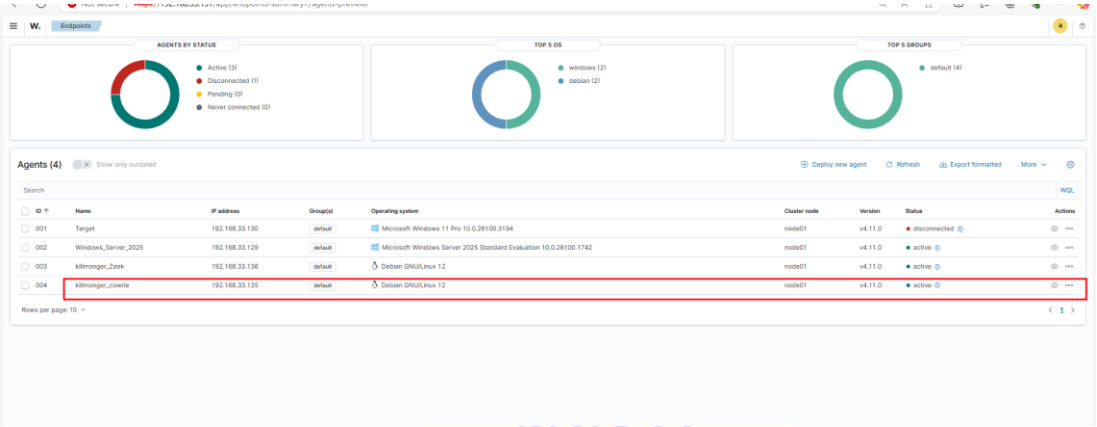

Figure 16 following same commands and successfully installing agent on cowrie

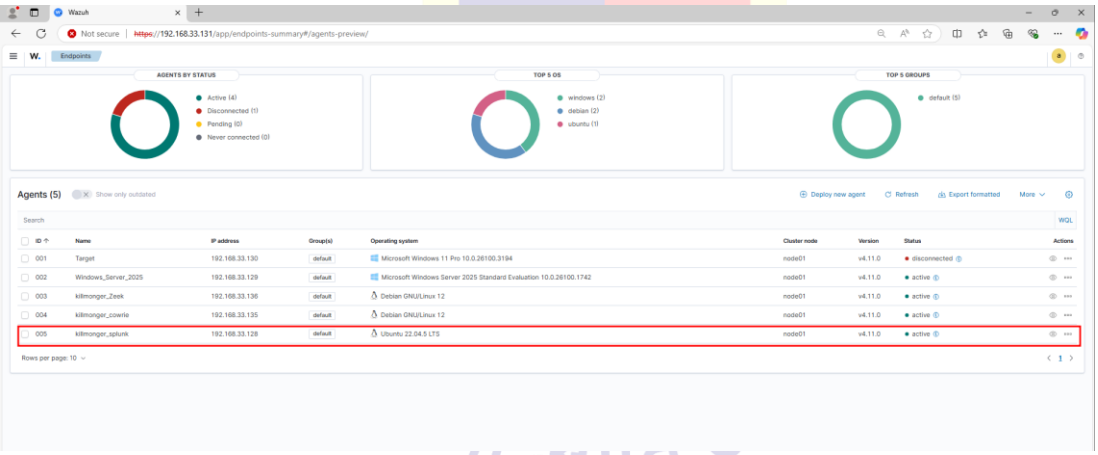

Figure 17 following same commands and successfully installing agent on splunk server

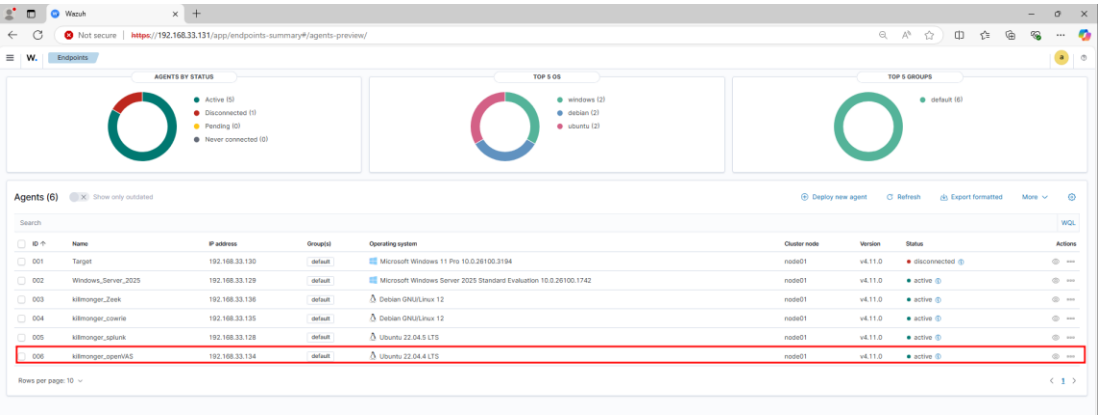

Figure 18 following same commands and successfully installing agent on openvas

Key Configurations:

Log Forwarding to Splunk via Universal Forwarder: To centralize logs, a Splunk Universal Forwarder was installed on the Wazuh Manager VM (192.168.33.131). The forwarder was configured to send data to the Splunk Indexer (192.168.33.128) on its receiving port (default 9997). An inputs.conf file was created or modified within the forwarder's configuration directory (e.g., /opt/splunkforwarder/etc/system/local/) to monitor the Wazuh alerts file (/var/ossec/logs/alerts/alerts.json). A specific sourcetype (e.g., wazuh:json) was assigned for proper parsing in Splunk. The forwarder service was then started and enabled.

- The Splunk Universal Forwarder was installed using the downloaded **.deb** package via the command: **dpkg -i splunkforwarder-<version>-<arch>.deb**
- After it we can configure the inputs.conf file & outputs.conf file

```
killmongerwazuh@wazuhh:~$ cat /opt/splunkforwarder/etc/system/local/inputs.conf
[monitor:///var/ossec/logs/alerts/alerts.json]
disabled = false
index = wazuh_forwarded
sourcetype = wazuh_alerts
# Add more sourcetype stanzas if needed for other files, e.g.:
# [monitor:///var/ossec/logs/ossec.log]
# disabled = false
# index = wazuh_forwarded
# sourcetype = wazuh_ossec
```

Figure 19 Showing the inputs.conf stanza monitoring alerts.json on the Wazuh Manager VM

```
killmongerwazuh@wazuhh:~$ sudo cat /opt/splunkforwarder/etc/system/local/outputs.conf
[sudo] password for killmongerwazuh:
[tcpout]
defaultGroup = default-autolb-group

[tcpout:default-autolb-group]
server = 192.168.33.128:9997

[tcpout-server://192.168.33.128:9997]
killmongerwazuh@wazuhh:~$
```

Figure 20 Showing the outputs.conf stanza pointing to the Splunk Indexer on the Wazuh Manager VM

3.2.3 Setting up Suricata for Network Intrusion Detection

Suricata was installed and configured on the dedicated Debian VM (192.168.33.144) to monitor network traffic within the ICDE.

- **Installation:** Suricata was installed using the standard Debian package manager (apt).

sudo apt update

sudo apt install suricata -y

Network Interface Configuration:

The primary configuration file, /etc/suricata/suricata.yaml, was edited to specify the network interface connected to the icde-project(NAT) network that Suricata should monitor. The af-packet section was configured, setting the interface parameter to the correct interface name (e.g., eth0, ens192). The interface might need to be put into promiscuous mode (sudo ip link set <interface_name> promisc on) to capture all traffic on the network segment, not just traffic destined for the Suricata VM itself.

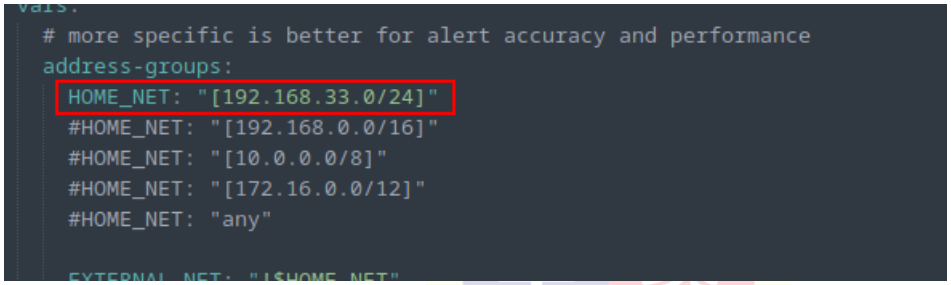

Figure 21 suricata.yaml [configuring network ip]

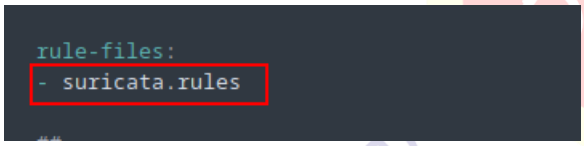

Figure 22 suricata.yaml [setting rules on which suricata works on]

Logging Configuration (Eve JSON): Suricata was configured to output alerts and other events in the comprehensive Eve JSON format, which is ideal for log shippers and SIEMs. In /etc/suricata/suricata.yaml, the eve-log output under the outputs section was enabled, ensuring enabled: yes. The types of events to log (e.g., alert, http, dns, tls, flow) were reviewed and kept at defaults or adjusted as needed.

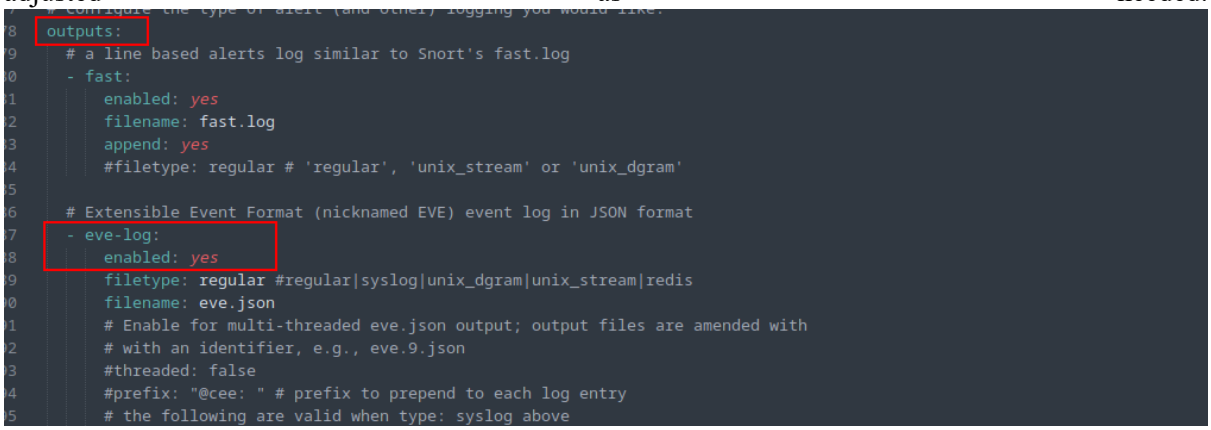

Figure 23 suricata.yaml [configuring suricata to generate event logs in json format "eve.json"]

```
# 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
    # This is only supported for Linux kernel > 3.1
```

Figure 24 suricata.yaml [configuring interface to current system using interface]

- **Service Management:** Finally, the Suricata service was enabled to start on boot and restarted to apply all configuration changes.

```
killmongersuricata@suricataa:~$ sudo systemctl enable suricata
Synchronizing state of suricata.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable suricata
killmongersuricata@suricataa:~$ sudo systemctl restart suricata
killmongersuricata@suricataa:~$ sudo systemctl status suricata
● suricata.service - Suricata IDS/IDP daemon
     Loaded: loaded (/lib/systemd/system/suricata.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-05-02 21:24:22 IST; 25s ago
       Docs: man:suricata(8)
             man:suricatasc(8)
             https://suricata-ids.org/docs/
    Process: 37167 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid (code=exited, status=0/SUCCESS)
   Main PID: 37169 (Suricata-Main)
      Tasks: 8 (limit: 2241)
     Memory: 596.4M
        CPU: 21.484s
     CGroup: /system.slice/suricata.service
             └─37169 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

May 02 21:24:22 suricataa systemd[1]: Starting suricata.service - Suricata IDS/IDP daemon...
May 02 21:24:22 suricataa suricata[37167]: 2/5/2025 -- 21:24:22 - <Notice> - This is Suricata version 6.0.10 RELEASE running in SYSTEM mode
May 02 21:24:22 suricataa systemd[1]: Started suricata.service - Suricata IDS/IDP daemon.
killmongersuricata@suricataa:~$
```

Figure 25 Checking the status of the Suricata service using systemctl status suricata

3.2.4 Configuring OpenVAS for Vulnerability Scanning

OpenVAS (Greenbone Vulnerability Management - GVM) was deployed using the official Greenbone Community Edition Virtual Appliance (OVA) file for VMware Workstation to ensure stability and simplify setup on the Debian VM (192.168.33.147).

Installation via OVA:

- The appropriate Greenbone Community Edition OVA file for VMware was downloaded from the official source.

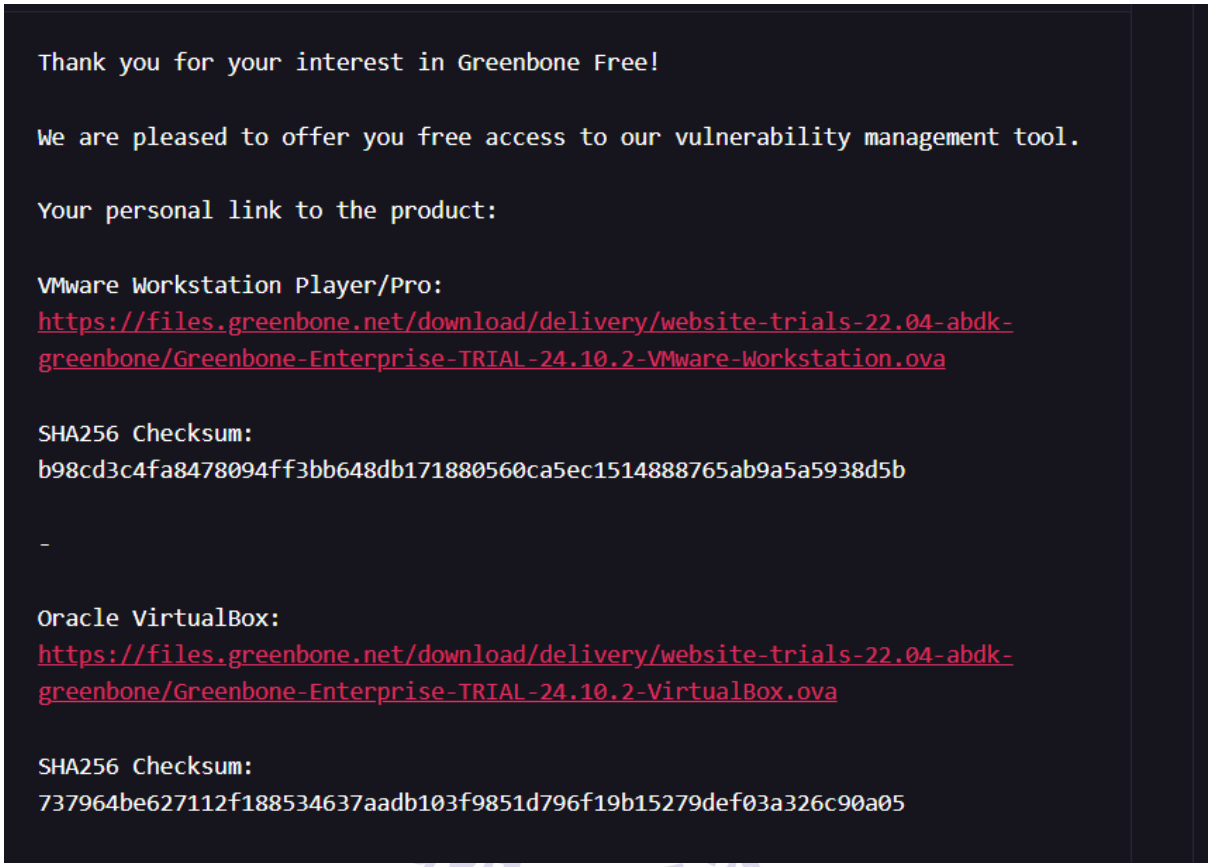

Figure 26 Greenbone Community Edition download page showing VMware OVA option

- The downloaded .ova file was imported into the VMware environment.
- Upon first boot, the VM required initial login using the default credentials (admin/admin).

- A setup wizard guided the initial configuration.

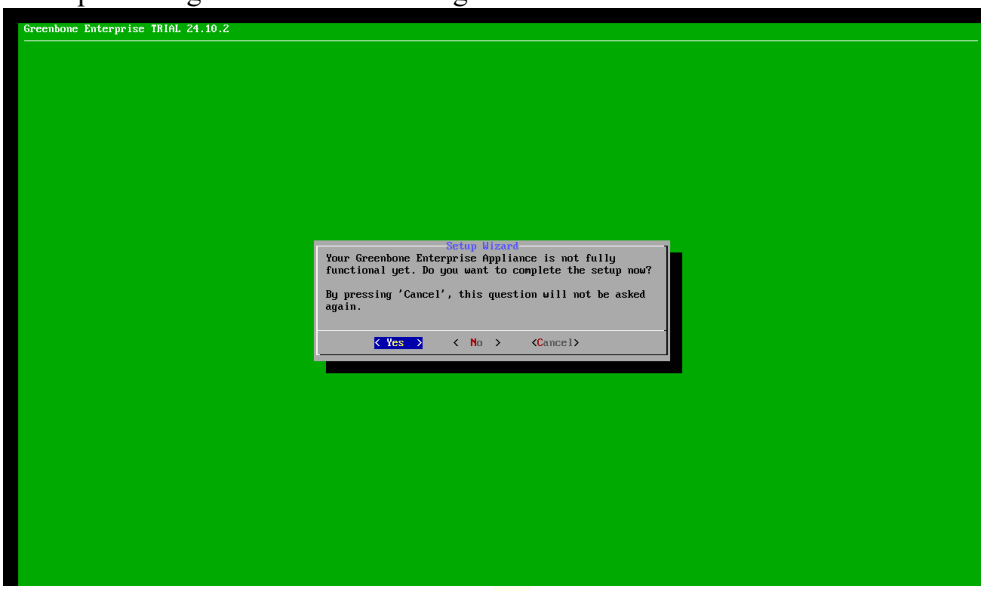

Figure 27 OpenVAS VM console showing the setup wizard prompt

- A dedicated administrator user account was created for accessing the Greenbone Security Assistant (GSA) web interface.

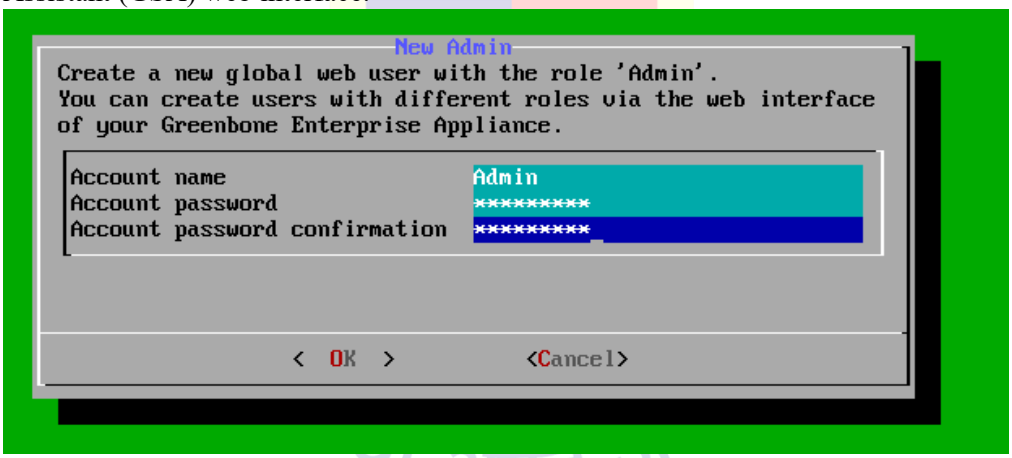

Figure 28 Setup wizard step for creating the GSA admin user

- After successful setup, the GSA web interface became accessible via HTTPS from other machines on the network (https://192.168.33.147).

