A PROJECT REPORT ENTITLED:

“ DESIGN AND IMPLEMENTATION OF A VIRTUAL NETWORK SECURITY MONITORING LAB USING PFSENSE FIREWALL WAZUH SIEM AND HONEYPOT IMPLEMENTATION”

BY

DAMOAH BASHIRU

22nd SEPTEMBER, 2024

ABSTRACT

This report documents the end-to-end implementation of a blue-team security monitoring lab that uses pfSense (as a perimeter firewall rule bound with gateway) with Wazuh SIEM (for endpoint data logging, file integrity monitoring, alerting and event prompting) and also to demosntrate the implementation of honeypot. The lab was built in VMware with Ubuntu host running the Wazuh stack and manager dashboard, a Windows 10 endpoint with the Wazuh Agent and also acting as victim, and an attacker VM (Kali) for controlled testing and attack simulation.

I configured pfSense with separated WAN (Bridged/NAT) and LAN (Host-Only/Internal) interfaces and also created a virtual private network with the pfsense rule governing the network, enabled DHCP on the LAN, and forwarded logs to the SIEM. On the Ubuntu host, I installed Wazuh Manager, Elasticsearch, Kibana, and Filebeat, verified that the Wazuh dashboard was reachable (served by Node on :443 on my setup), and onboarded the Windows 10 agent using the manager IP (192.168.2.1). I then simulated attacker behaviour (e.g., Nmap scans) and harmless user actions (file and registry changes). Wazuh successfully generated alerts (medium/low severities), and pfSense registered connection activity to the endpoint (192.168.2.130).

The build demonstrates how a firewall + SIEM pair creates defence-in-depth: traffic controls at the edge (pfSense) and rich endpoint visibility and analytics (Wazuh). The report closes with actionable recommendations for hardening and scaling.

TABLE OF CONTENTS

10. LIST OF TABLES

INTRODUCTION

1.1 Background

Hands-on monitoring is the fastest way to understand what “defense-in-depth” looks like in practice. In this project, I built a self-contained blue-team lab: pfSense for traffic control/logging, and Wazuh for endpoint telemetry, correlation, and alerting. The environment is isolated (Host-Only/Internal LAN) but can optionally egress to the Internet through pf Sense’s bridged/NAT WAN.

1.2 Problem Statement

Most student environments lack a safe place to observe real security events end-to-end. Without a lab, it’s hard to connect the dots between a port scan, an endpoint change, and the alert that lands in a SIEM.

1.3 Objectives

- Build a virtual lab with pfSense (firewall) and Wazuh SIEM (manager + dashboards).
- Onboard a Windows 10 endpoint and validate FIM/registry telemetry.
- Generate controlled attacker behaviours (e.g., Nmap scans) and verify alerts & logs.

1.4 Scope and Limitations

The lab runs on a single workstation using VMware networks. It is not connected to any production environment. IDS like Suricata is optional and not the core focus here.

1.5 Report Organisation

I start with related work, then explain the exact lab architecture, describe what I implemented step-by-step, and finally present the results and analysis.

LITERATURE REVIEW

1.6 Firewalls and Segmentation (pfSense)

Firewalls enforce policy at network boundaries. pfSense (FreeBSD-based) provides stateful filtering, NAT, DHCP, DNS forwarder/Resolver, VPN, and logging, making it ideal for a compact teaching lab.

1.7 SIEM (Wazuh) and Endpoint Telemetry

SIEMs aggregate events, normalise them, and produce alerts. Wazuh adds agents for FIM, registry auditing, policy monitoring, and MITRE ATT&CK mapping. Kibana dashboards help visualise alerts and trends.

1.8 Related Academic Labs

Academic SOC/blue-team labs generally isolate traffic (Host-Only/Internal) and then pinhole Internet access via a gateway to control risk. This lab follows that pattern.

2. METHODOLOGY AND LAB ARCHITECTURE.

2.1 Lab Requirements & Folders

- Host: Ubuntu (8GB+ RAM, 4 cores recommended).
- VMs: pfSense (firewall), Windows 10 (endpoint), Kali (attacker).
- Folders: ISOs/, Configs/, Screenshots/, Docs/.

2.2 Virtual Networking & VPN Considerations

- Host-Only/Internal keeps lab traffic off the real LAN.

- WAN (Bridged/NAT) on pfSense controls any outside access.

- Using a VPN on the host will change your host's default route/DNS and can impact VM Internet reachability. In this lab, the LAN remained Internal/Host-Only, with pfSense WAN handling egress.

2.3 Topology and IP Plan

pfSense LAN serves 192.168.2.0/24 (DHCP enabled). The Ubuntu host (Wazuh) sits at 192.168.2.1; Windows 10 received 192.168.2.130; Kali is another LAN IP in the same range.

Table 1. IP Plan and Interfaces

Node	Interface	Mode	IP / Notes
pfSense (WAN)	em0	Bridged/NAT	DHCP from upstream
pfSense (LAN)	em1	Host-Only/Internal	192.168.2.1/24 (GW)
Ubuntu (Wazuh)	ensX	Host-Only/Internal	192.168.2.1
Windows 10	ethX	Host-Only/Internal	192.168.2.130 (DHCP)
Kali	ethX	Host-Only/Internal	192.168.2.x (DHCP)

IMPLEMENTATION

2.4 pfSense Installation and Core Services

3. Installer and WebGUI Access. I installed pfSense from ISO and assigned WAN = Bridged/NAT, LAN = Internal.

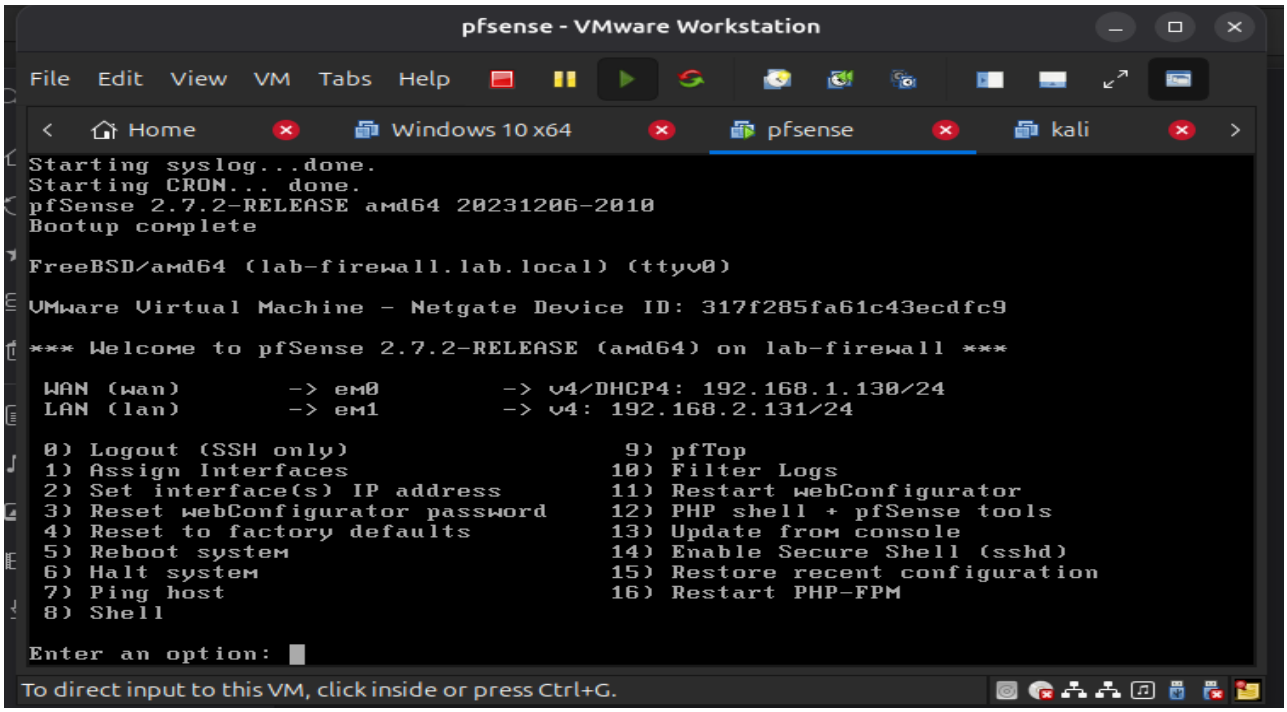


Figure 4.1.1:Pfsense interface showing all the interfaces

I accessed the WebGUI via the LAN IP and logged in with the default admin credentials (then changed the password).