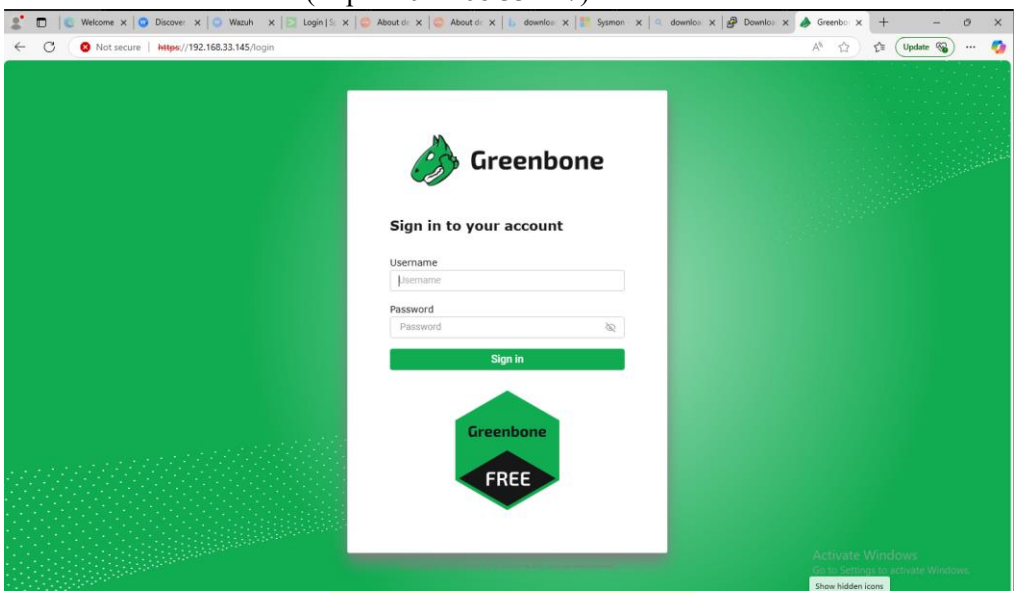

Figure 29 GSA Login Page

- After accessing credentials we can access the dashboard.

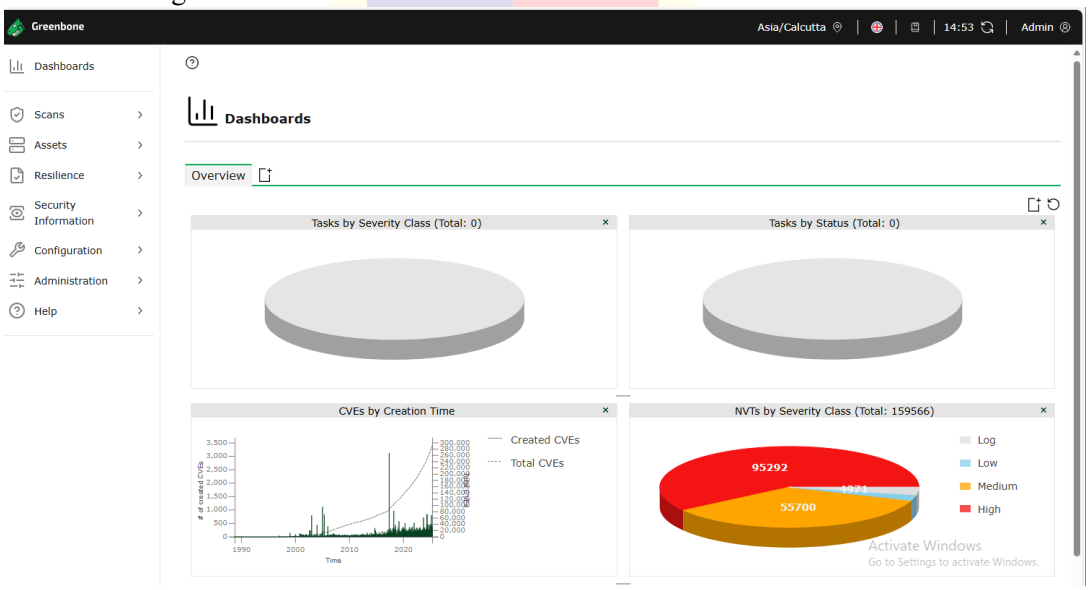

Figure 30 OpenVAS web dashboard

- **Feed Synchronization:** Following installation, the vulnerability feeds (NVTs, SCAP data, CERT data) needed to be synchronized. This process typically starts automatically after setup and can take a significant amount of time (30-60 minutes or more depending on network speed). The status can be monitored via the GSA web interface or potentially the VM console. Synchronization is complete when the feed status shows as "Current".

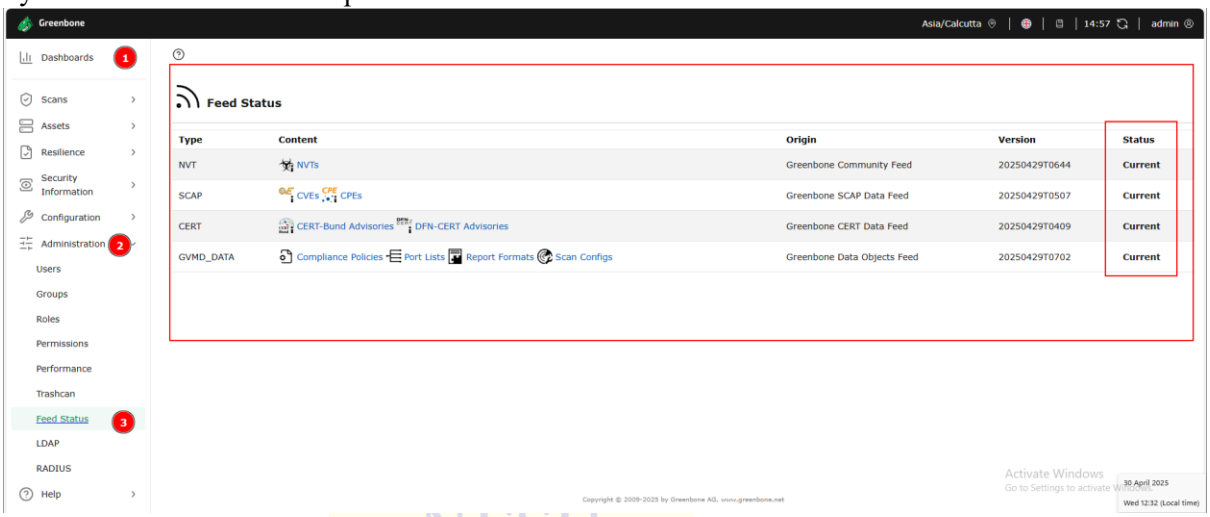

Figure 31 GSA Feed Status page showing synchronization in progress or current

- **Target Configuration:** Target hosts within the ICDE network were defined for scanning. This involved navigating to Configuration > Targets in the GSA interface and creating new targets, specifying their IP addresses (e.g., 192.168.33.129 for the AD Server, 192.168.33.130 for the Windows 11 workstation).

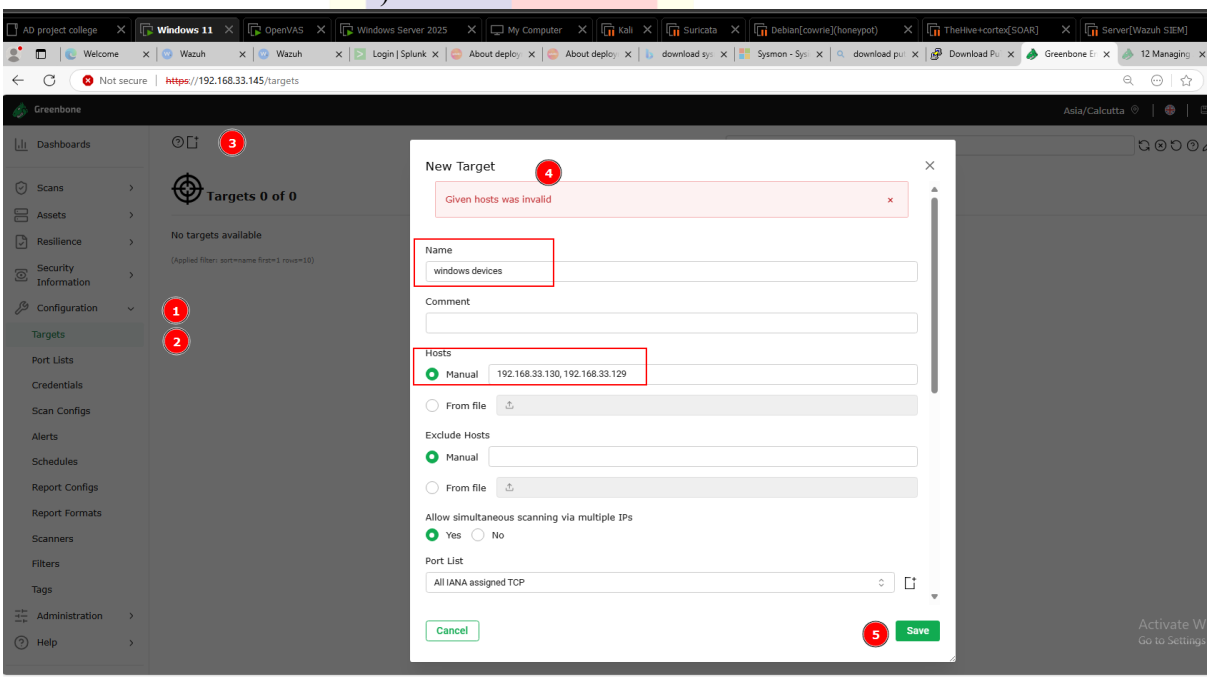

Figure 32 GSA Targets configuration page showing added target hosts

- **Scan Execution:** Basic vulnerability scans were initiated against the configured targets. This involved navigating to Scans > Tasks, creating a new task, selecting the target(s), and choosing a scan configuration (e.g., "Full and fast"). The scan was then started.

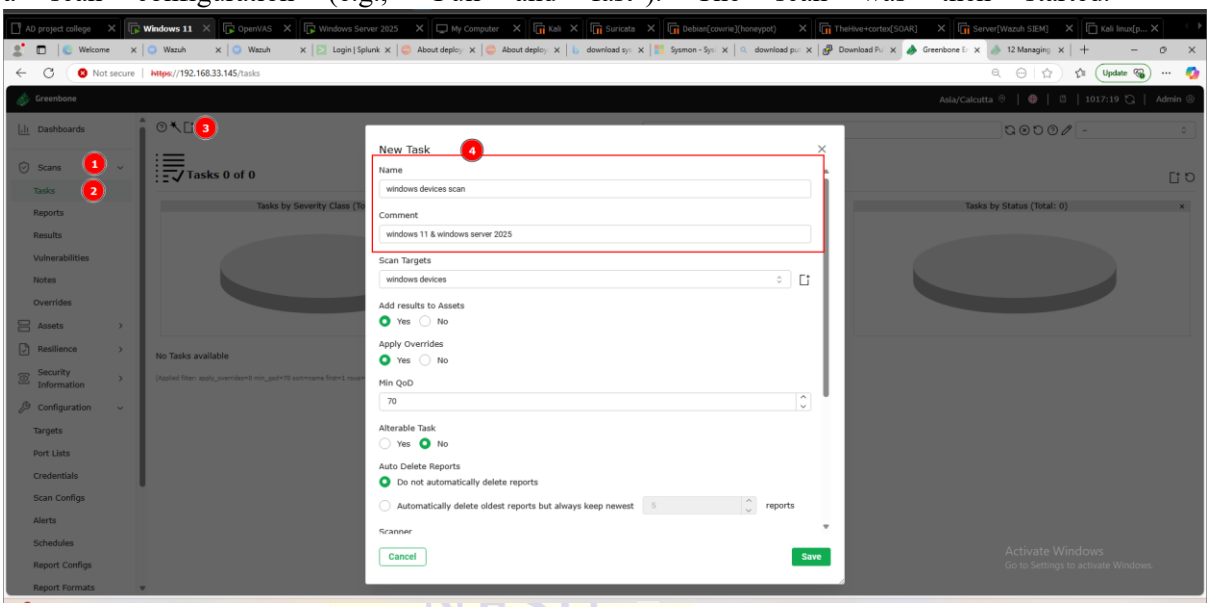

Figure 33 GSA New Task creation page showing target selection

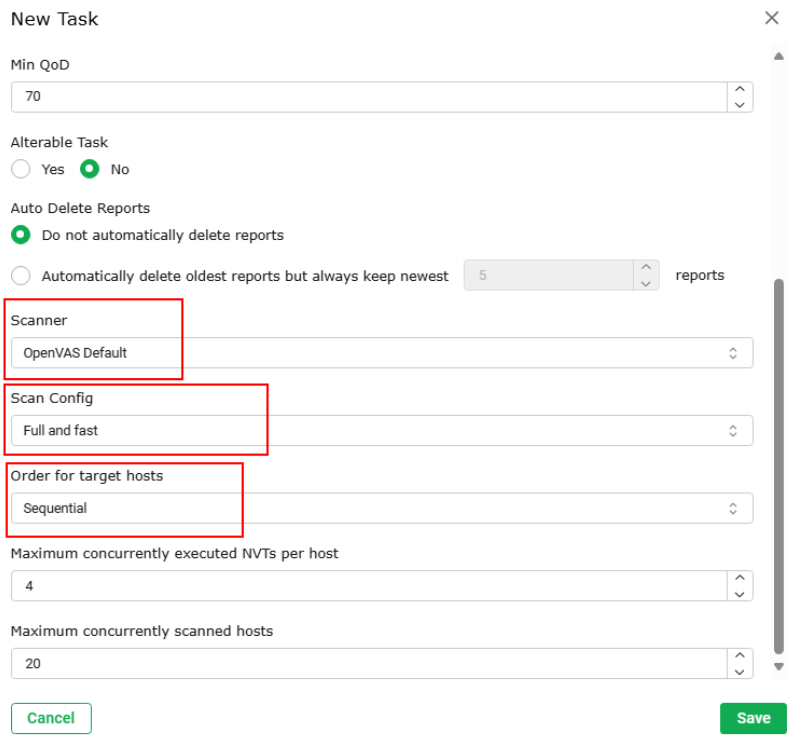

Figure 34 GSA Task page showing scan configuration selection

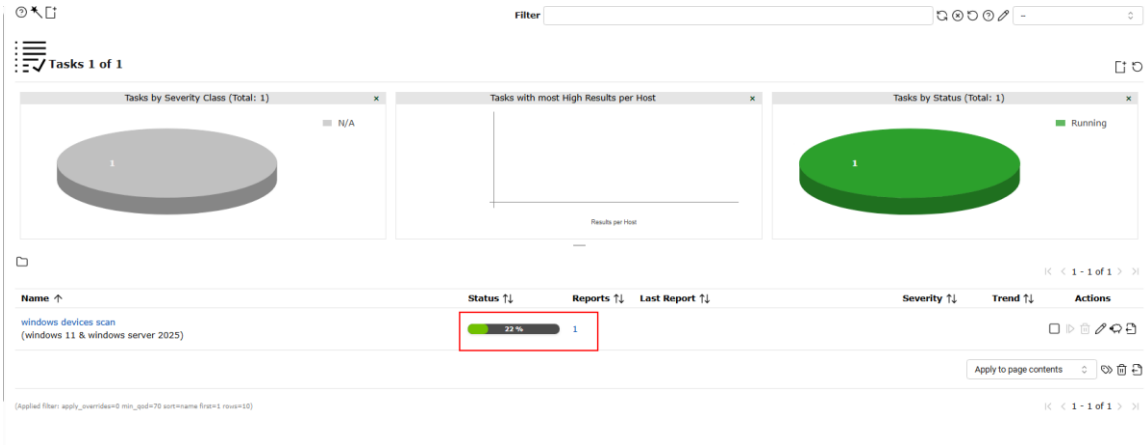

Figure 35 GSA Tasks page showing the scan task running

Scan results, including identified vulnerabilities and severity levels, are viewed within the GSA interface upon completion.

3.2.5 Deploying and Configuring Cowrie Honeypot

The Cowrie SSH and Telnet honeypot was installed on the dedicated Debian VM (192.168.33.142) to capture attacker interaction attempts.

Installation:

Cowrie was installed following the procedures outlined in the official Cowrie documentation on GitHub (https://github.com/cowrie/cowrie/blob/main/INSTALL.rst). This typically involves installing dependencies (git, python3-venv, libssl-dev, etc.), creating a dedicated user, cloning the Cowrie repository, setting up a Python virtual environment, and installing required Python packages using pip.

Configuration:

The primary configuration file, cowrie.cfg (copied from cowrie.cfg.dist), located in the etc/ directory within the Cowrie installation path, was modified. Key changes included:

- Setting the hostname to make the honeypot appear realistic.
- Enabling the Telnet listener alongside the default SSH listener by modifying the [telnet] section (enabled = true).

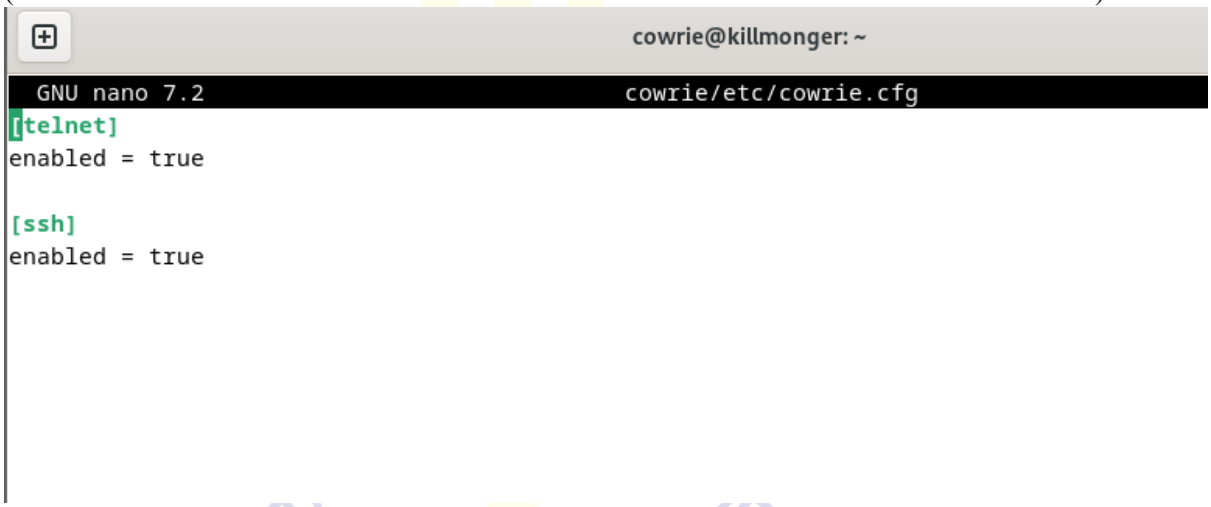

Figure 36 Section of cowrie.cfg showing enabled SSH and Telnet listeners

Log Collection via Wazuh Agent:

Cowrie logs (/opt/cowrie/var/log/cowrie/cowrie.log and cowrie.json) are monitored by the Wazuh agent installed on this VM. Specific rules within Wazuh are used to parse these logs and generate alerts based on honeypot activity. Direct log forwarding from Cowrie via syslog was not configured in this setup.

Starting Cowrie:

The honeypot was started using the cowrie start command from within the installation directory. It's recommended to configure it to run as a service for persistence.

Testing:

The honeypot's functionality was tested by attempting connections from the Kali Linux VM (192.168.33.143)

SSH Test: An SSH connection attempt was made to the Cowrie VM's IP address (192.168.33.142).

TEST #1

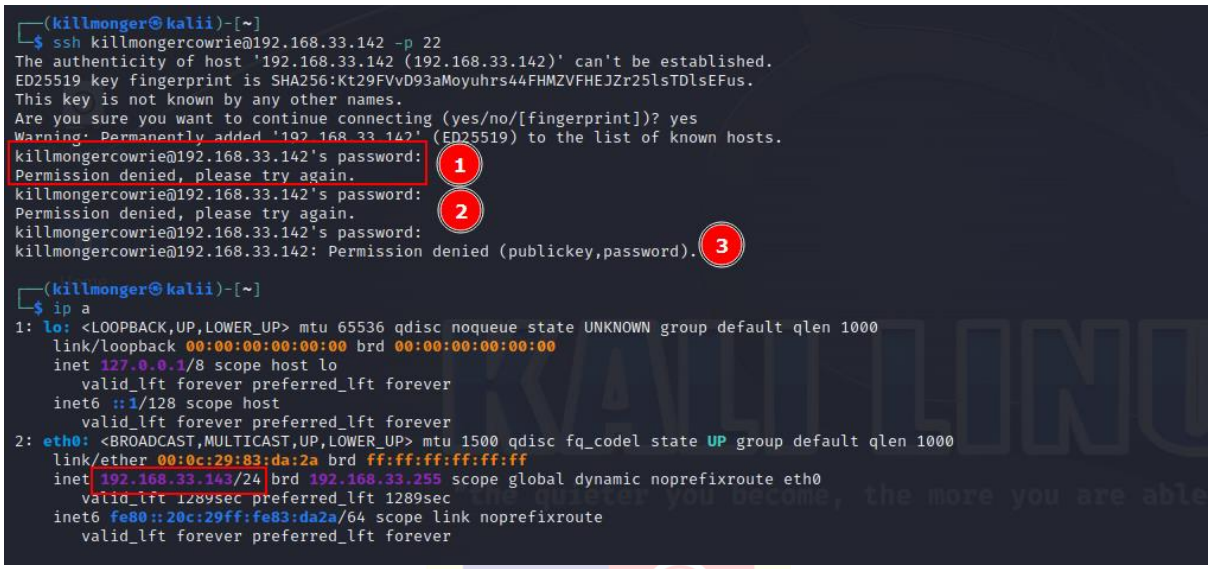

Figure 37 Kali terminal showing SSH connection attempt to Cowrie IP

Figure 38 Cowrie log file or console output showing the captured SSH attempt

TEST #2

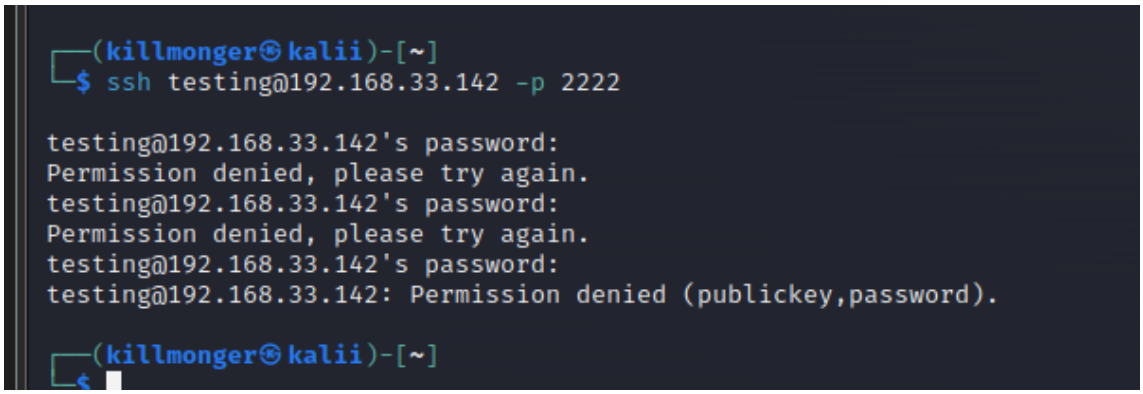

Figure 39 TEST no 2 of ssh to cowrie

Figure 40 OUTPUT of Test #2 showing the username and password in plain text

Verification in Wazuh: As the Wazuh agent is installed on the Cowrie VM and configured to monitor its logs, alerts related to Cowrie log entries (e.g., failed logins, commands executed) were observed in the Wazuh dashboard, confirming log collection and analysis via the agent.

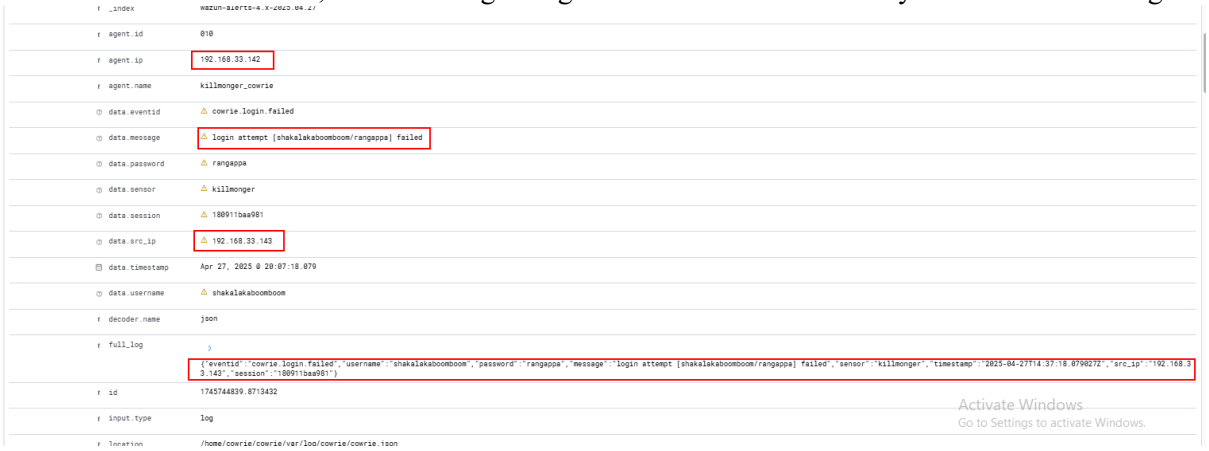

Figure 41 Wazuh dashboard showing alerts generated from Cowrie logs

3.2.5 Splunk Configuration for Log Aggregation and Analysis

Splunk Enterprise (Free License) was installed on the Ubuntu Server VM (192.168.33.128) to serve as the central log aggregation and analysis platform.

Installation:

Splunk Enterprise was installed using the .deb package downloaded from the Splunk website.

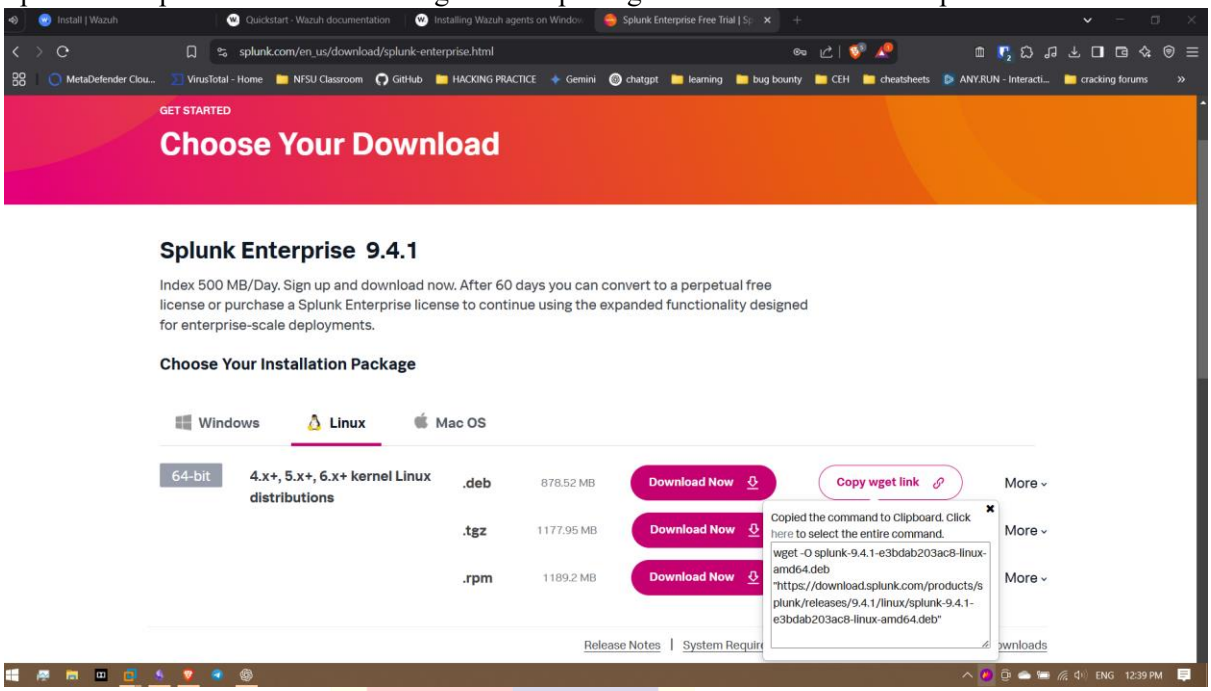

Figure 42 Splunk official website to download .deb file

- ***wget -O splunk.deb*** ***https://download.splunk.com/products/splunk/releases/9.4.1/linux/splunk-9.4.1-e3bdab203ac8-linux-amd64.deb***

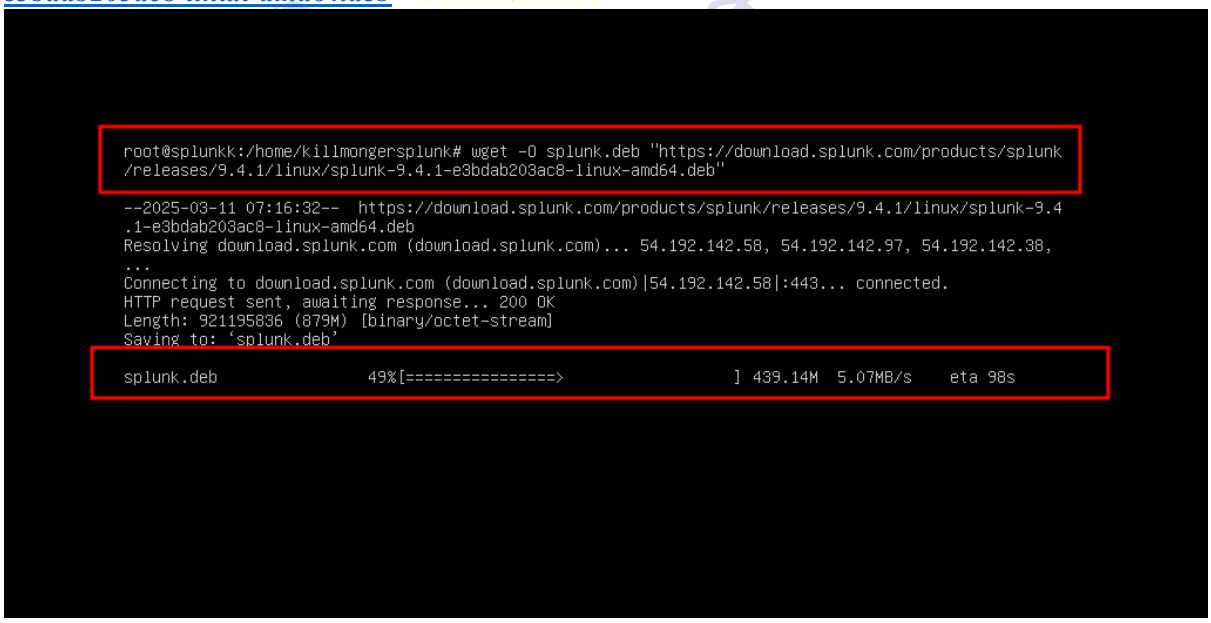

Figure 43 Terminal showing wget command downloading Splunk

- Splunk was started for the first time, accepting the license agreement and setting the initial administrator password:

  sudo /opt/splunk/bin/splunk start --accept-license

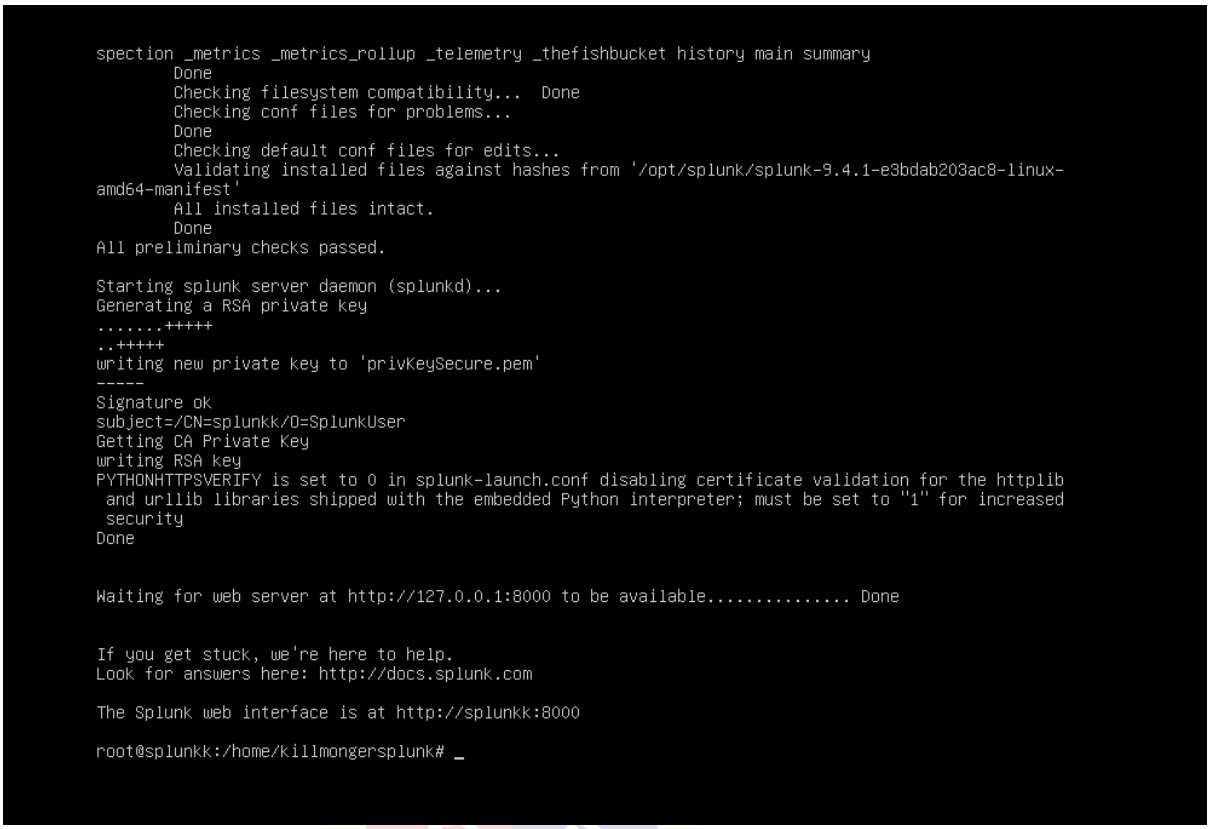

Figure 44 At bottom We can find splunk dashboard is live

Accessing Splunk Web:

The Splunk Web interface was accessed from another machine on the network via http://192.168.33.128:8000.

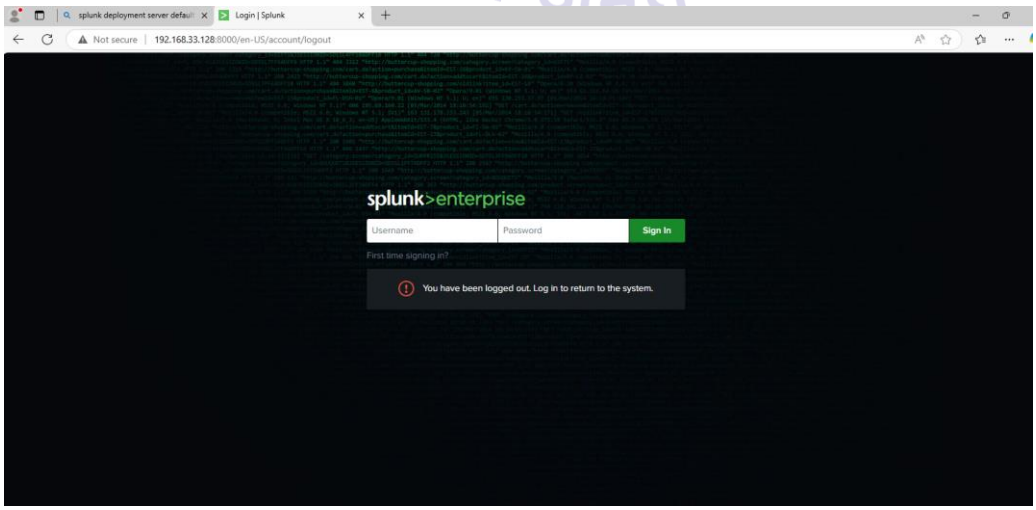

Figure 45 Accessing splunk dashboard login page

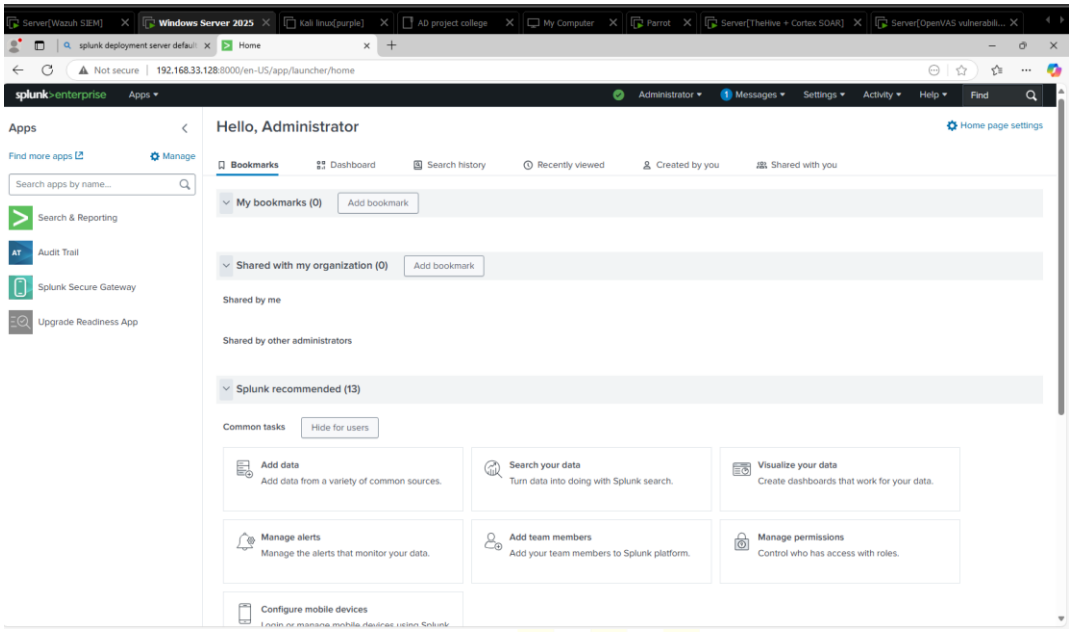

Figure 46 Splunk dashboard after logging in

Data Inputs Configuration:

Splunk was configured to receive data from the various sources within the ICDE:

- **Splunk Forwarder Input:** The Splunk TCP input was enabled (typically via Settings > Forwarding and receiving > Configure receiving in the UI, or by editing inputs.conf) on the default port 9997. This allows Splunk to receive logs forwarded by the Universal Forwarder installed on the Wazuh Manager VM.

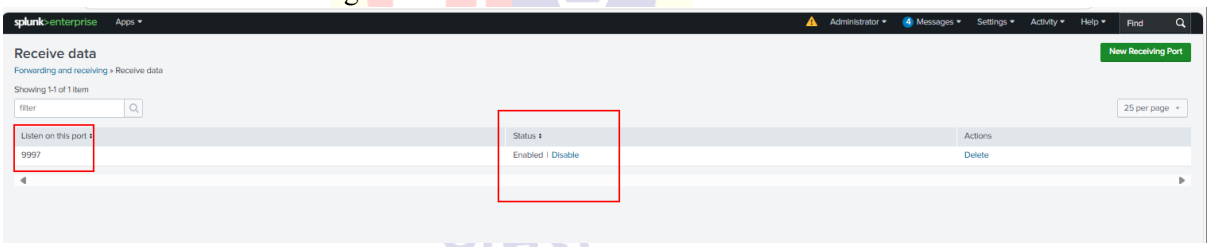

Figure 47Splunk UI showing TCP input on port 9997 enabled

- **Index Creation:** Specific indexes were created to organize the incoming data (e.g., via Settings > Indexes.

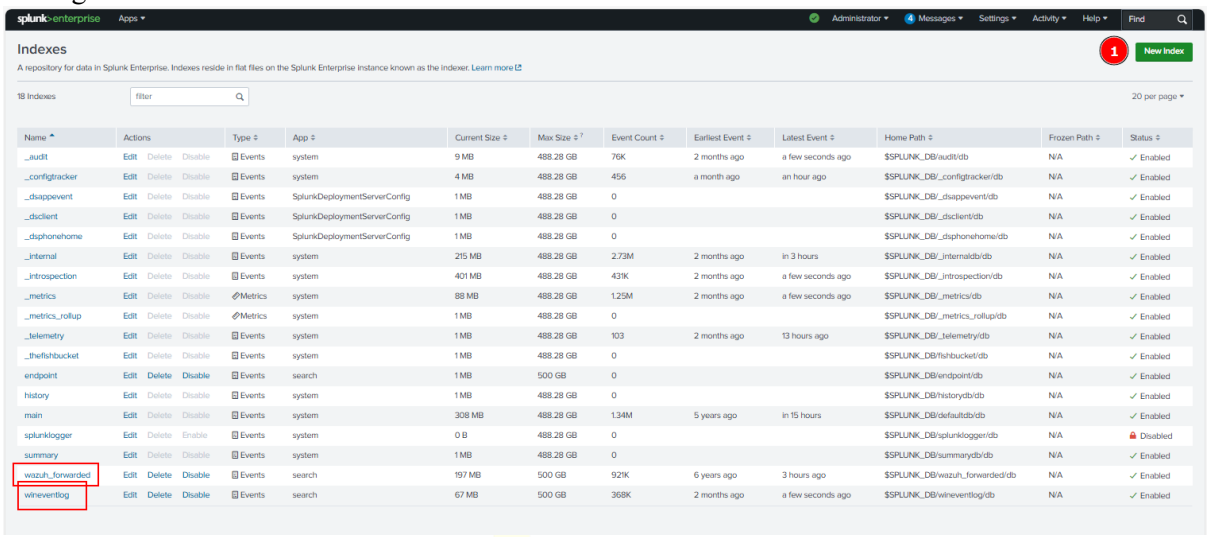

Figure 48 Custom indexes made for ICDE in splunk

3.2.6 Active Directory and Endpoint Setup for Realistic Simulation

A realistic target environment was established using Windows Server 2025 as the Active Directory Domain Controller and Windows 11 as a client endpoint.