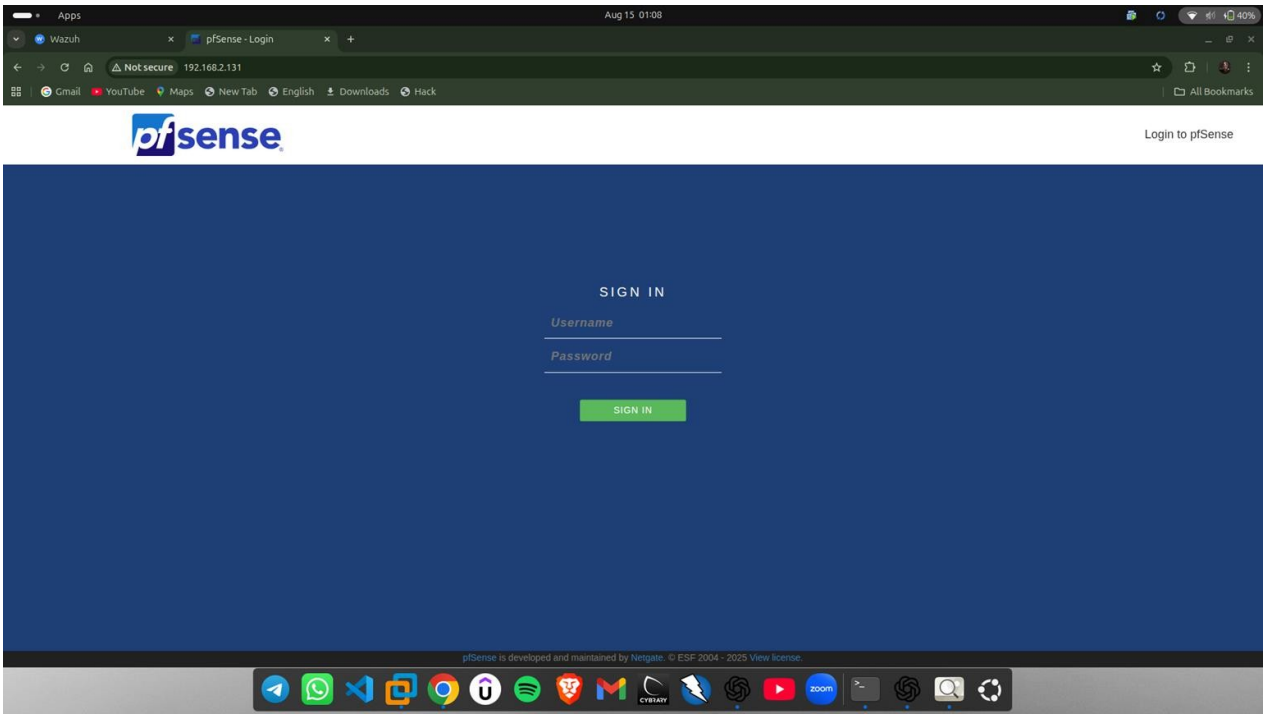


Figure 4.1.2:Web Graphical Interface of Pfsense

DHCP on LAN. I enabled the DHCP server on LAN to make client onboarding painless and to keep all lab addressing consistent.

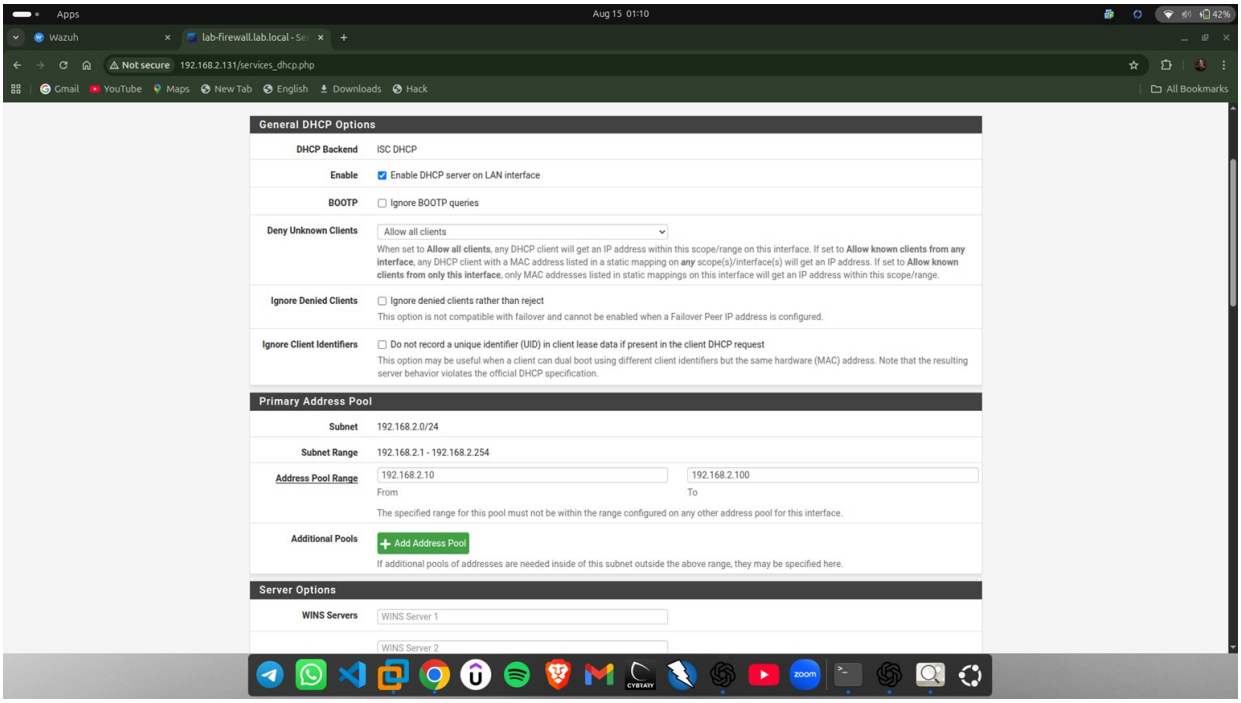
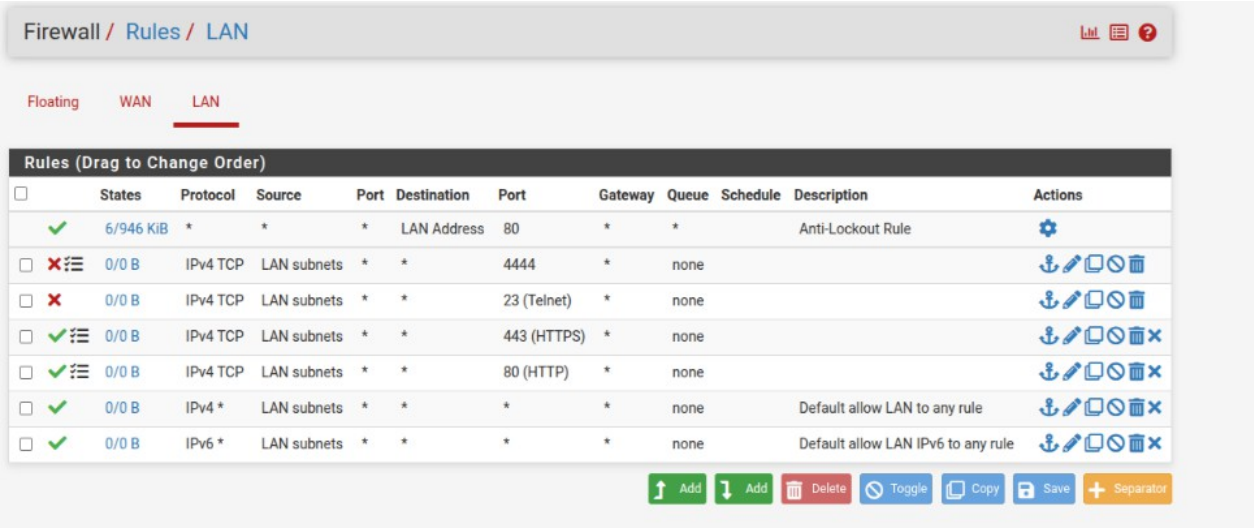


Figure 4.1.3: dhcp server on lan

Initial Allow Rules (Bring-up). To get everything talking during setup, I allowed LAN → Any

and enabled logging on the rule. Later, this can be tightened to least privilege.