Active Directory Domain Setup:

The Windows Server 2025 VM (192.168.33.129) was promoted to a Domain Controller for a new forest with the domain name ***home.lab***. This involved installing the Active Directory Domain Services (AD DS) role and running the promotion wizard.

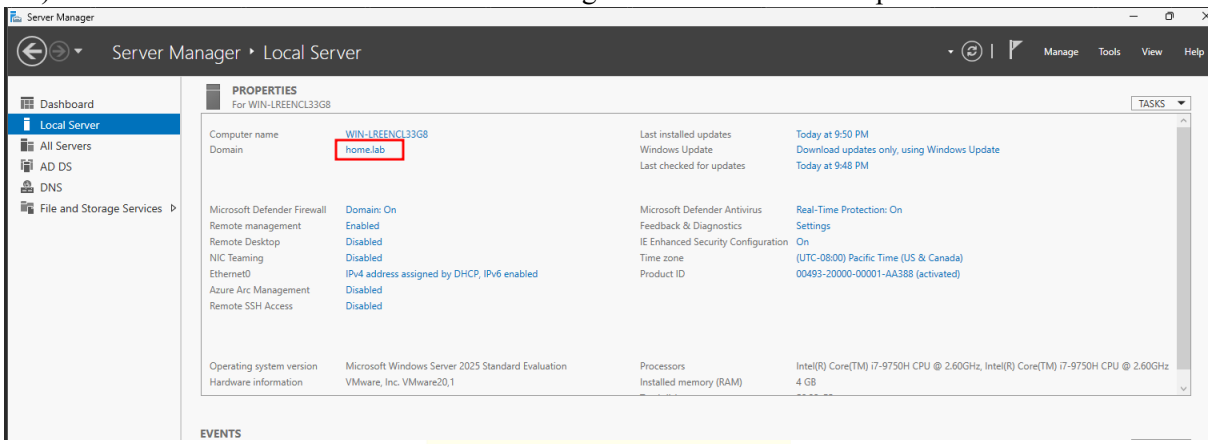

Figure 49 We can view home.lab Domain created

Organizational Units (OUs), Users, and Groups: To simulate a corporate structure, several OUs, user accounts, and security groups were created using the "Active Directory Users and Computers" tool:

- **OU Structure:** OUs such as IT, Finance, Security, GeneralUsers, Workstations, and Servers were created to organize objects logically.

Users and Groups: Sample users (AdminUser, FinanceUser, SecurityAnalyst, Bob, Alice) and groups (IT Admins, Finance Team, Security Analysts, Log Readers, Remote Access Users) were created within their respective OUs. Group memberships were configured as follows:

OU Name	Groups Inside OU	Users Inside OU	Group Memberships
IT	IT Admins	AdminUser	AdminUser → IT Admins, Domain Users
Finance	Finance Team	FinanceUser	FinanceUser → Finance Team, Domain Users
Security	Security Analysts, Log Readers	SecurityAnalyst	SecurityAnalyst → Security Analysts, Log Readers, Domain Users
GeneralUsers	Remote Access Users	Bob, Alice	Bob, Alice → Remote Access Users, Domain Users
Workstations	(No groups, just stores computers)	Windows 11 client	(N/A)
Servers	(No groups, just stores servers)	Windows Server, Wazuh, TheHive, Splunk, OpenVAS, Cowrie, Zeek	(N/A)