*Figure 4.1.4: Rules being set for LAN*Remote Syslog to SIEM. I configured Status → System Logs →



Settings → Remote Syslog Servers to point to the Ubuntu host (192.168.2.1) on UDP/514, so pfSense events could be consumed downstream.

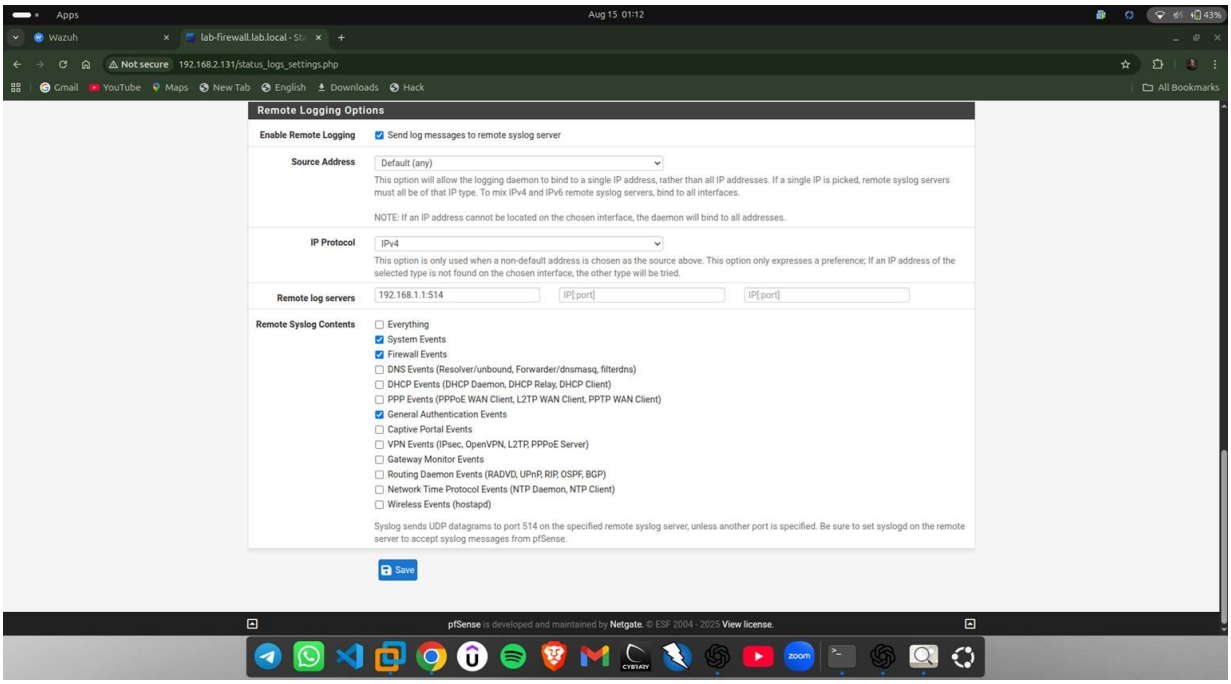


Figure 4.1.5:Remote Syslog to SIEM

Table 4.1.6. pfSense LAN Rules (Initial Bring-up Profile)

#	Action	Proto	Source	Destination	Ports	Log
1	Pass	Any	LAN net	Any	Any	yes

3.1 Wazuh SIEM on Ubuntu Host (Manager, ES, Kibana, Filebeat)

Manager Installation. On Ubuntu, I added the Wazuh apt key & repo, then installed wazuh-manager.

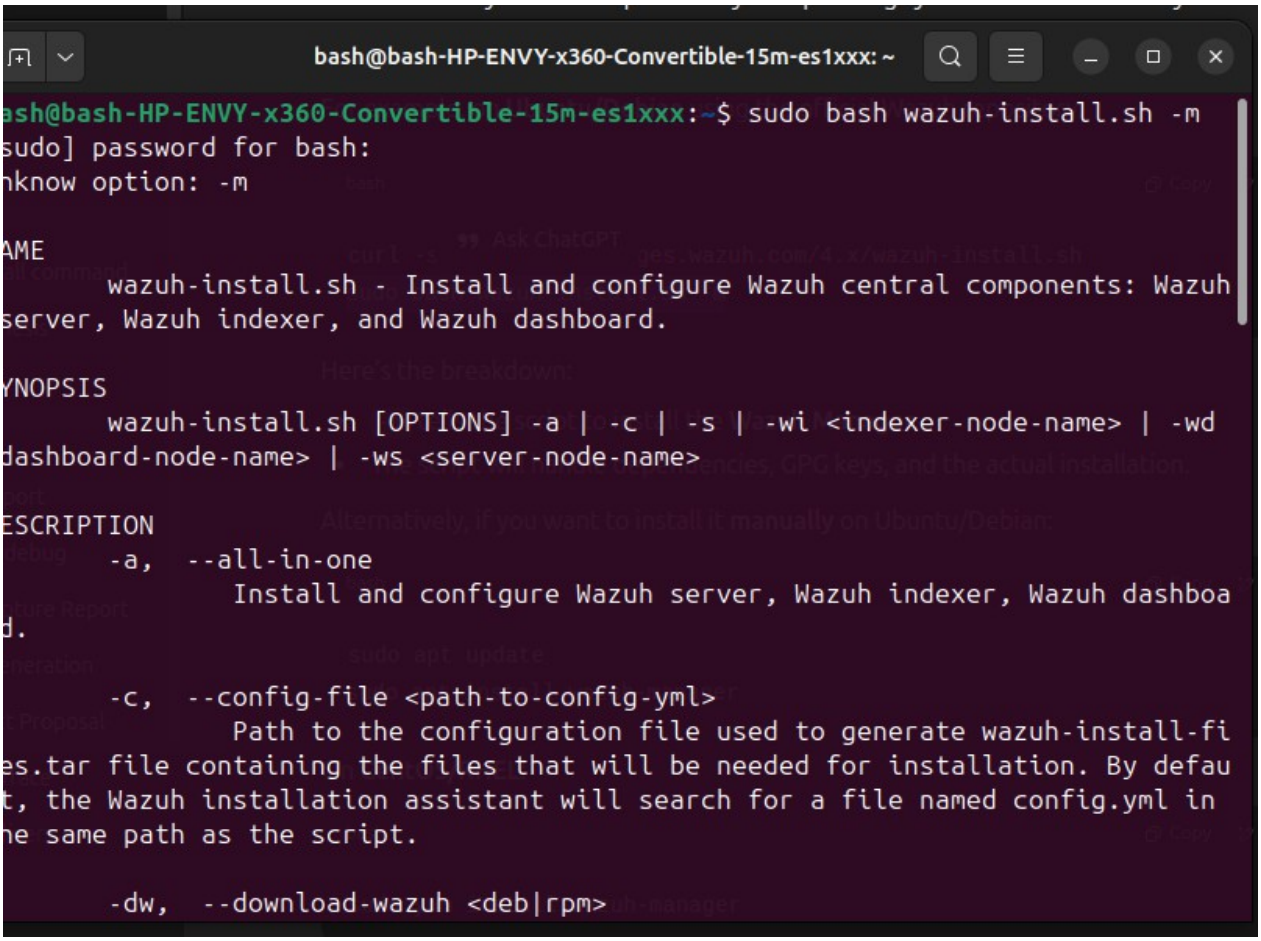


Figure 4.2.1 :Wazuh Installation

Dashboard Availability. Kibana/Wazuh UI was reachable; on my machine the Node process was listening on :443, verified with:

Figure 4.2.2: Checking Wazuh statuses

Wazuh Overview. After components were running, the Wazuh Overview showed the stack and health status.