Table 3 Simulated Organizational Structure in Active Directory

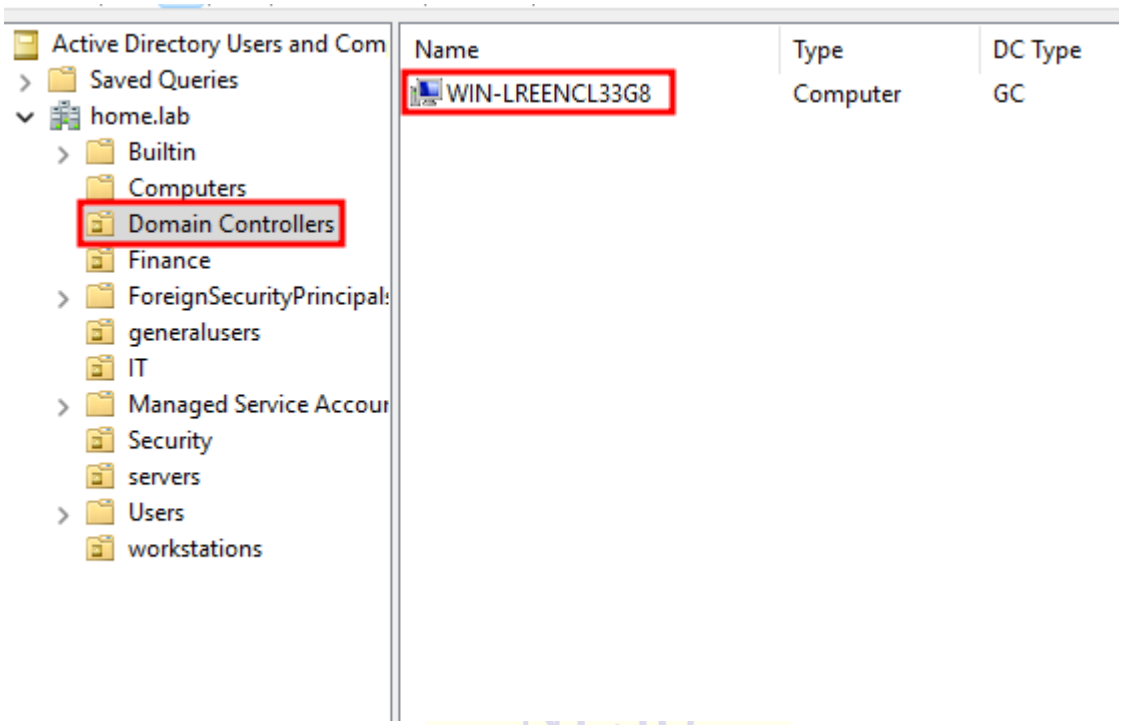

Figure 50 AD Users & Computers showing Domain Controllers OU contents

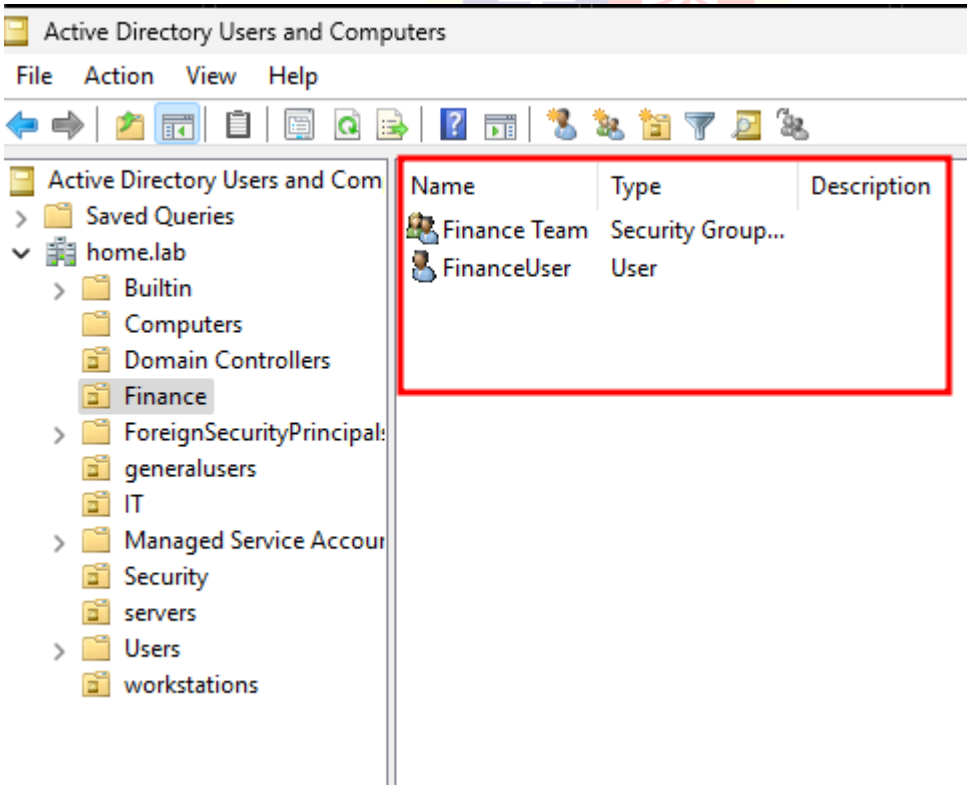

Figure 51 AD Users & Computers showing Finance OU contents

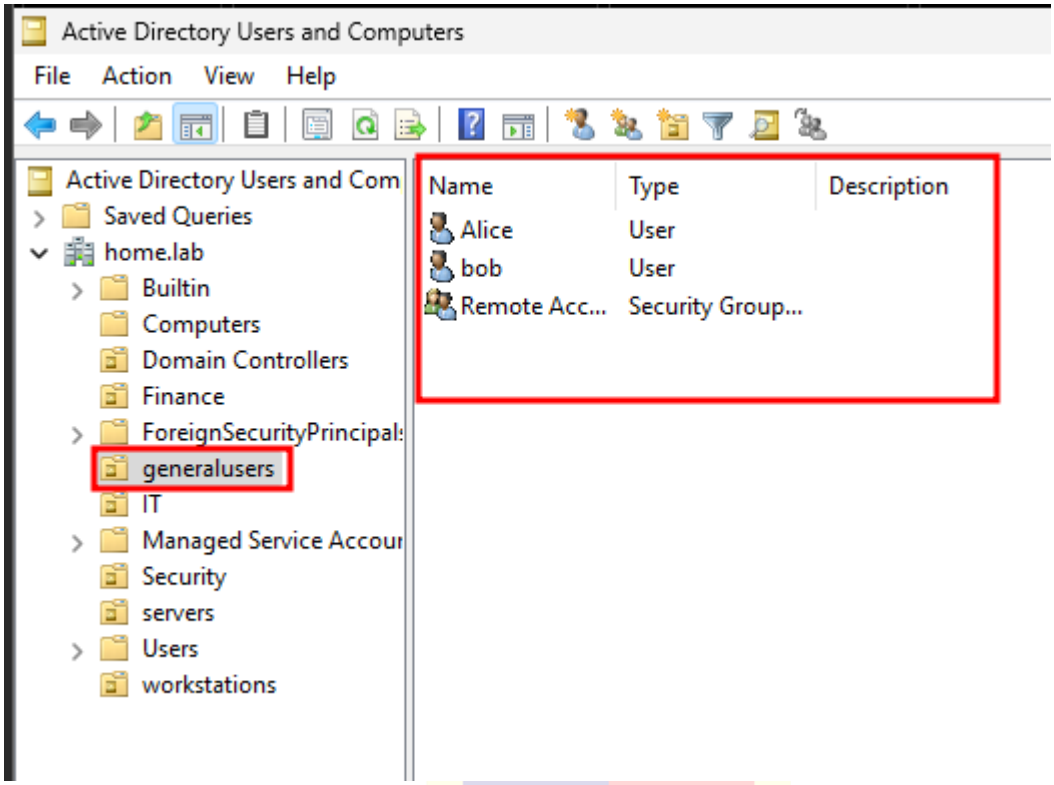

Figure 52 AD Users & Computers showing GeneralUsers OU contents

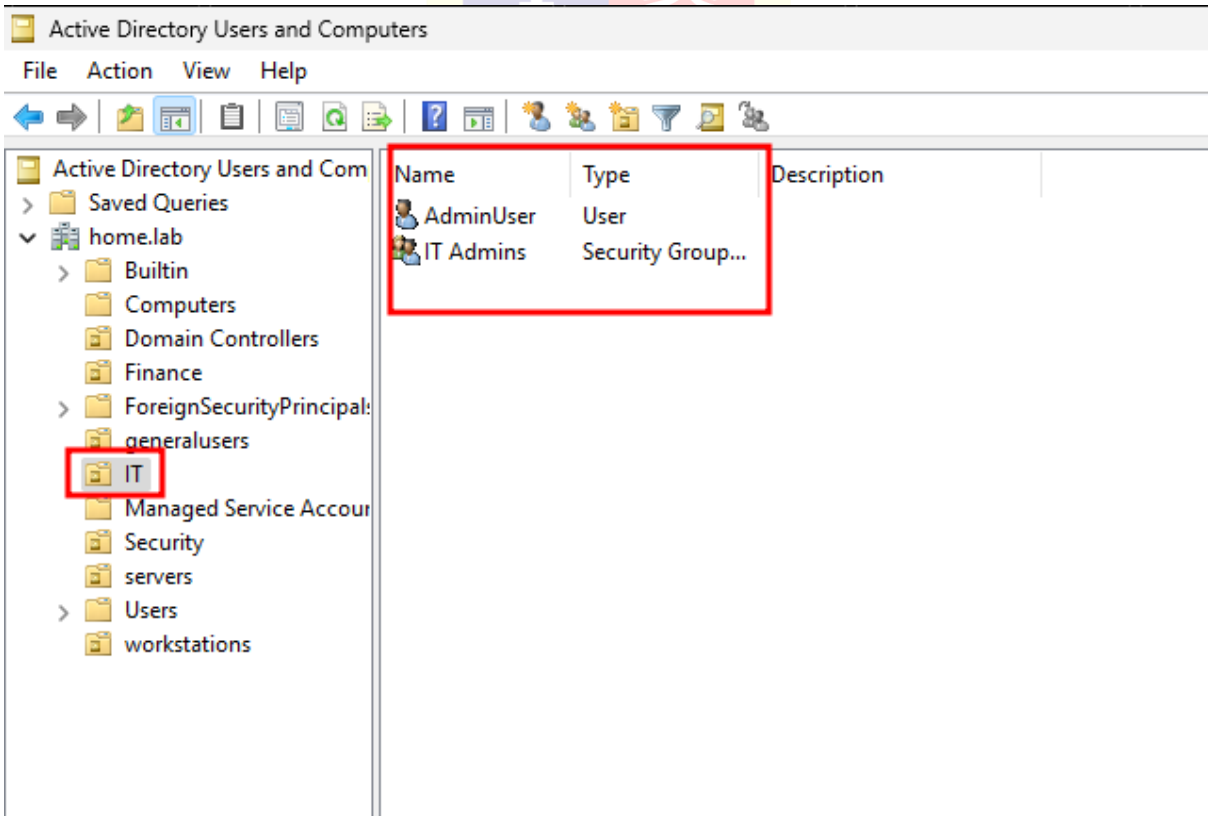

Figure 53 AD Users & Computers showing IT OU contents

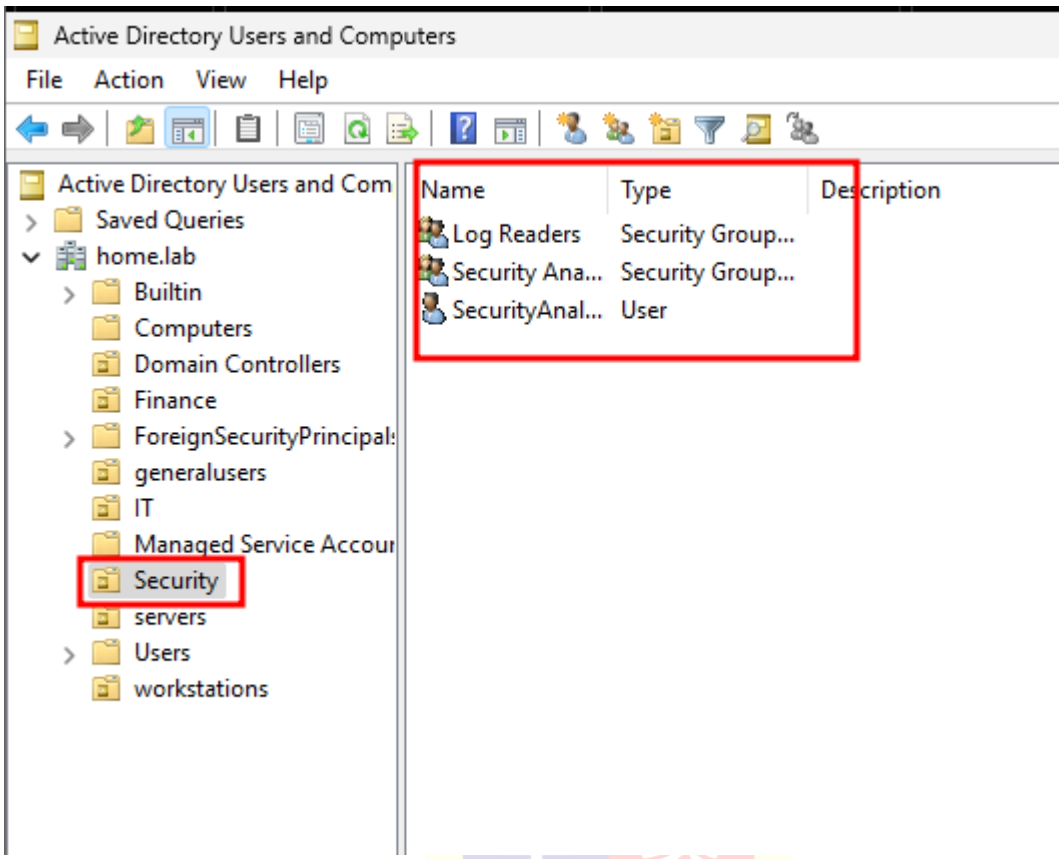

Figure 54 AD Users & Computers showing Security OU contents

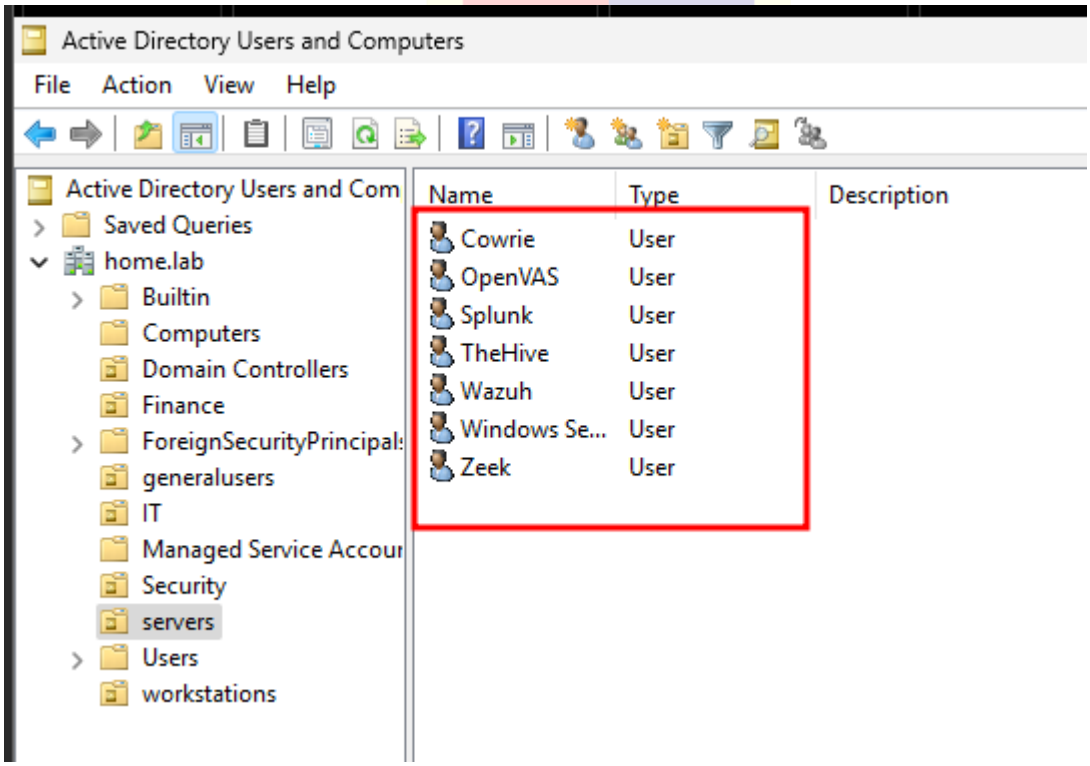

Figure 55 AD Users & Computers showing Servers OU contents

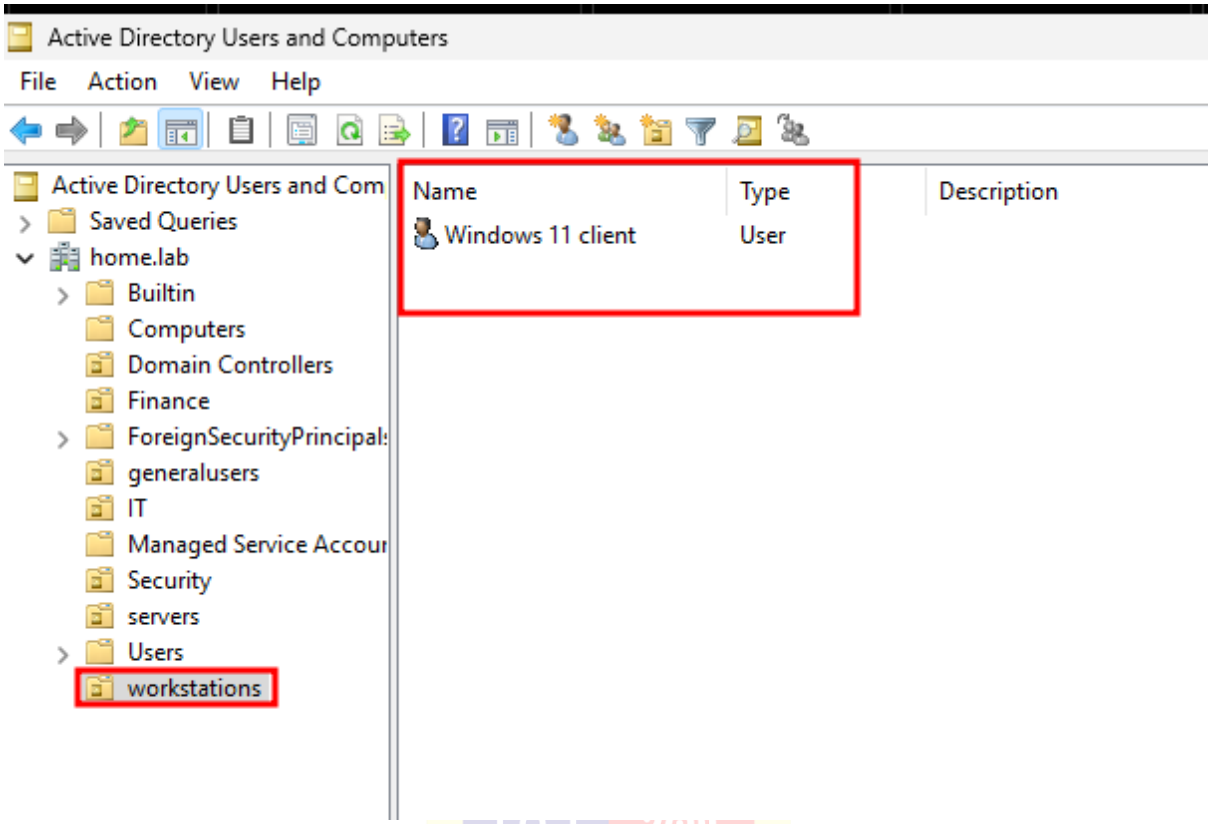

Figure 56 AD Users & Computers showing Workstations OU contents

Endpoint Joining:

The Windows 11 VM (192.168.33.130) was joined to the home.lab domain.

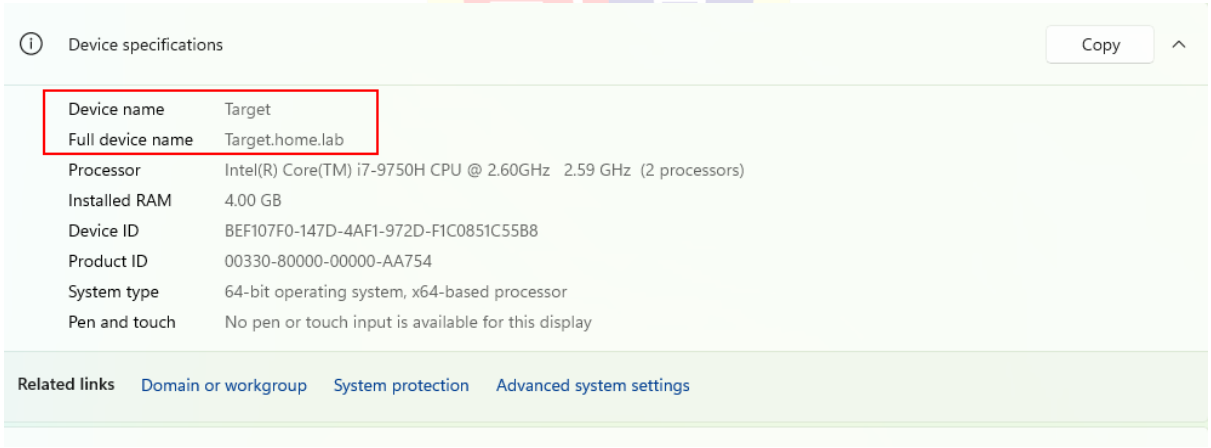

Figure 57 home.lab joined by target machine [windows 11]

Enhanced Logging via Group Policy:

To improve visibility for detection and response, specific logging policies were enabled via Group Policy Objects (GPOs), primarily applied through the ***Default Domain Policy.***

- **Logon/Logoff Auditing:** Using the Group Policy Management Console (gpmc.msc), the Default Domain Policy was edited. Under Computer Configuration → Policies → Windows Settings → Security Settings → Advanced Audit Policy Configuration → Audit Policies → Logon/Logoff, the Audit Logon policy was configured to log both Success and Failure events. This captures Event IDs like 4624 (successful logon) and 4625 (failed logon) in the Security event log, crucial for detecting brute-force attempts and tracking user activity.

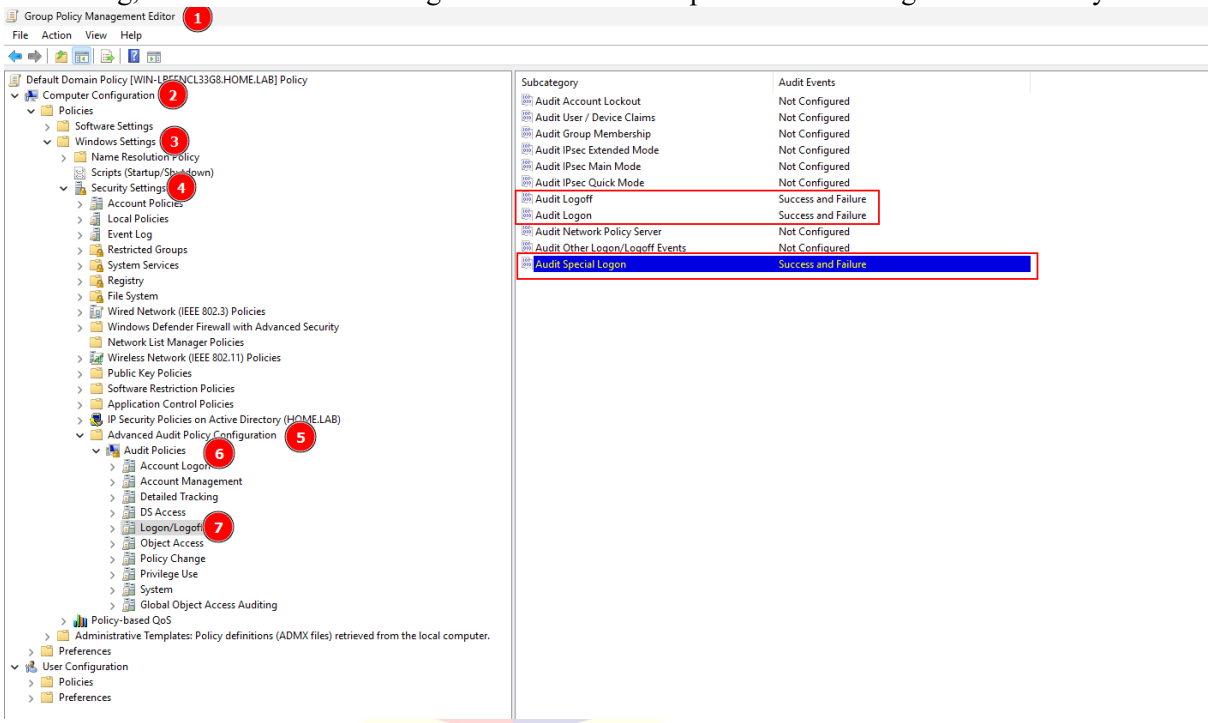

Figure 58 Group Policy Management Editor showing Audit Logon policy enabled for Success and Failure

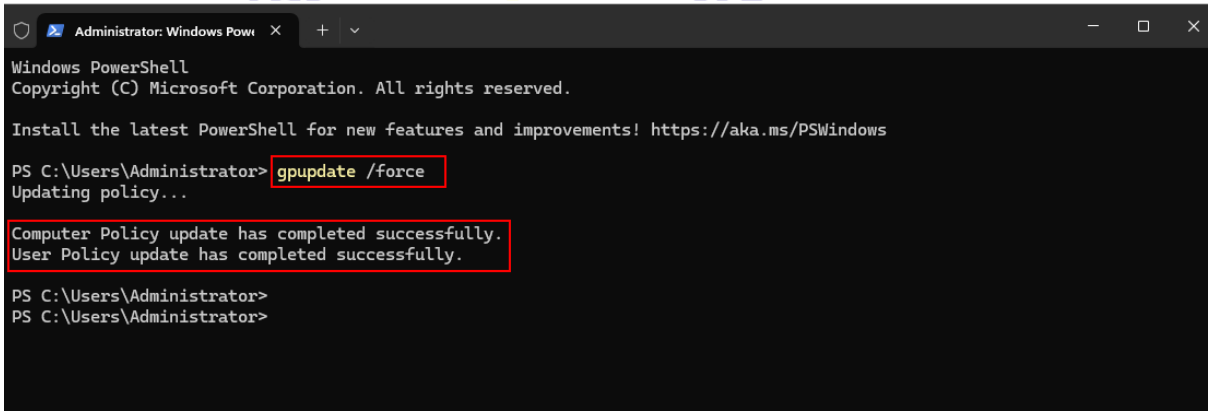

Figure 59 Force updating the policies which we changed

PowerShell Logging:

Using the Group Policy Management Editor (or Local Group Policy Editor gpedit.msc applied domain-wide), under Computer Configuration → Administrative Templates → Windows Components → Windows PowerShell, the following policies were enabled:

- Turn on PowerShell Script Block Logging: Logs the content of scripts executed.
- Turn on Module Logging: Logs pipeline execution details, specifying * for modules to log activity from all modules. These settings enhance the detection of fileless attacks and malicious script usage by logging detailed PowerShell activity (Event IDs 4103, 4104) in the Microsoft-Windows-PowerShell/Operational event log.

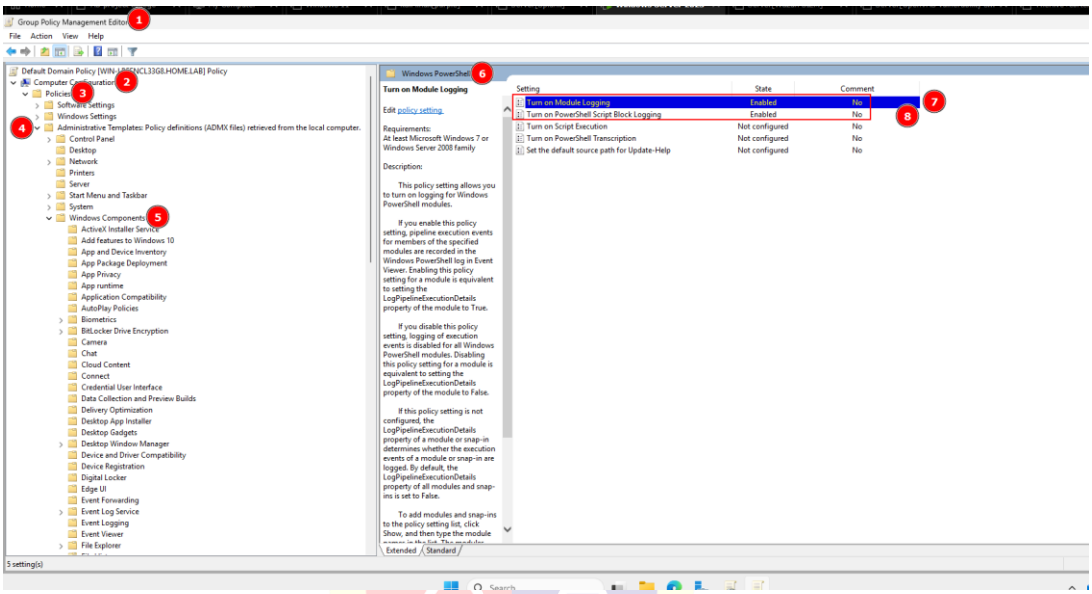

Figure 60 Steps to access the windows powershell configuration

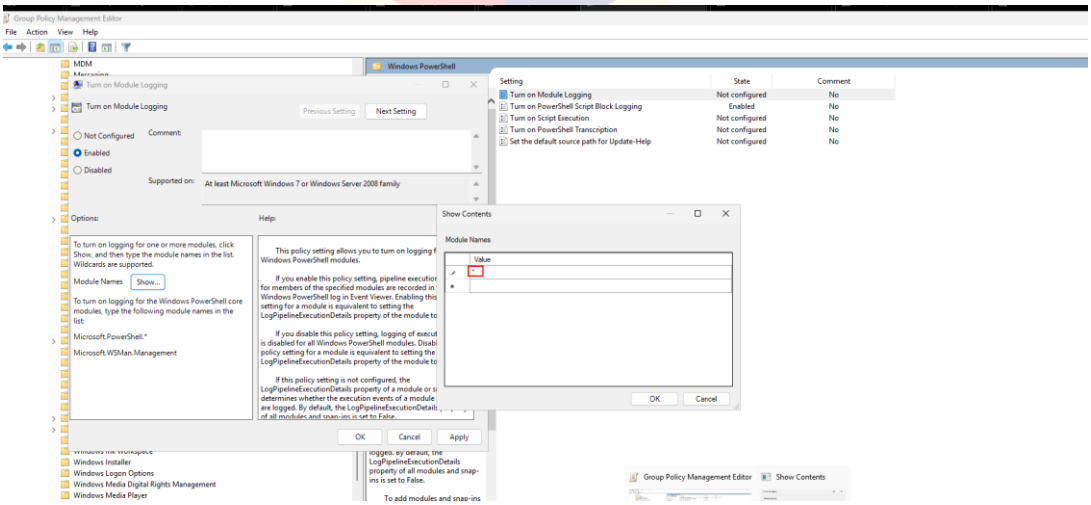

Figure 61 Group Policy setting for enabling Module Logging with ''*

Sysmon Installation:

System Monitor (Sysmon), part of the Sysinternals suite, was installed on the Windows endpoints (Server and Workstation) to provide deep visibility into process creation, network connections, and other system events. It was installed using a configuration file based on Olaf Hartong's sysmon-modular project to provide a robust starting point for event collection.

.\sysmon.exe -i sysmonconfig.xml

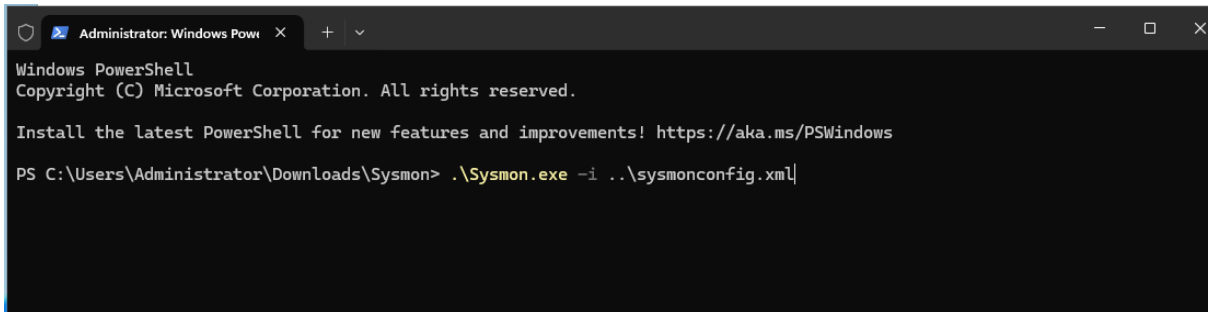

Figure 62 Installation of sysmon using olaf's sysmonconfig.xml

Agent Installations:

As detailed in section 3.2.2, Wazuh agents were installed on both the Windows Server 2025 DC and the Windows 11 endpoint to collect security events, including those generated by the enhanced audit policies and Sysmon, forwarding them to the Wazuh Manager.

3.2.8 Implementing Automation Workflows with Shuffle

Shuffle was successfully installed and configured within the virtual environment, although full workflow automation is designated as future work. Key steps included:

- Setting up the core Shuffle services (Backend, Worker, Frontend) on the designated Ubuntu Server VM (192.168.33.149). *[Docker Compose]*

Figure 63 Terminal showing Shuffle installation command

- Configuring user authentication for accessing the Shuffle UI.

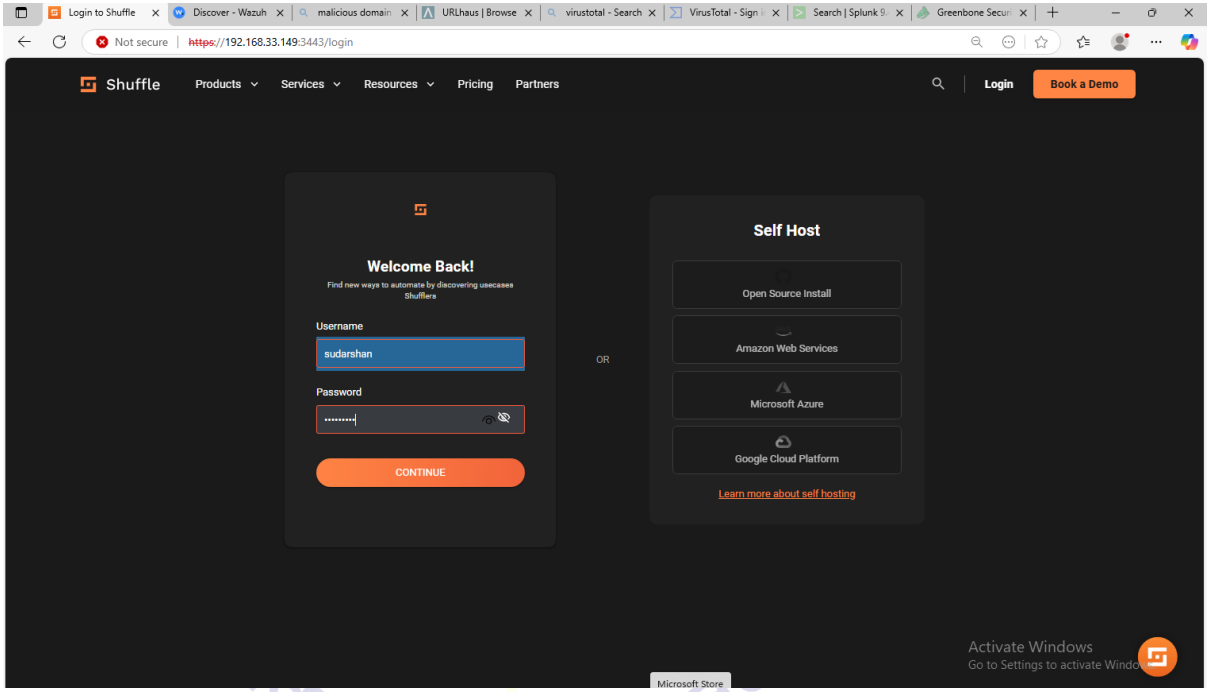

Figure 64 Shuffle UI Login Page

- Configuring a webhook listener in Shuffle and modifying Wazuh's ossec.conf to send alerts via webhook integration.
    1. Create workflow and name it
    2. Add webhook
    3. Name the webhook
    4. Copy the webhook url

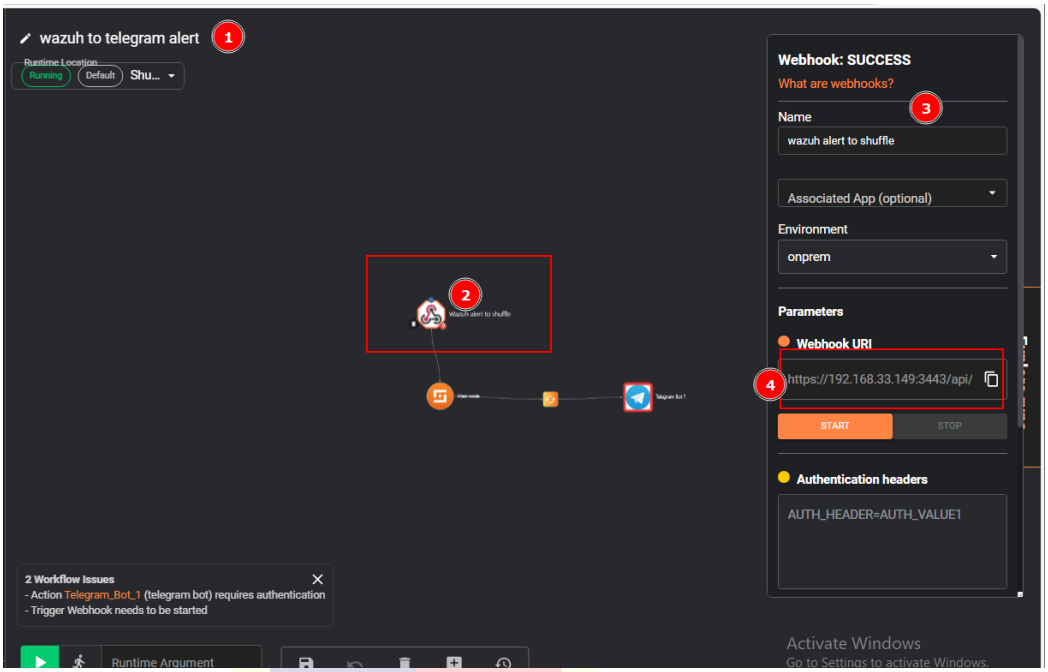

Figure 65 steps to create webhook

- Copy the webhook in following location in wazuh manager.
  sudo nano /var/ossec/etc/ossec.conf
- Paste the url like below picture

```
    <logall>yes</logall>
    <logall_json>yes</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>wazuh@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
  </global>
  <integration>
    <name>wazuh-cowrie-alert</name>
    <hook_url>https://192.168.33.149:3443/api/v1/hooks/webhook_dd0acdc7-7804-4d6f-884e-eb12ac72894a </hook_url>
    <rule_id>120003</rule_id>
    <alert_format>json</alert_format>
  </integration>
  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>12</email_alert_level>
  </alerts>

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

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

  <!-- Policy monitoring -->
```

Figure 66 pasting the webhook url in ossec.conf file

- **Successfully configuring Shuffle to ingest alerts/events from Wazuh:** Test alerts generated by Wazuh were verified to appear within the Shuffle interface, ready for processing.
- Configuring the Telegram integration by adding the necessary Bot Token to allow Shuffle to send notifications.

Figure 67 Shuffle UI showing the configuration of the Telegram App/Action node

- The Shuffle workflow designer interface was explored, and foundational structures for potential automation playbooks were conceptualized.

Figure 68 Shuffle workflow designer showing a basic conceptual workflow

4.0Results and Discussions

This chapter presents the results obtained from testing the implemented Integrated Cyber Defense Environment (ICDE). It analyzes the effectiveness of each deployed security tool in detecting simulated attacks and managing security events, discusses the integration outcomes, and evaluates the overall functionality of the lab setup.

4.1 Automation of alert sending to analyst at time of bruteforce ssh to cowrie honeypot

A key test involved simulating an attacker attempting to gain unauthorized access via SSH to the Cowrie honeypot.

Attack Simulation: An SSH connection attempt was initiated from the Kali Linux VM (192.168.33.143) to the Cowrie honeypot VM (192.168.33.142) on its listening port 2222, using arbitrary credentials.

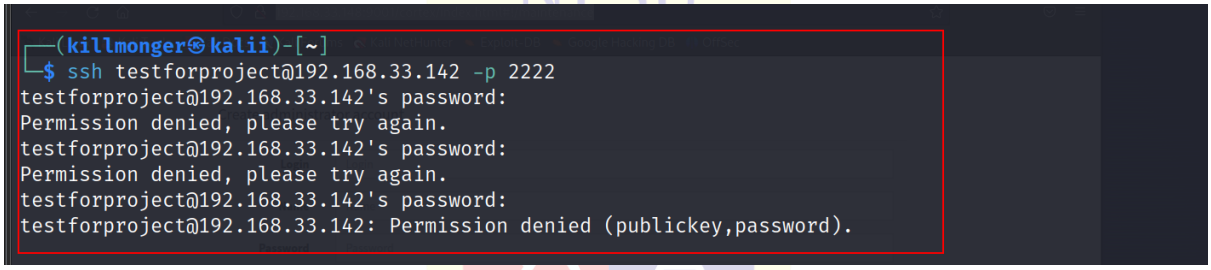

Figure 69 kali Linux terminal showing the SSH command targeting the Cowrie VM's IP and port

Honeypot Logging: Cowrie successfully captured the failed login attempt, logging details such as the source IP, username, and password used.

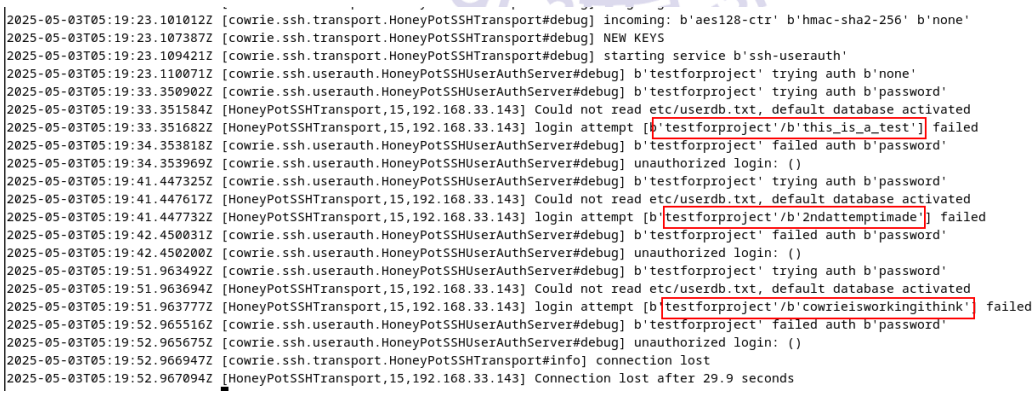

Figure 70 Cowrie storing logs locally at cowrie/var/log/cowrie/cowrie.json

Wazuh Agent Collection: The Wazuh agent running on the Cowrie VM monitored the Cowrie log files and forwarded the relevant log entry to the Wazuh Manager (192.168.33.131).

Wazuh Alert Generation: The Wazuh Manager processed the incoming log, matching it against custom rule 120003 specifically created to detect Cowrie failed logins. This generated a high-severity alert (Level 15).

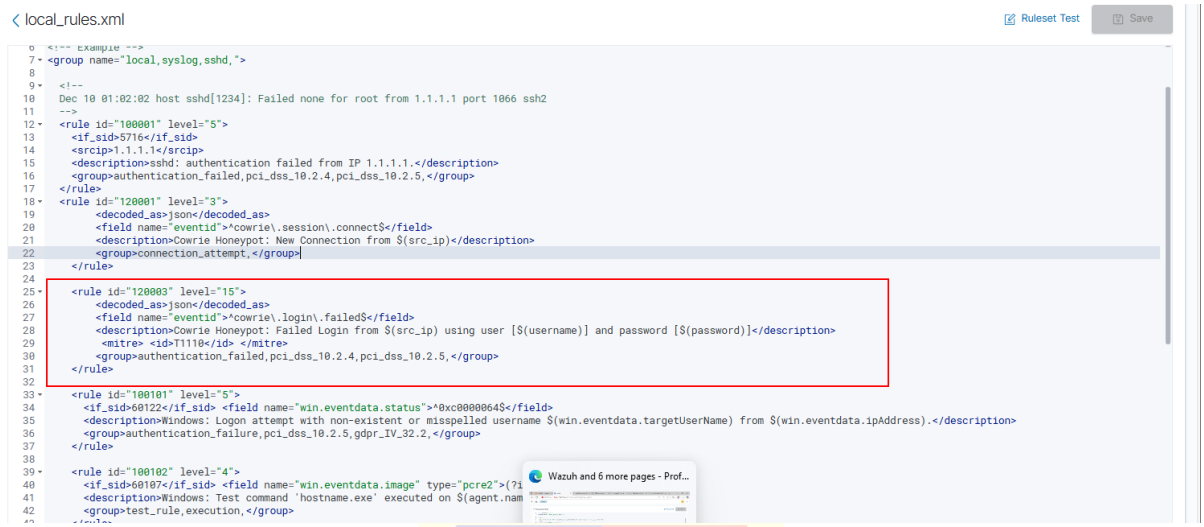

Figure 71 Checking the custom rule made for cowrie ssh attempt in local_rules.xml

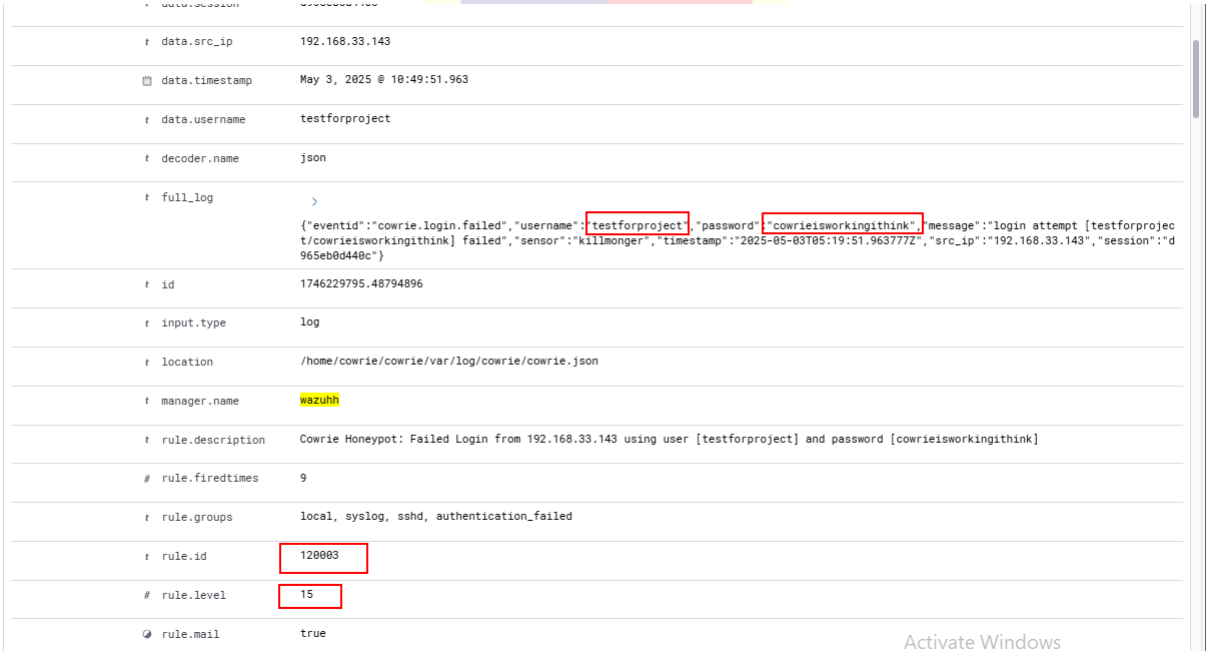

Figure 72 Wazuh Dashboard showing the generated alert with Rule ID 120003, Level 15, and details from the log

Webhook Integration with Shuffle: Wazuh was configured to send alerts matching certain criteria to a configured Shuffle webhook URL.

```
<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>yes</logall>
    <logall_json>yes</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>wazuh@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
  </global>
  <integration>
    <name>wazuh-cowrie-alert</name>
    <hook_url>https://192.168.33.149:3443/api/v1/hooks/webhook_dd0acdc7-7804-4d6f-884e-eb12ac72894a </hook_url>
    <rule_id>120003</rule_id>
    <alert_format>json</alert_format>
  </integration>
  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>12</email_alert_level>
  </alerts>

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

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

  <!-- Policy monitoring -->
```

Figure 73 inserting code in ossec.conf to send alert via webhook to shuffle

Shuffle Notification: Upon receiving the alert via webhook, a simple Shuffle workflow was triggered, which extracted key information and sent a notification message via the configured Telegram Bot.

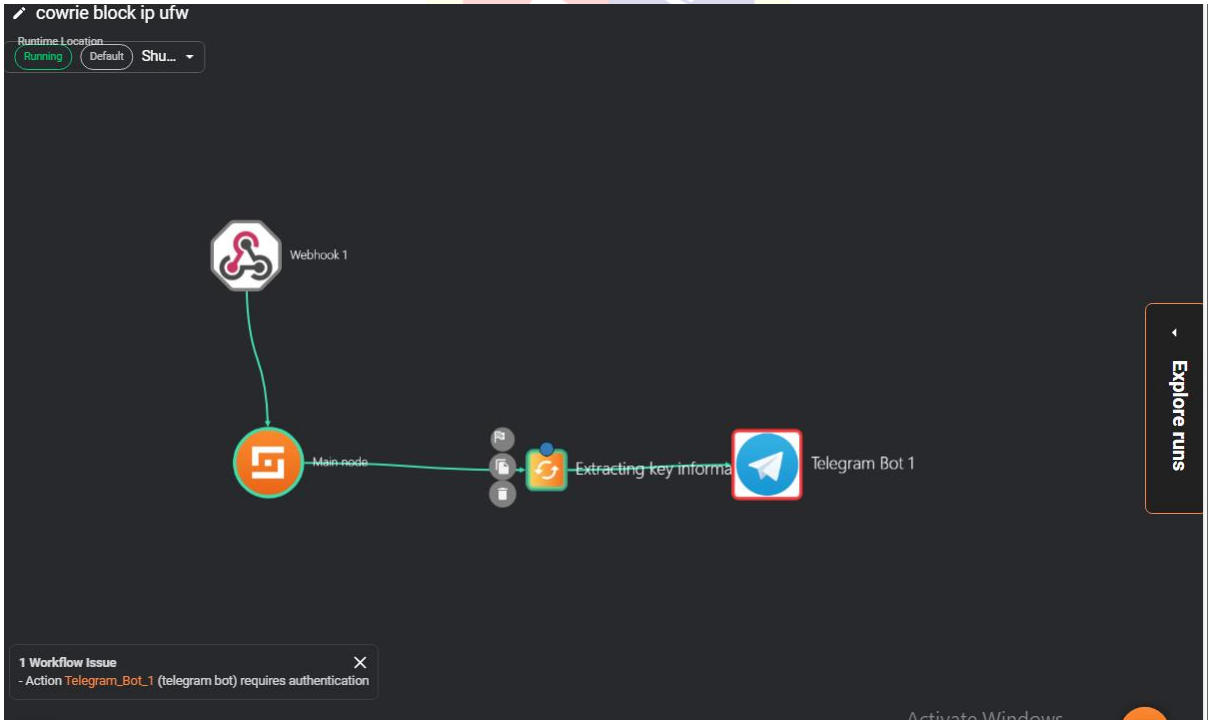

Figure 74 Cowrie telegram workflow on shuffle

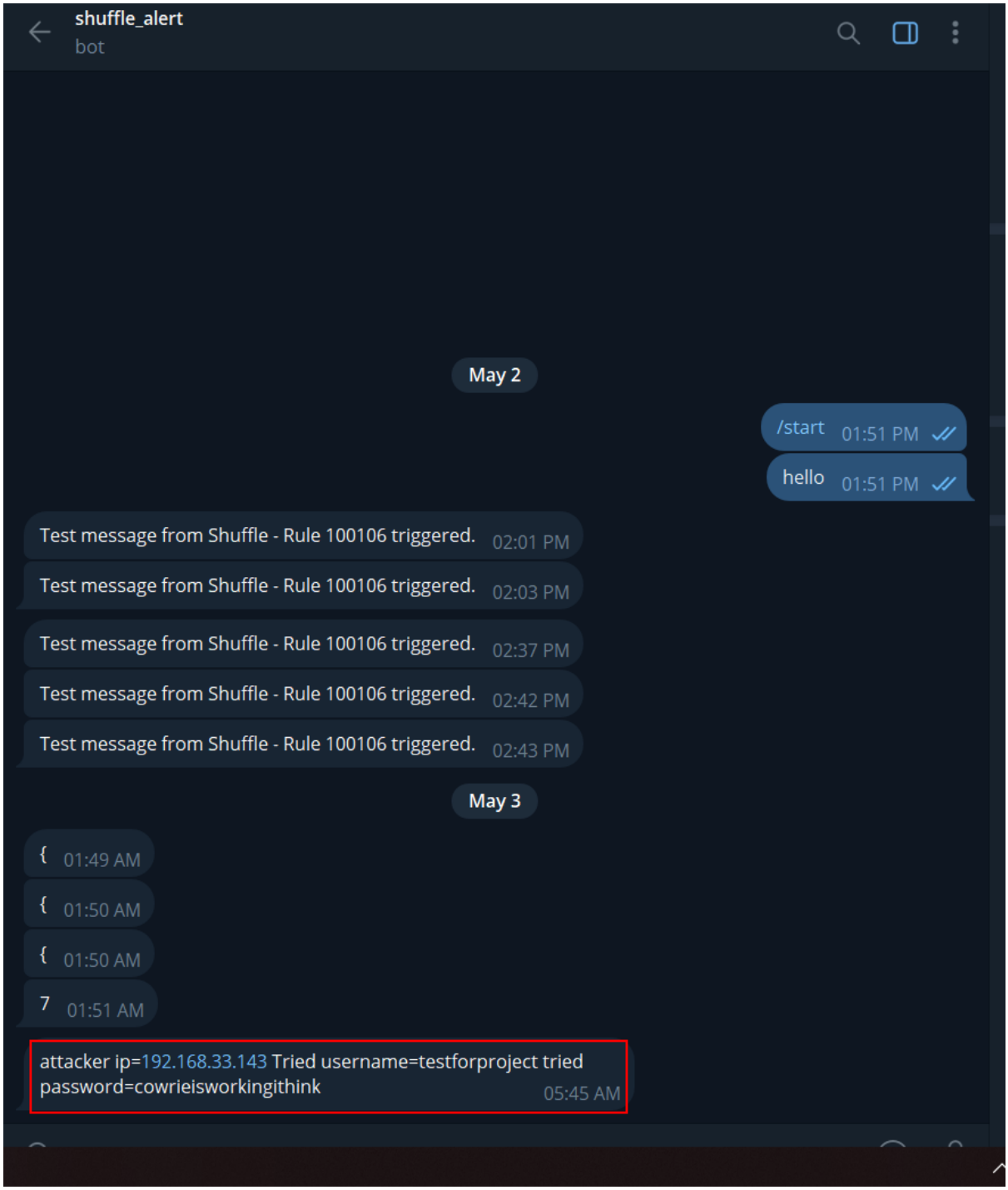

Figure 75 Telegram alert being recieved about the ssh trial from kali machine

4.2 Evaluation of Suricata's Network Intrusion Detection Capabilities

Suricata's effectiveness as a Network Intrusion Detection System (NIDS) within the ICDE was evaluated by simulating common network reconnaissance activities.

Attack Simulation (Network Scan): An Nmap port scan (a SYN scan) was launched from the Kali Linux VM (192.168.33.143) targeting the Windows Server DC (192.168.33.129).