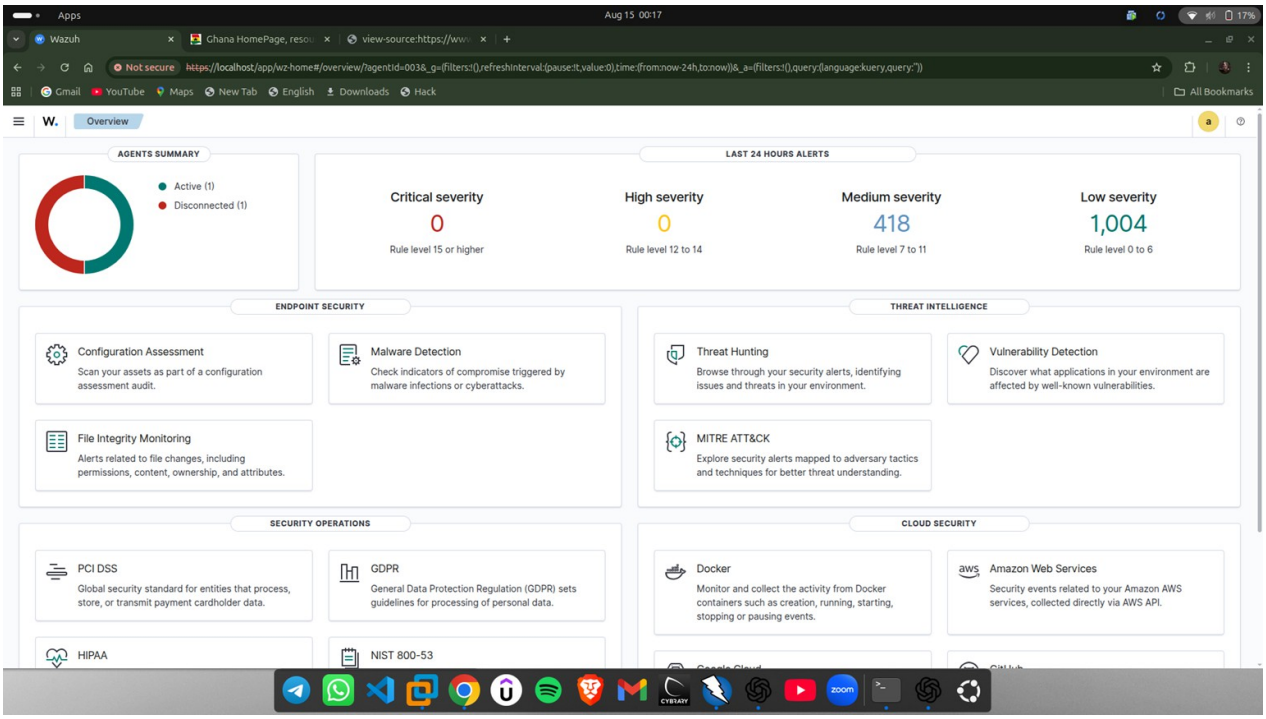


Figure 4.2.3:Wazuh dashboard showing all agent availability

3.2 Windows 10 Agent Onboarding and Manager Authentication

Agent Install. I installed Wazuh Agent on the Windows endpoint, entering the Manager IP = 192.168.2.1 during setup.

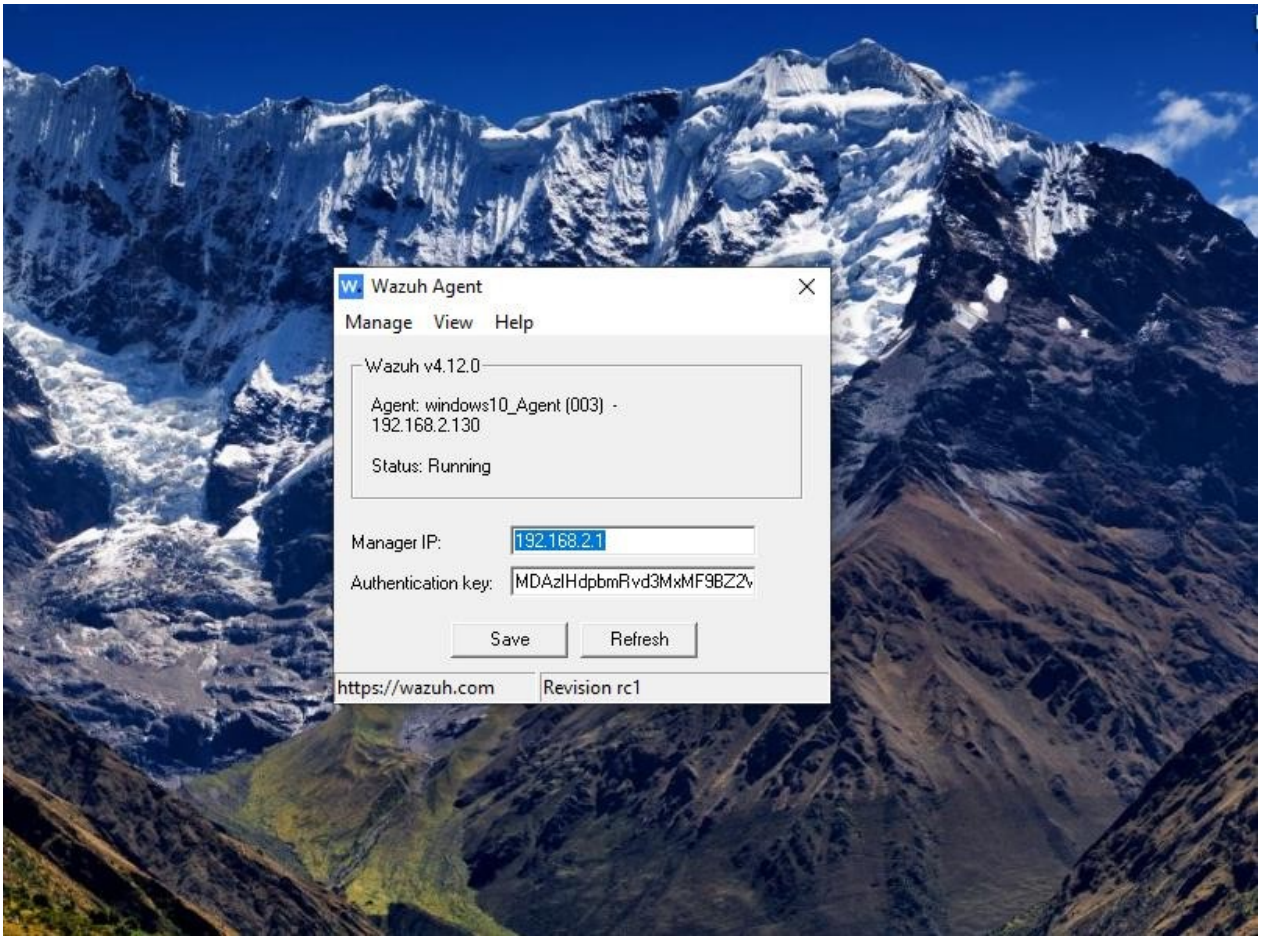


Figure 4.3.1: Wazuh Agent installation

Agent Manager Config. I verified settings with the Windows Agent Manager (agent name, server, and connectivity).