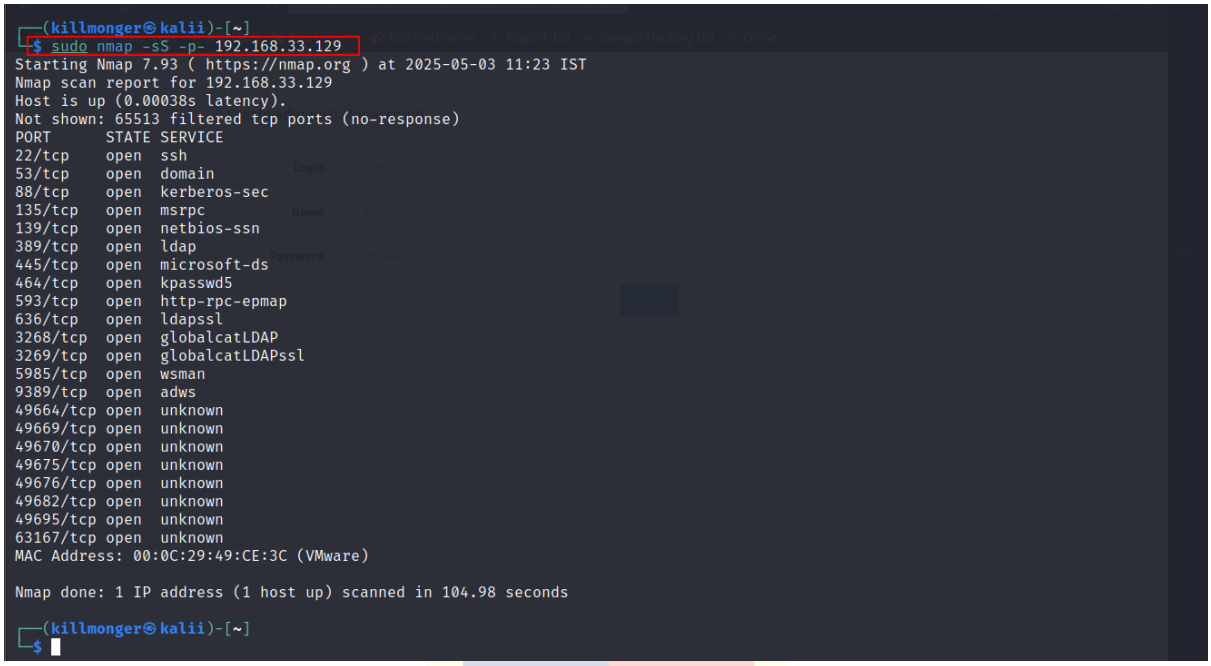

Figure 76 performing TCP syn port scan on windows server 2025

Suricata Detection: As the Nmap scan traffic traversed the monitored network segment (icde-project (NAT), Suricata processed the packets against its loaded rulesets (including the ET Open ruleset). The scan activity matched specific signatures designed to detect network scanning behavior.

Wazuh Agent Queue Flooding: The Wazuh agent running on the Suricata VM attempted to read the large volume of alerts generated in eve.json and forward them to the Wazuh Manager. However, the sheer number of events produced by the -p- (all ports) scan overwhelmed the agent's internal event queue faster than it could transmit them to the manager.

Observed Wazuh Alerts: Consequently, instead of receiving the specific Suricata scan detection alerts, the Wazuh Manager received repeated alerts from the Suricata agent (agent.id:012,

agent.name:suricata) indicating a flooded buffer, specifically **Rule ID 203: "Agent event queue is full. Events may be lost."** and potentially **Rule ID 202: "Agent event queue is 90% full."**

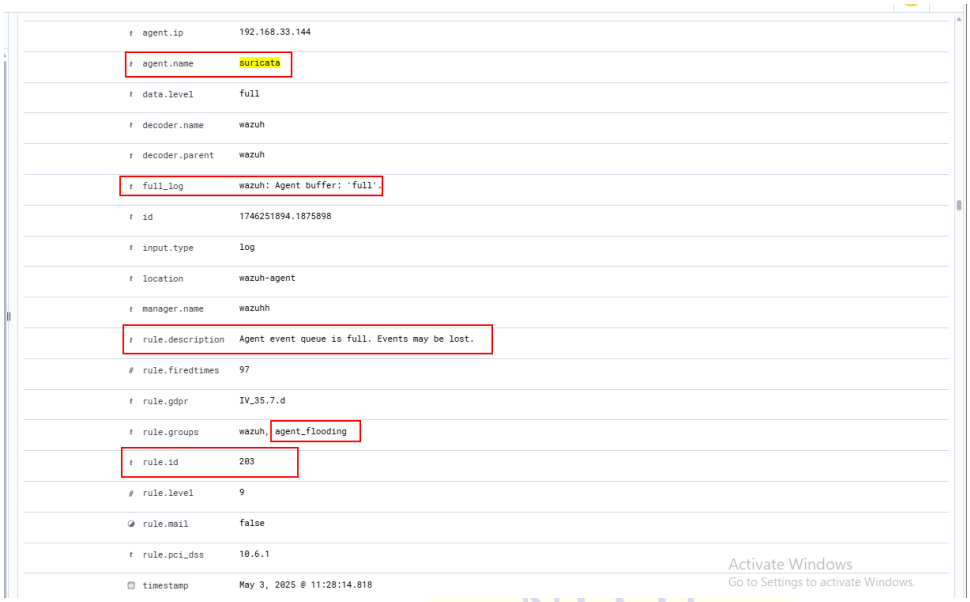

Figure 77 Wazuh Dashboard showing repeated alerts for Rule ID 203 originating from the 'suricata' agent during the Nmap scan test

Conclusion:

This test demonstrated that while Suricata was likely effective in *detecting* the aggressive network scan locally, the integration pipeline via the Wazuh agent was overwhelmed by the high volume of resulting events. This led to the loss of specific detection alerts at the manager level, replaced by alerts about the agent's buffer limitations. This highlights a potential need for performance tuning, either by adjusting the Wazuh agent's queue size (<queue_size>) and EPS (events per second) limits in its ossec.conf (if feasible), rate-limiting certain noisy rules within Suricata itself, or using less aggressive scanning techniques for standard testing. The result validates the *potential* for detection but also reveals a bottleneck in the current configuration under high load conditions.

4.3 Results of Vulnerability Assessments with OpenVAS

Vulnerability scans were performed using the deployed OpenVAS (GVM) instance (192.168.33.147) against the primary target systems within the ICDE: the Windows Server 2025 Domain Controller (192.168.33.129) and the Windows 11 workstation (192.168.33.130).

1. **Scan Initiation and Completion:** Scans were initiated using the Greenbone Security Assistant (GSA) web interface as described in section 3.2.3. The "Full and fast" scan configuration was typically used. Scans completed successfully, generating detailed reports for each target.

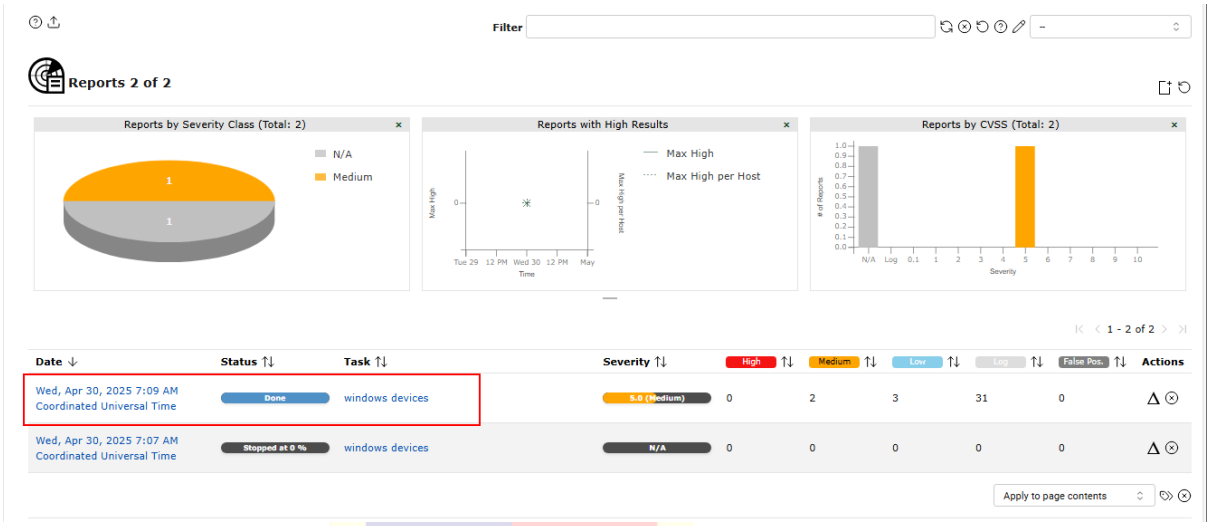

Figure 78 GSA Scans > Reports page showing completed scan reports for the target Ips

2. **Vulnerability Findings Summary:** The scan reports provided a summary of vulnerabilities categorized by severity (High, Medium, Low, Log).

Windows Server 2025 (DC)

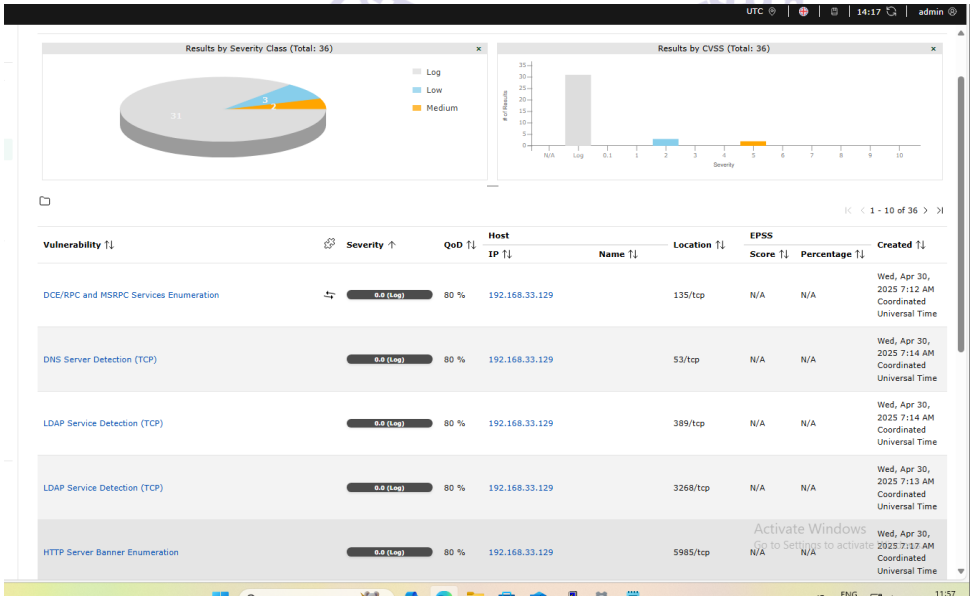

Figure 79 picture showing ip of windows server showing multiple vulnerability present

Windows 11 Workstation:

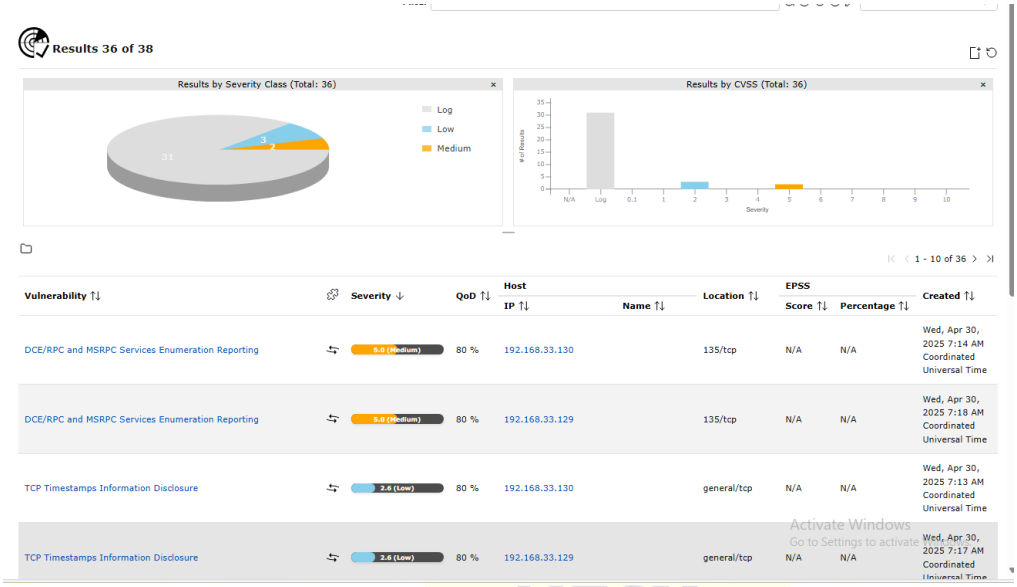

Figure 80 picture showing ip of windows 11 machine with multiple type of severity vulnerability present

Example Vulnerability Details:

Drilling down into the reports allowed for examination of specific vulnerabilities. For instance, a medium-severity finding related to RPC service enumeration was observed on the Windows 11 workstation (192.168.33.130):

- **Vulnerability Name:** DCE/RPC and MSRPC Services Enumeration Reporting
- **CVE ID:** N/A
- **Severity:** 5.0 (Medium)
- **Summary:** Distributed Computing Environment / Remote Procedure Calls (DCE/RPC) or MSRPC services running on the remote host can be enumerated by connecting on port 135 and doing the appropriate queries. The report listed several UUIDs and endpoints accessible via dynamic TCP ports (e.g., 49664, 49665, 49667, 49669) associated with services like lsass (SAM access), Windows Event Log, and the Spooler service.
- **Impact:** An attacker may use this information (list of running services and endpoints) to gain more knowledge about the remote host, potentially identifying specific services to target further.
- **Solution/Remediation:** Mitigation involves filtering incoming traffic to port 135 and the associated dynamic RPC ports where possible, although filtering may impact legitimate functionality depending on the environment's needs.

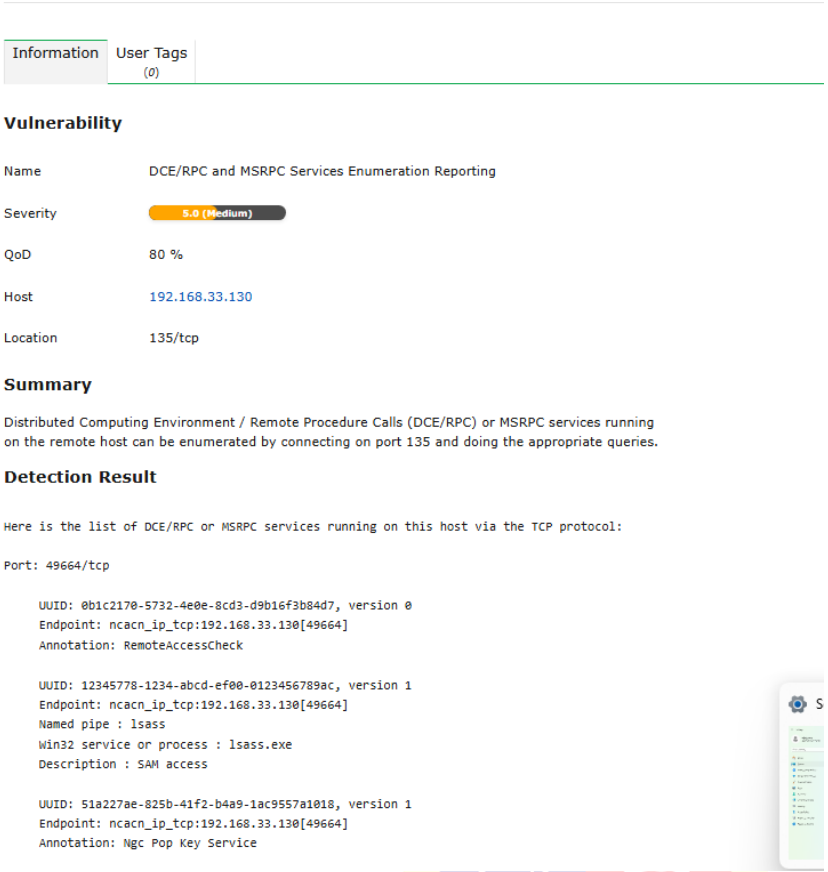

Figure 81 GSA Report showing the details page for the 'DCE/RPC and MSRPC Services Enumeration Reporting' vulnerability on 192.168.33.130

Conclusion:

The OpenVAS scans successfully identified a range of potential security weaknesses across the target systems. The results provide actionable intelligence for prioritizing remediation efforts within the simulated environment. While log collection from OpenVAS itself was handled by the Wazuh agent (capturing system-level logs, not scan results), the primary value demonstrated here is the ability to proactively identify vulnerabilities using the scanner's dedicated interface and reporting capabilities. These findings serve as a baseline security posture assessment for the lab environment.

4.4 Analysis of Splunk Data and Reporting

Splunk served as the central log aggregation platform, receiving Wazuh alerts and logs via the Universal Forwarder installed on the Wazuh Manager. This allowed for searching, analysis, and correlation of security events across the environment using the Search Processing Language (SPL).

Searching for Specific Alerts:

The indexed Wazuh data, forwarded using the specified sourcetype (e.g., wazuh_alerts or wazuh:json) and potentially a dedicated index (e.g., wazuh_forwarded), could be queried effectively. For instance, to investigate successful Windows logons (Wazuh Rule ID 60106) on the target endpoint (agent.name=Target), the following SPL query was used:

- **index="wazuh_forwarded" sourcetype="wazuh_alerts" rule.id=60106 agent.name=Target**

This query successfully retrieved the relevant logon events, including details like the username (data.win.eventdata.targetUserName), source workstation, and logon type (data.win.eventdata.logonType).

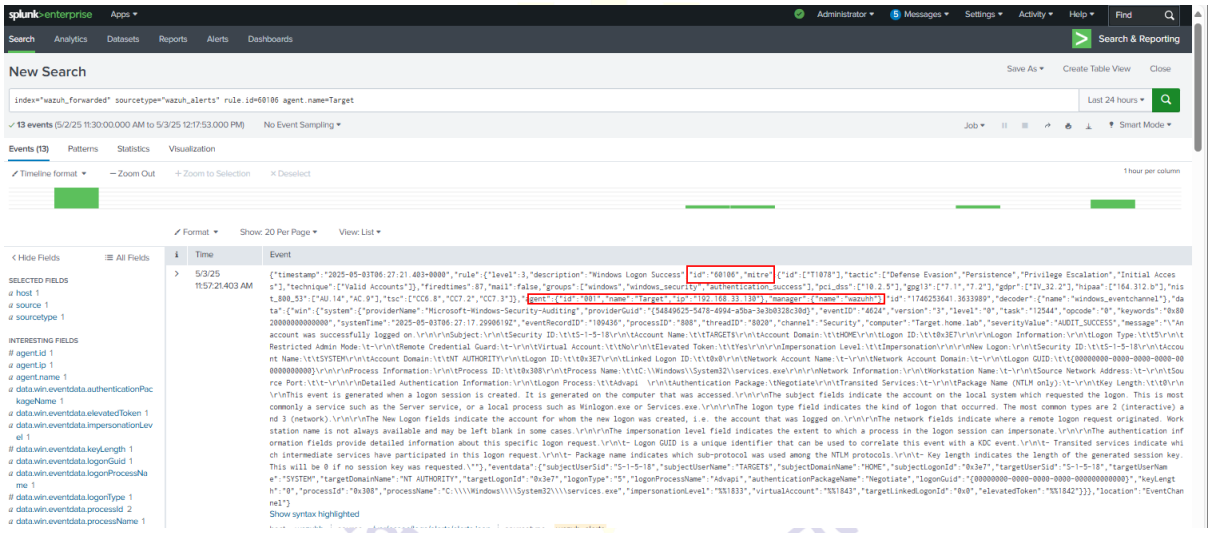

Figure 82 Splunk search results page showing the query for Windows logon success (Rule 60106) on the Target agent and the resulting events

Analyzing Logoff Activity:

Similarly, user logoff events (Wazuh Rule ID 60137) on the Windows Server (agent.name=Windows_Server-2025) could be isolated using:

- **index="wazuh_forwarded" sourcetype="wazuh_alerts" rule.id=60137 agent.name=Windows_Server-2025**

This allowed for tracking user session durations or identifying potentially unusual logoff times.