Manager-Side Verification. On Ubuntu, I confirmed registration and status:
sudo /var/ossec/bin/agent_control -l

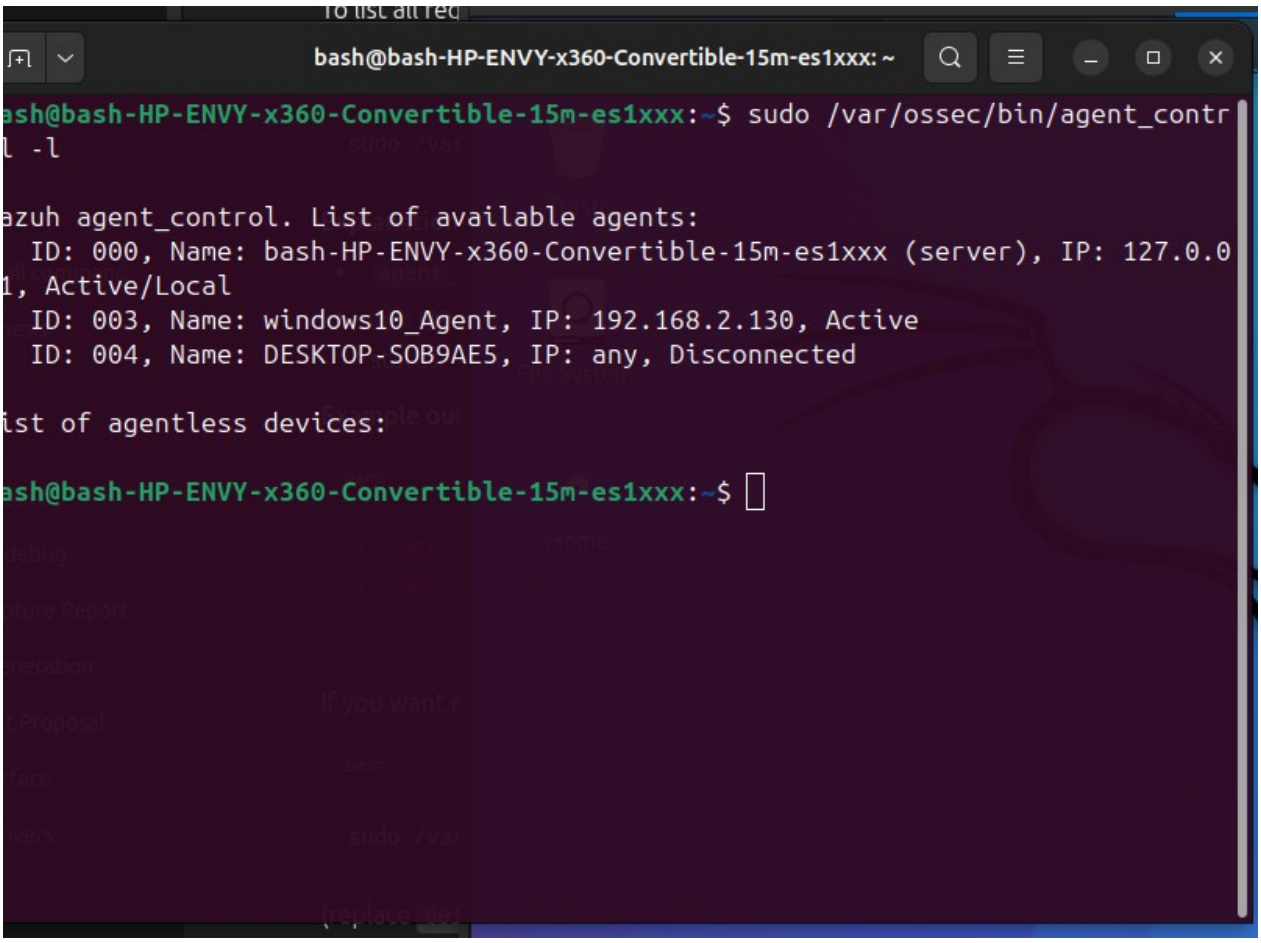


Figure 4.3.2:Wazuh agent verification

Dashboard View of Agent. The new Windows endpoint appeared as Active under Wazuh → Agents.

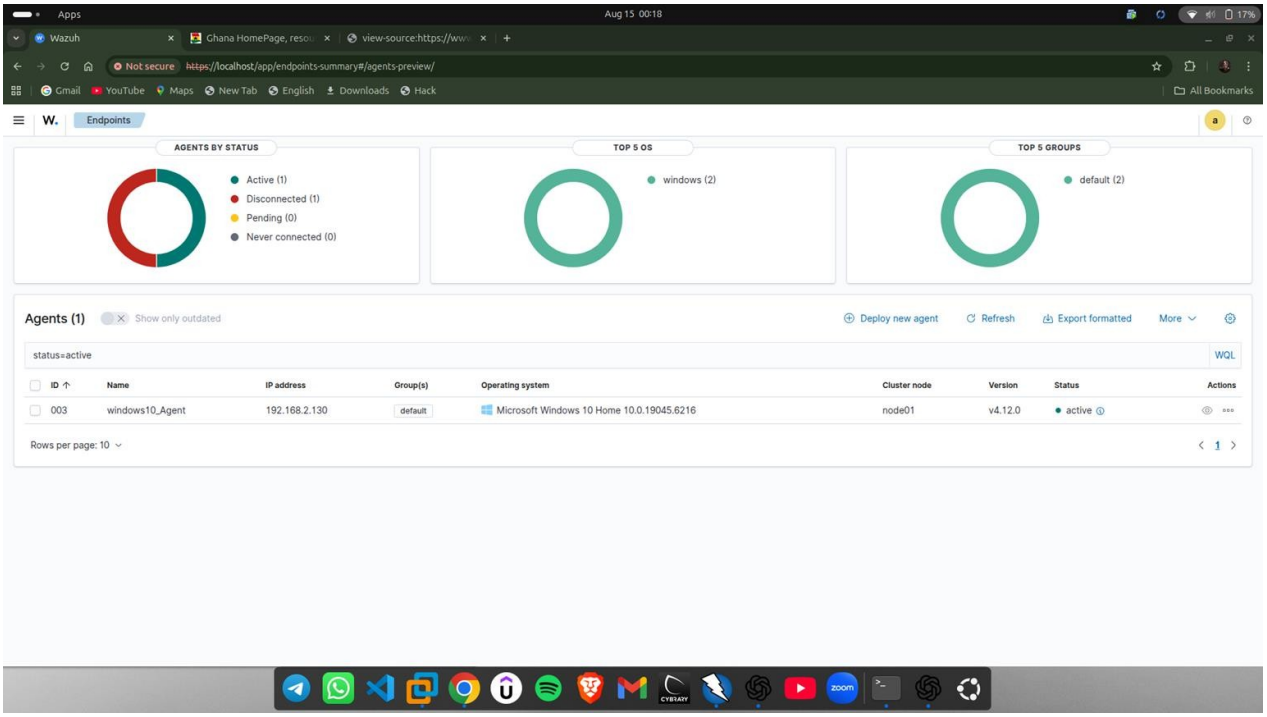


Figure 4.3.3:Dashboard view of agents

All created agents were listed above

VERIFICATION,TESTING AND RESULTS

3.3 File Integrity Monitoring (FIM) and Registry Alerts

FIM Test. I modified test files on the Windows endpoint to trigger integrity checks. In the dashboard, I observed “Integrity checksum changed” alerts with associated file paths and hashes.