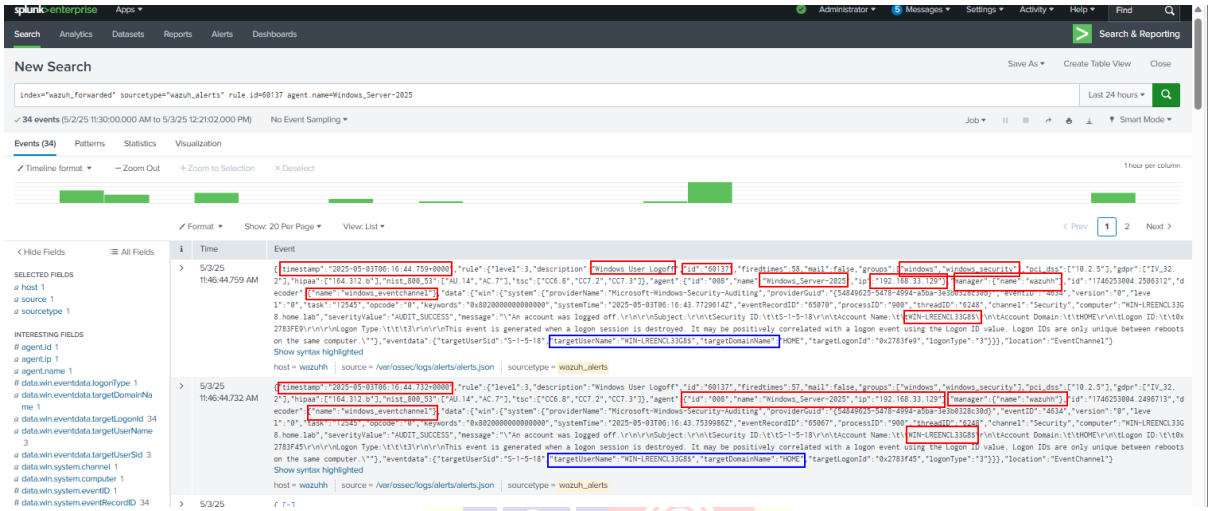

Figure 83 Splunk search results page showing the query for Windows logoff events (Rule 60137) on the Windows_Server-2025 agent

Manual Correlation Example:

Splunk enables manual correlation and verification of alerts received from Wazuh. For instance, after observing the high-severity Cowrie failed login alert (Rule ID 120003, Level 15) in Wazuh (Section 4.1), the analyst sought to confirm its presence in the centralized Splunk logs. A search was performed in Splunk targeting high-severity alerts forwarded from the Wazuh manager:

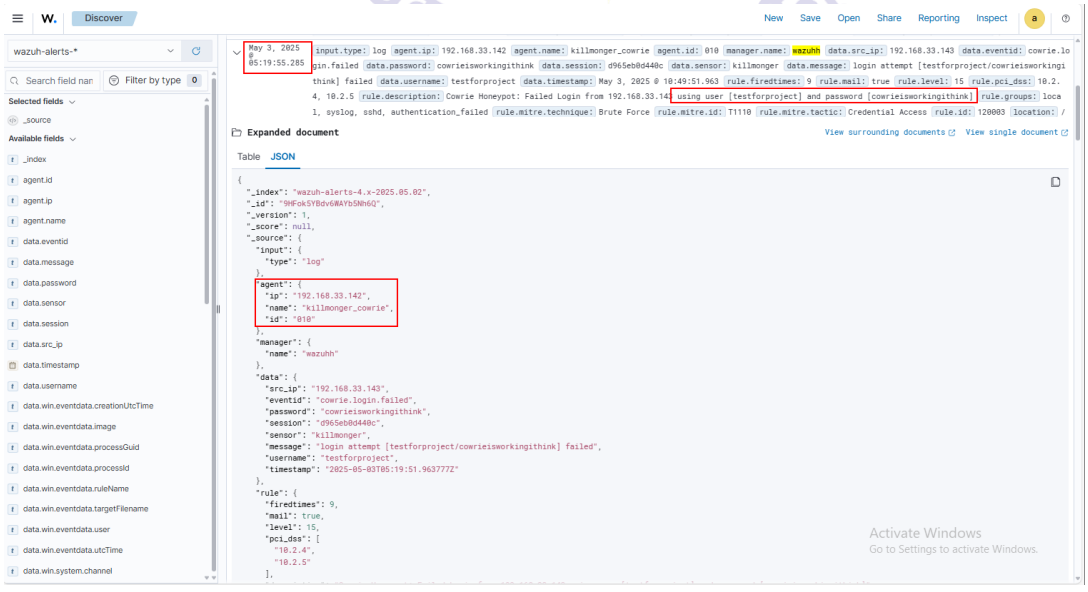

Figure 84 Wazuh showing log of test performed in 4.1 section

- We used **host=”wazuhh” “rule-level”=15**

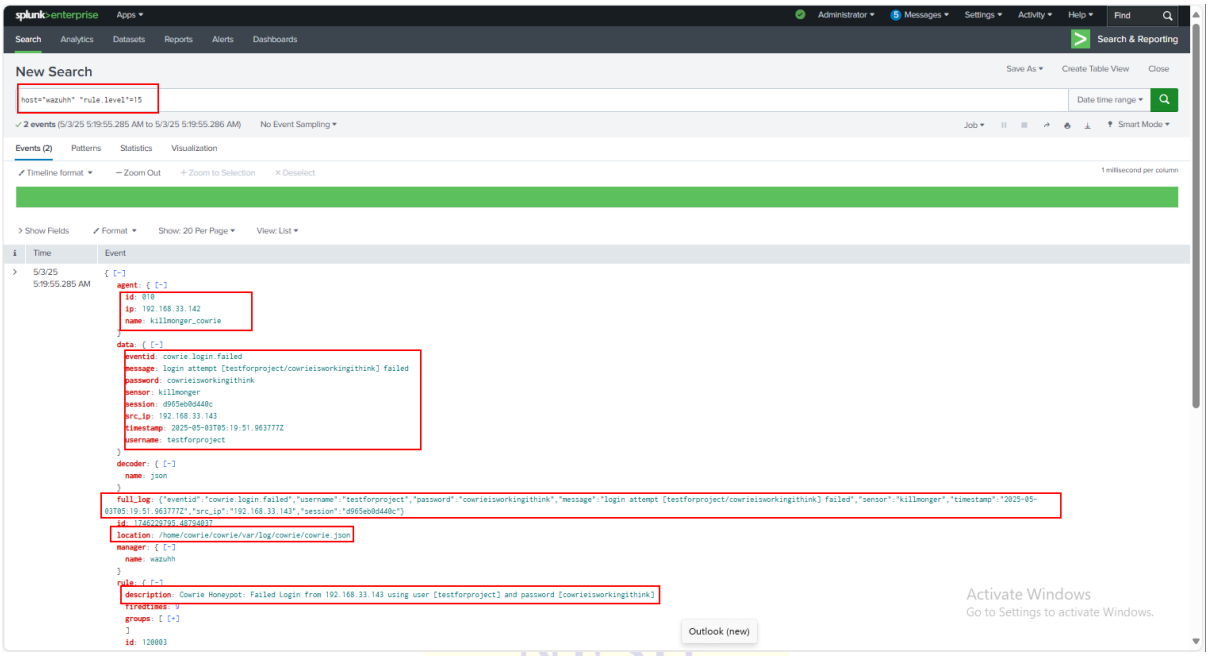

Figure 85 Successfully coorelated the log in splunk

This confirmation allows the analyst to trust the data pipeline and proceed with further investigation within Splunk, potentially pivoting off the confirmed alert's details (like source IP 192.168.33.143 and timestamp 2025-05-03T05:19:51Z) to search for related activities from the attacker across other log sources within the same timeframe.

Conclusion:

Splunk successfully received and indexed alerts forwarded from the Wazuh manager via the Universal Forwarder. It provided essential search capabilities using SPL to retrieve specific alerts based on rule IDs, agent names, severity levels, and other criteria. This centralization proved valuable for verifying alert forwarding, analyzing specific event types like logons/logoffs, and enabling manual correlation by pivoting on key data points found within confirmed Wazuh alerts.

4.5 Discussion of Findings and Key Observations

The project successfully demonstrated the feasibility of integrating diverse open-source security tools into a cohesive virtual SOC environment. Key observations include:

- **Integration Complexity:** While tools offer integration points (APIs, syslog, forwarders, webhooks), careful configuration is required for seamless data flow. Ensuring correct parsing (e.g., JSON format) and reliable transport (e.g., configuring Splunk forwarder, Wazuh agent log collection, Wazuh webhook integration) is crucial.
- **Performance Considerations:** High-volume events, such as those generated by aggressive network scans detected by Suricata, can overwhelm default agent queue sizes, leading to event loss and highlighting the need for potential tuning (<queue_size> in agent ossec.conf) or more targeted detection strategies.
- **Data Correlation:** Splunk proved effective in aggregating logs forwarded from Wazuh via the Universal Forwarder. This allowed for manual correlation across different detection tools using SPL queries, although automated correlation relies heavily on the SIEM/SOAR capabilities. The Wazuh dashboard provided valuable real-time alerting based on agent data from endpoints and other tools like Suricata and Cowrie.
- **SOAR Foundation:** Shuffle was successfully installed and configured to receive alerts via webhook, laying the groundwork for future automation efforts. The ability to ingest alerts and trigger basic notifications (Telegram) is the critical first step towards orchestration. The current setup highlights the potential for streamlining responses once playbooks are developed.
- **Simulation Value:** The lab environment effectively simulated real-world scenarios (brute-force, vulnerability scanning, honeypot interaction, basic endpoint activity). This allowed for practical testing of the detection capabilities of Wazuh, Suricata, and Cowrie, and validated the log collection pipeline into Splunk and the alert pipeline into Shuffle.
- **Visibility Enhancements:** Enabling advanced audit policies and PowerShell logging in Active Directory, along with deploying Sysmon, significantly increased endpoint visibility compared to default Windows logging, providing richer data for analysis in Wazuh and Splunk.

5.0 Conclusion and Future Enhancements

This chapter summarizes the key outcomes of the Integrated Cyber Defense Environment (ICDE) project, evaluates the achievements against the initial objectives, discusses the limitations encountered, and proposes potential aven ues for future development and enhancement.

5.1 Summary of Project Outcomes and Achievements

This project successfully designed and implemented a virtualized cybersecurity laboratory simulating key functions of a modern Security Operations Center (SOC). Hosted within a VMware environment, the ICDE integrates a suite of essential security tools, including Wazuh for SIEM/HIDS capabilities, Suricata for Network Intrusion Detection (NIDS), OpenVAS for vulnerability scanning, Cowrie for honeypot deception, Splunk for centralized log aggregation and analysis, and Shuffle as the foundational SOAR platform. A realistic target environment, featuring an Active Directory domain (home.lab) on Windows Server 2025 and a Windows 11 endpoint, was established to provide context for security monitoring and attack simulation.

Key achievements include:

- Successful deployment and configuration of all core security components and the target environment VMs.
- Establishment of effective log collection pipelines: Wazuh agents collecting endpoint (Windows, Linux), Suricata, and Cowrie logs; Wazuh manager logs forwarded to Splunk via Universal Forwarder.
- Demonstrated detection capabilities for various simulated activities, including network scans (Suricata/Wazuh), failed logins and interactions (Cowrie/Wazuh), and endpoint events (Wazuh HIDS, Sysmon, enhanced Windows logging).
- Successful vulnerability identification using OpenVAS against the target systems.
- Centralized log storage and search functionality provided by Splunk.
- Successful setup of the Shuffle SOAR platform, including integration for ingesting high-severity Wazuh alerts via webhook and demonstrating a basic notification workflow using Telegram.

The resulting ICDE provides a valuable, hands-on platform for practicing threat detection, incident analysis, vulnerability management, and understanding the interplay between different security technologies within a simulated enterprise network.

5.2 Evaluation of the ICDE Against the Project Objectives

The project successfully met most of its initial objectives:

- **Virtualized Infrastructure:** A scalable VMware environment hosting all components was successfully established. (Met)
- **Tool Deployment & Configuration:** All specified tools (Wazuh, Shuffle, Suricata, OpenVAS, Cowrie, Splunk) were deployed and configured to a functional state. (Met)
- **Tool Integration & Data Flow:** Data flows were established: agents to Wazuh, Wazuh logs to Splunk via UF, Wazuh alerts to Shuffle via webhook. (Met)
- **Realistic Target Environment:** An AD domain and Windows endpoint were successfully set up and configured with enhanced logging. (Met)
- **Shuffle SOAR Configuration:** Shuffle was successfully installed, configured for alert ingestion from Wazuh, and demonstrated basic notification capabilities. The objective of *preparing* it for workflow automation. (Met)
- **Demonstrate Capabilities:** Detection of simulated attacks (scans, honeypot interaction) and event management (viewing alerts in Wazuh/Splunk/Shuffle) were demonstrated. (Met)
- **Validation via Kali:** Kali Linux was used effectively to simulate attacks and test the defensive stack's responses. (Met)

The primary area where the objective was only partially met relates to the full implementation of automated workflows within Shuffle, which was identified as a key area for future work. Additionally, performance limitations were observed regarding the Wazuh agent's handling of high-volume Suricata alerts.

5.3 Limitations and Challenges Encountered

Several limitations and challenges were identified during the project:

- **Resource Constraints:** The performance and scale of the lab were limited by the available CPU, RAM, and disk resources of the host machine, impacting the number of VMs and the intensity of simulations possible.

- **Integration Complexity:** Configuring seamless data flow between diverse tools required careful attention to file formats (JSON), transport methods (UF, webhooks), network ports, and specific tool configurations (e.g., Wazuh rules, Splunk inputs).

- **Wazuh Agent Performance:** The Wazuh agent on the Suricata VM experienced queue flooding during high-volume Nmap scans, indicating a bottleneck in processing and forwarding the large number of NIDS alerts generated. This resulted in the loss of specific Suricata alerts at the manager level during these intense scans.

- **SOAR Workflow Implementation:** While Shuffle was successfully set up for alert ingestion and basic notification, the development, testing, and refinement of comprehensive, automated response playbooks require significant additional time and effort.

- **Tool Version Compatibility:** Issues were encountered with the Splunk App for Wazuh due to version incompatibility with the deployed Splunk Enterprise version, preventing its use for pre-built dashboards.

- **Manual Correlation:** While Splunk provides powerful search capabilities, correlating complex events across multiple data sources often still required manual analysis and pivoting by the analyst in the absence of fully developed SIEM correlation rules or advanced Splunk applications.

5.4 Future Work and Potential Enhancements

The current ICDE provides a solid foundation, but numerous enhancements could further increase its value and realism:

5.4.1 Integration of Additional Security Tools:

- **Threat Intelligence Platform (TIP):** Integrate MISP (Malware Information Sharing Platform) or similar tools. Shuffle could query the TIP to enrich alerts with IOCs (Indicators of Compromise).
- **Endpoint Detection and Response (EDR):** Deploy an open-source EDR solution (like Wazuh's extended capabilities or Velociraptor) for deeper endpoint visibility and response actions.
- **Network Forensics:** Add tools like Zeek or Arkime for deeper network traffic analysis and session reconstruction, potentially feeding logs into Splunk/Wazuh.
- **Firewall Integration:** Include a virtual firewall (pfSense, OPNsense) and integrate it with Shuffle for automated IP blocking actions.

5.4.2 Cloud Integration and Deployment:

- Explore migrating the entire lab or specific resource-intensive components (like Splunk or Wazuh Indexer) to a cloud platform (AWS, Azure, GCP) to leverage greater scalability and potentially reduce local hardware requirements.
- Investigate using cloud-native security services alongside the lab components.

5.4.3 Enhancing Automation and Response Capabilities (Shuffle):

- **Develop Core Playbooks:** Build and test specific Shuffle workflows for common scenarios:
    - *Malware Containment:* Isolate infected host via Wazuh active response or firewall integration.
    - *Phishing Analysis:* Extract URLs/attachments, query threat intelligence (VirusTotal, URLscan.io), detonate in a sandbox (if available).
    - *IP Blocking:* Automatically add malicious IPs identified by Wazuh/Suricata/Cowrie to a firewall blocklist.
    - *User Account Actions:* Disable potentially compromised user accounts in Active Directory via PowerShell/API calls triggered by Shuffle.
- **Bi-directional Communication:** Configure Shuffle to not only ingest alerts but also update Wazuh (e.g., close alerts) or interact with ticketing systems (e.g., create Jira/ServiceNow tickets).
- **Expand Shuffle Apps:** Integrate more Shuffle apps for threat intelligence feeds, sandbox analysis, and communication platforms.

5.4.4 Developing Advanced Training Scenarios:

- **Multi-Stage Attack Simulations:** Design scenarios mimicking APT kill chains, requiring correlation across multiple tools (NIDS, HIDS, Logs).
- **Ransomware Simulation:** Safely simulate ransomware deployment and test detection/response capabilities.
- **Web Application Attacks:** Add a vulnerable web server (e.g., DVWA, Juice Shop) and simulate common attacks (SQLi, XSS) to test WAF/NIDS/HIDS detection.
- **Performance Tuning:** Investigate and implement solutions for the Wazuh agent flooding issue, such as adjusting agent queue size/EPS settings or implementing rule/alert rate-limiting in Suricata.
- **User Guides & Exercises:** Create specific lab exercises and user guides to direct training activities within the ICDE.

By addressing these future work items, the ICDE can evolve into an even more powerful and comprehensive platform for cybersecurity education, training, and experimentation.

6.0 References

Official Documentation & Online Resources:

1. Wazuh Inc. (2024). *Wazuh Documentation*. https://documentation.wazuh.com/current/index.html [Accessed Online, Last Accessed On: 03.05.2025]
2. Splunk Inc. (2024). *Splunk Enterprise Documentation*. https://docs.splunk.com/Documentation/Splunk [Accessed Online, Last Accessed On: 03.05.2025]
3. Splunk Inc. (2024). *Splunk Universal Forwarder Manual*. https://docs.splunk.com/Documentation/Forwarder [Accessed Online, Last Accessed On: 03.05.2025]
4. Open Information Security Foundation (OISF). (2024). *Suricata User Guide*. https://suricata.readthedocs.io/en/latest/ [Accessed Online, Last Accessed On: 03.05.2025]
5. Greenbone Networks GmbH. (2024). *Greenbone Community Documentation*. https://docs.greenbone.net/ [Accessed Online, Last Accessed On: 03.05.2025]
6. Cowrie Development Team. (2024). *Cowrie SSH/Telnet Honeypot Documentation*. https://cowrie.readthedocs.io/en/latest/ [Accessed Online, Last Accessed On: 03.05.2025]
7. Shuffle Automation Community. (2024). *Shuffle Documentation*. https://shuffler.io/docs [Accessed Online, Last Accessed On: 03.05.2025]
8. Microsoft Learn. (2024). *Active Directory Domain Services Overview*. https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview [Accessed Online, Last Accessed On: 03.05.2025]
9. Microsoft Learn. (2024). *Advanced Security Audit Policy Settings*. https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/advanced-security-audit-policy-settings [Accessed Online, Last Accessed On: 03.05.2025]
10. Microsoft Learn. (2024). *About PowerShell Logging*. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_logging [Accessed Online, Last Accessed On: 03.05.2025]
11. Russinovich, M., & Margosis, A. (2023). *Sysmon - Windows Sysinternals*. Microsoft Learn. https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon [Accessed Online, Last Accessed On: 03.05.2025]
12. Hartong, O. (2024). *Sysmon-Modular Configuration*. GitHub Repository. https://github.com/olafhartong/sysmon-modular [Accessed Online, Last Accessed On: 03.05.2025]

13. VMware Inc. (2024). *VMware Workstation Pro Documentation*. https://docs.vmware.com/en/VMware-Workstation-Pro/index.html [Accessed Online, Last Accessed On: 03.05.2025]

14. ET Labs / Proofpoint. (2024). *Emerging Threats Open Ruleset*. https://rules.emergingthreats.net/open/ [Accessed Online, Last Accessed On: 03.05.2025]

15. MITRE Corporation. (2024). *MITRE ATT&CK Framework*. https://attack.mitre.org/ [Accessed Online, Last Accessed On: 03.05.2025]

Books, Papers, and Articles (APA Style):

16. Kent, K., & Souppaya, M. (2006). *Guide to Computer Security Log Management*. NIST Special Publication 800-92. National Institute of Standards and Technology.

17. Scarfone, K., & Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS)*. NIST Special Publication 800-94. National Institute of Standards and Technology.

18. Sanders, C., & Smith, J. (2014). *Applied Network Security Monitoring: Collection, Detection, and Analysis*. Syngress.

19. Chuvakin, A., Schmidt, K., & Philips, C. (2013). *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*. Syngress.

20. Rehmani, S. S., & Saleem, K. (Eds.). (2019). *Security Operations Centers: Building, Operating, and Maintaining your SOC*. CRC Press.

21. Provos, N. (2004). *A Virtual Honeypot Framework*. Proceedings of the 13th USENIX Security Symposium.

22. Spitzner, L. (2003). *Honeypots: Tracking Hackers*. Addison-Wesley Professional.

23. Bhatt, S., Manadhata, P. K., & Zomlot, L. (2014). The operational role of security information and event management systems. *IEEE Security & Privacy*, *12*(5), 35-41.

24. Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). *Computer Security Incident Handling Guide*. NIST Special Publication 800-61 Rev. 2. National Institute of Standards and Technology.

25. Tounsi, W., & Frikha, M. (2018). Security orchestration, automation and response (SOAR): A survey and a taxonomy. *Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018)*.

26. Metoyer, R. A., & Parrish, A. (2007). Using virtualization software to enhance computer security education. *Journal of Computing Sciences in Colleges*, *22*(5), 183-189.

27. Wazuh Team. (2023). *Integrating Wazuh with Splunk*. Wazuh Blog. https://wazuh.com/blog/integrating-wazuh-with-splunk/ [Accessed Online, Last Accessed On: 03.05.2025]