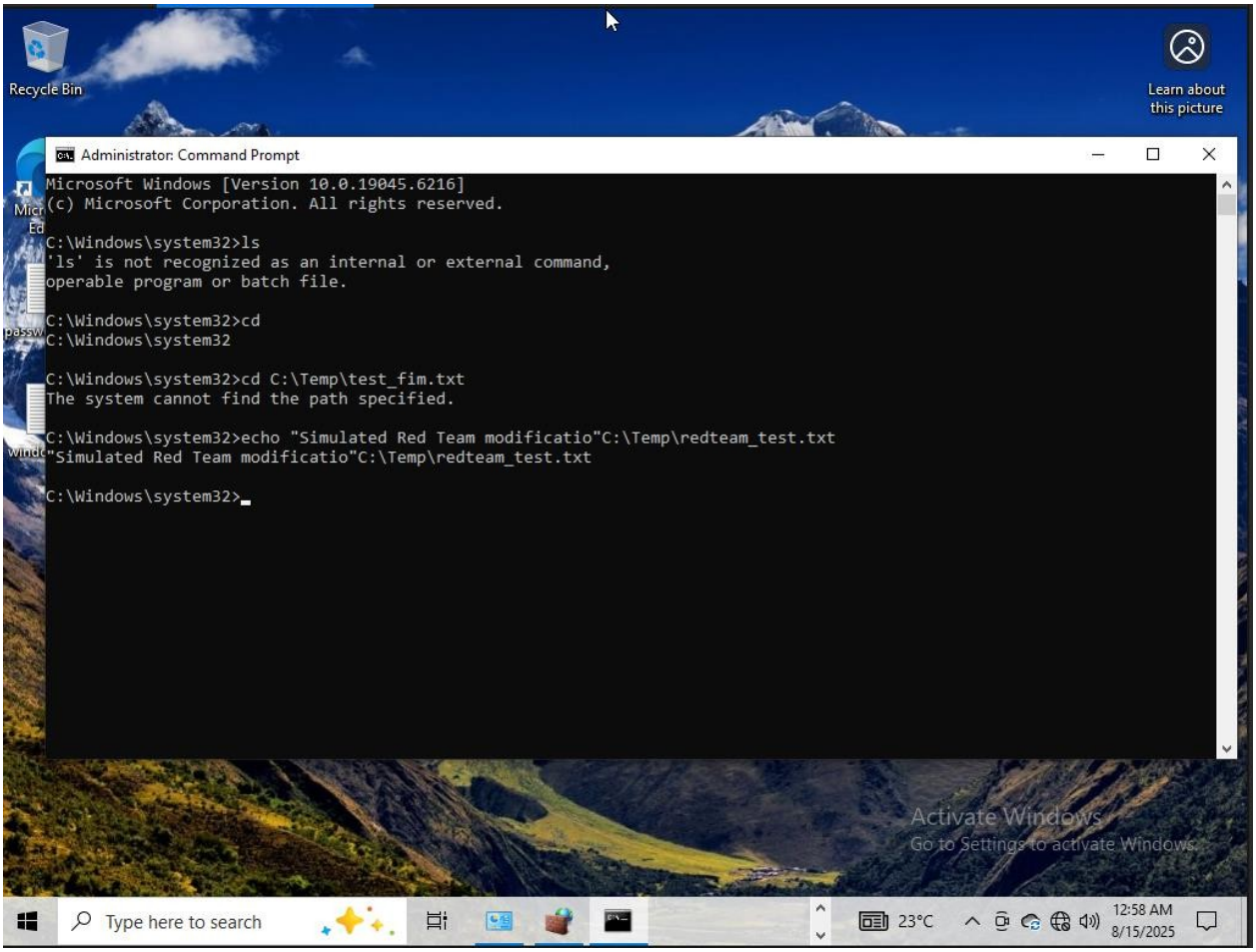


Figure 5.1.1: Initialising alerts with filepath and hashes

Registry Test. I adjusted selected registry keys (harmless settings) to validate Windows registry monitoring. Wazuh created alerts referencing the changed keys and showed MITRE ATT&CK context (where available).

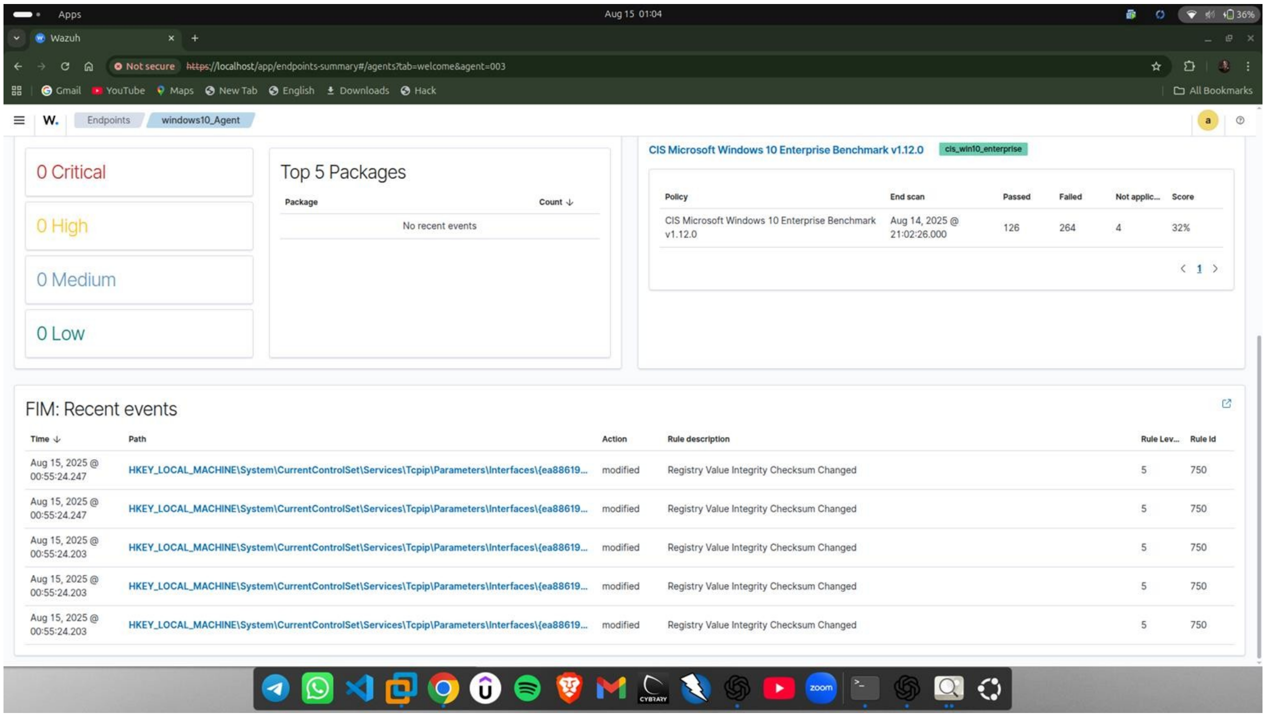


Figure 5.1.2:Event logs

What I looked for in the alerts (examples):

- Rule names/tags (e.g., fim, windows_registry)
- Old vs. new hash for files (SHA1/SHA256)
- Registry hive/path and value type changes
- Agent name and IP (mapping to the endpoint)

3.4 Reconnaissance from Kali (Nmap) and Network Activity

From Kali, I scanned the Windows endpoint 192.168.2.130:

nmap -sS 192.168.2.130

Observed open ports: 135/tcp, 139/tcp, 445/tcp (classic Windows services: MSRPC, NetBIOS-SSN, SMB).

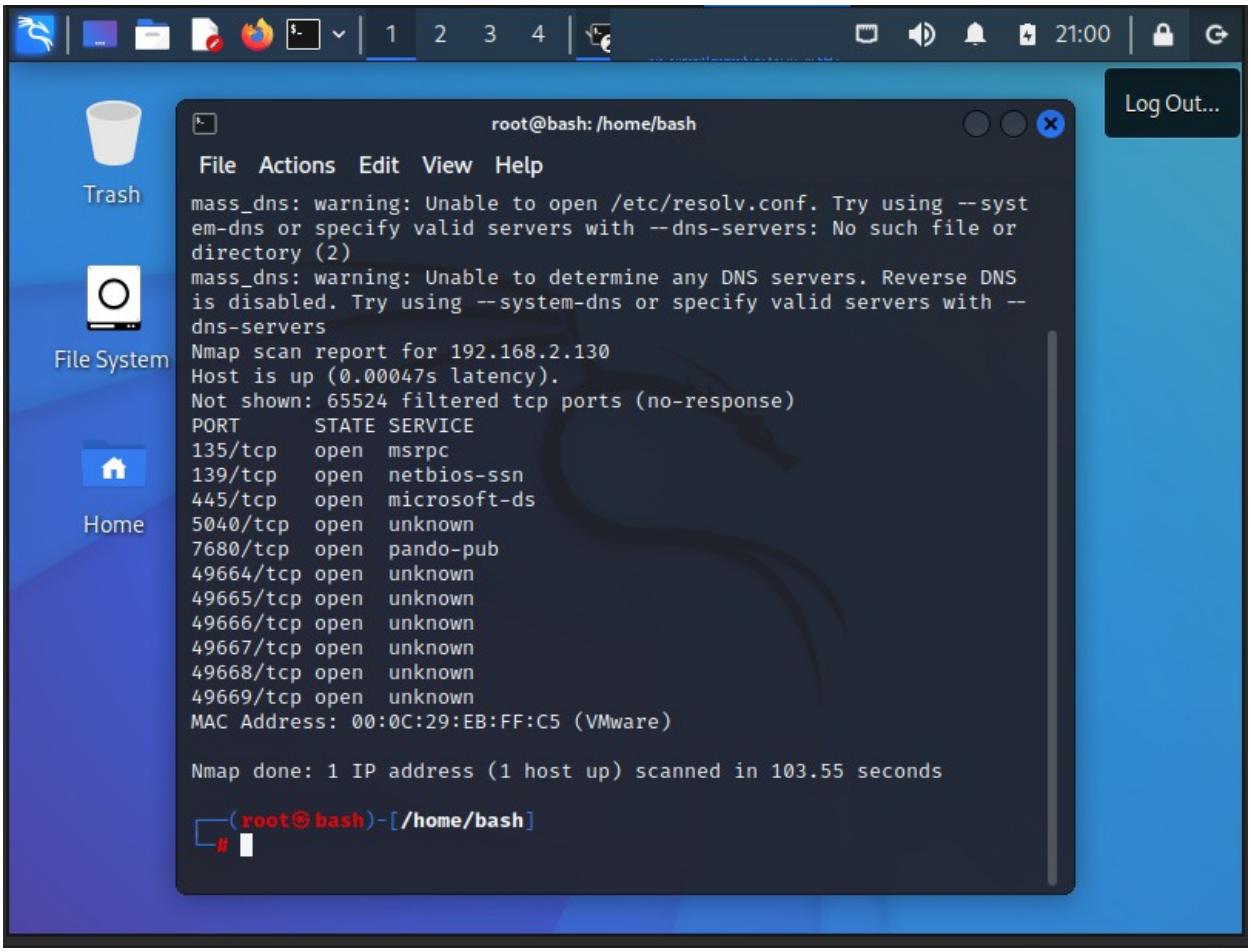


Figure 5.2.1: Scanning windows endpoints

In Wazuh, I reviewed network-related events around the scan timeframe (agent perspective and management logs).

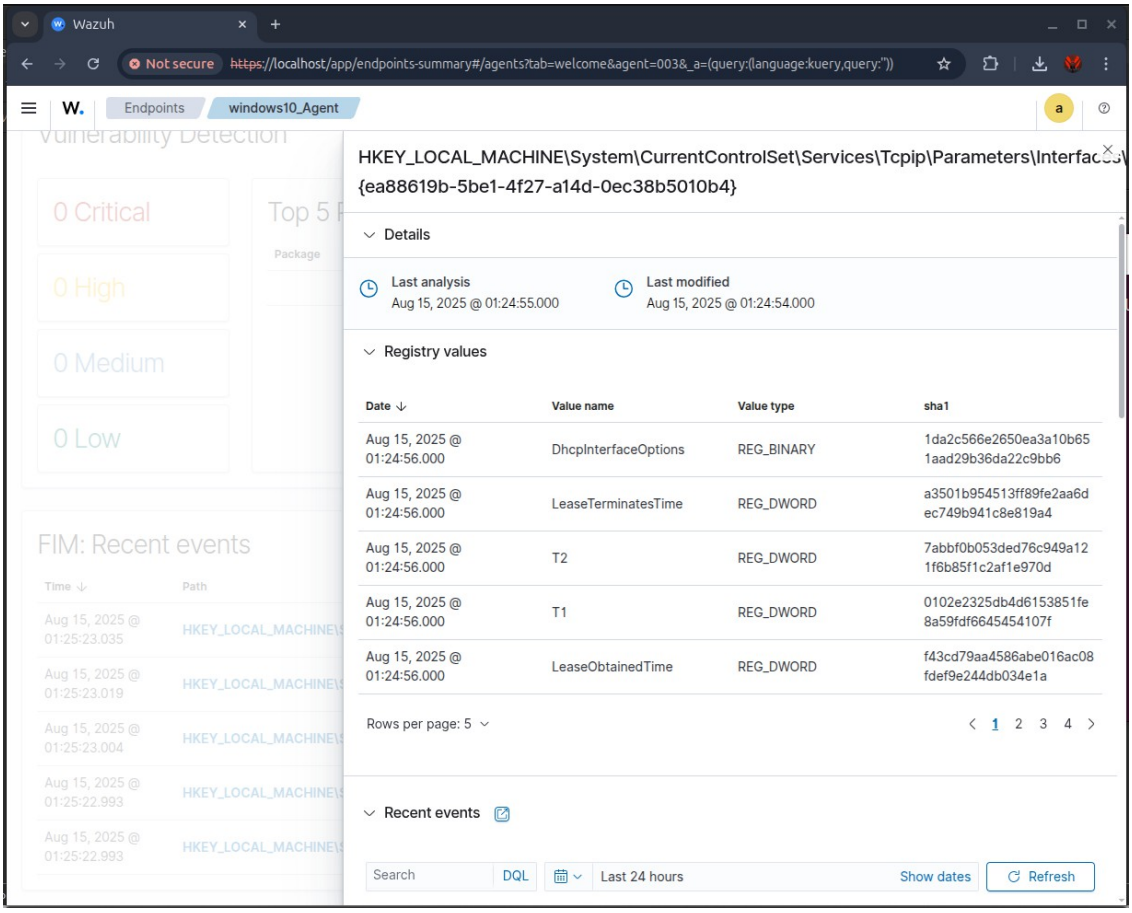


Figure 5.2.2: Wazuh Dashboards, Compliance, and Event Statistics

I explored Compliance/Policy and Event Statistics dashboards to understand which categories were most active (FIM, system, security, etc.), and the distribution across time.

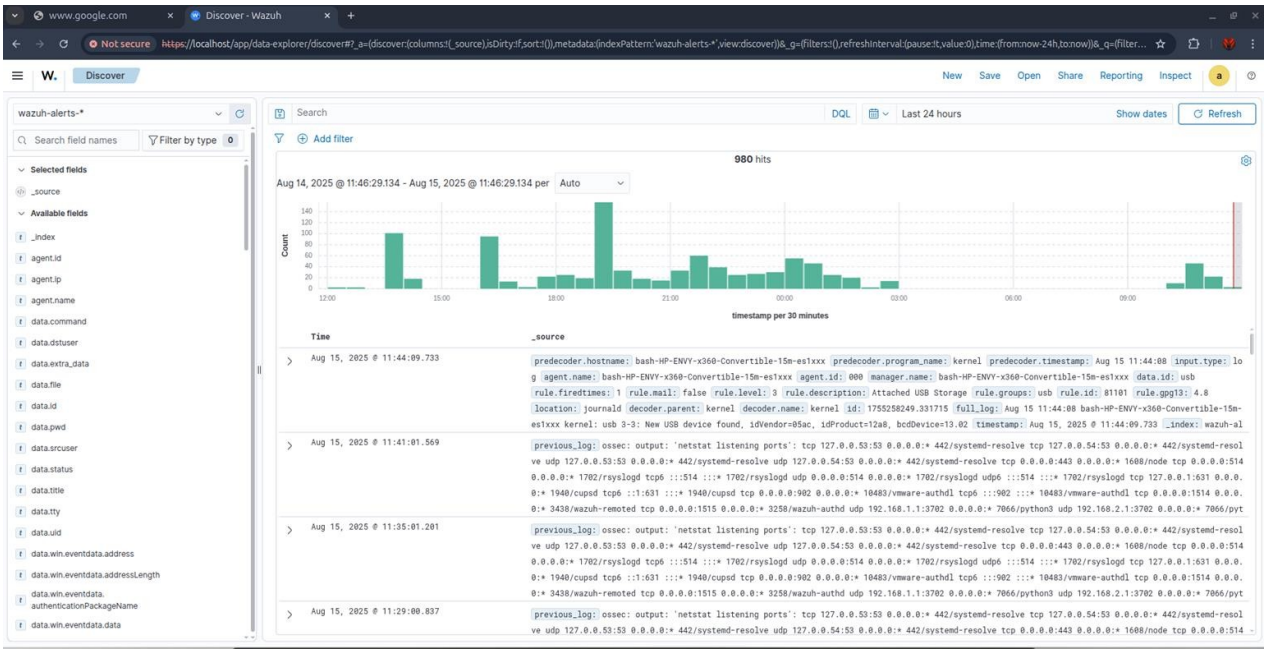


Figure 5.2.3: compliance and event status dashboard

For deeper analysis, I drilled into specific alerts to view the full JSON document, including rule IDs, categories, and the raw event.

The Agents page confirmed the Windows endpoint was active and communicating, with heartbeat and recent event timestamps.

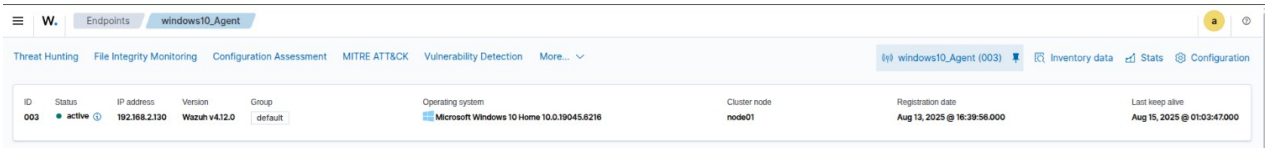


Figure 5.2.4: Agent page confirmation

5.4 pfSense Logs and Cross-Validation

I checked pfSense Firewall Logs to cross-validate timing of inbound attempts from the Kali scanner towards the Windows IP during the test window. This step ties together edge logging with endpoint alerts, proving the visibility chain is intact.

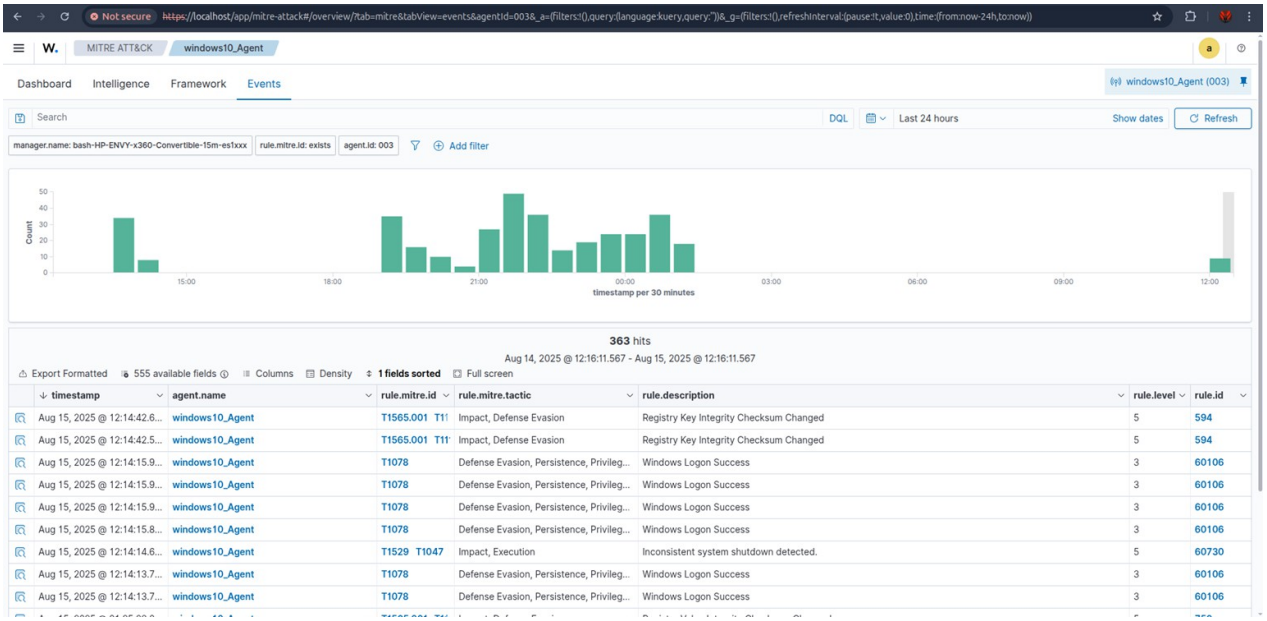


Table 2. Test Scenarios and Expected vs. Observed Alerts

Scenario	Expected Outcome
Windows file edit (FIM)	FIM alerts with file path + new hash
Windows registry tweak	Registry change alert + key path
Kali Nmap TCP SYN scan	Network activity spikes; pfSense log entries

Agent heartbeat after changes Agent remains Active with fresh events

DISCUSSION

3.5 What Worked Well

• Isolation: Host-Only LAN kept tests safe and predictable.

• Agent Telemetry: Wazuh provided rich, correlated alerts (FIM + registry + system).

• Edge + Endpoint Story: pfSense logs + Wazuh alerts gave a full picture from perimeter to host.

• Operational Visibility: The Node process on :443 and agent_control -l checks helped confirm service health quickly.

3.6 Observed Gaps and Limitations

• Initial “Allow Any” Rule: Helpful for bring-up but should be narrowed post-testing.

• Single Endpoint: More endpoints (Linux server, additional Windows client) would provide stronger behavioural baselines.

CONCLUSIONS AND RECOMMENDATIONS

3.7 Conclusions

The lab shows a clear, reproducible path to building a small SOC-style environment. pfSense reliably provided routing, DHCP, and logs; Wazuh captured endpoint behaviour and turned it into actionable alerts. The combined view made it straightforward to explain what happened on the wire and on the host.

3.8 Recommendations

1. Tighten LAN Rules: Replace “LAN → Any” with explicit allowlists (DNS, HTTP/HTTPS to the SIEM/Kibana, Windows update if needed).

2. Add IDS: Deploy Suricata or Zeek (on pfSense or a sensor VM) and forward EVE JSON into Wazuh for DPI-backed detections.

3. Use Agent Groups and Policies: Standardise agent configs (FIM paths, registry keys, Sysmon) using Wazuh’s centralised management.

4. Automate: Create detection rules for port-scan bursts and automatic ticketing/notifications.

5. Scale Out: Add more endpoints and a small Linux server in a DMZ segment, then test lateral movement detections.

REFERENCES

• Bejtlich, R. (2013). The Practice of Network Security Monitoring. No Starch Press.

• The pfSense Project. (2025). pfSense Documentation.

• Wazuh, Inc. (2025). Wazuh Documentation.

• MITRE ATT&CK® (2025). Enterprise matrix and technique references.