

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

22<sup>nd</sup> 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 demonstrate 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

Abstract.....	i
Acknowledgments.....	ii
Table of Contents.....	iii
List of Figures.....	iv
List of Tables.....	v
1. Introduction.....	1
1.1 Background.....	1
1.2 Problem Statement.....	2
1.3 Objectives.....	2
1.4 Scope and Limitations.....	3
1.5 Report Organization.....	3
2. Literature Review.....	4
2.1 Firewalls and Segmentation (pf Sense).....	4
2.2 SIEM (Wazuh) and Endpoint Telemetry.....	5
2.3 Related Academic Labs and Teaching Setups.....	7
3. Methodology and Lab Architecture.....	8
3.1 Lab Requirements and Folder Structure.....	8
3.2 Virtual Networking and VPN Considerations.....	9
3.3 Topology and IP Plan.....	11
4. Implementation.....	13
4.1 pfSense Installation and Core Services.....	13
4.2 Wazuh SIEM on Ubuntu Host (Manager, ES, Kibana, Filebeat)	
.....	18

4.3 Windows 10 Agent Onboarding and Manager Authentication	22
5. Verification, Testing, and Results.....	26
5.1 File Integrity Monitoring (FIM) and Registry Alerts.....	26
5.2 Reconnaissance from Kali (Nmap) and Network Activity .....	
29	
5.3 Wazuh Dashboards, Compliance, and Event Statistics.....	31
5.4 pfSense Logs and Cross-Validation.....	33
6. Discussion.....	35
6.1 What Worked Well.....	35
6.2 Observed Gaps and Limitations.....	36
7. Conclusions and Recommendations.....	38
8.....References	40
9.....Appendices	42

## 10. LIST OF TABLES

11. Table 1. IP Plan and Interfaces

12. Table 2. Test Scenarios and Expected vs. Observed Alerts

Table 3. pfSense LAN Rules (Initial Bring-up Profile)

# 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 pfSense’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.

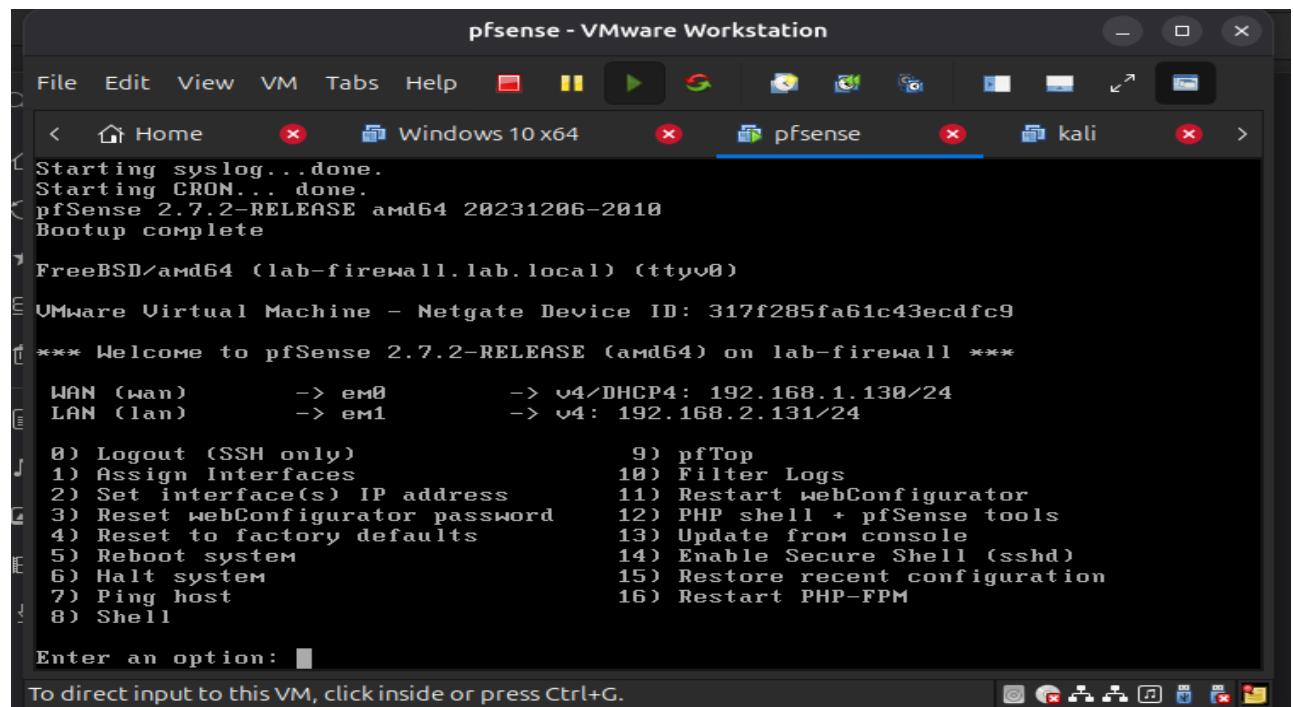


Figure 4.1.1:Pfsense interface showing all the interfaces

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

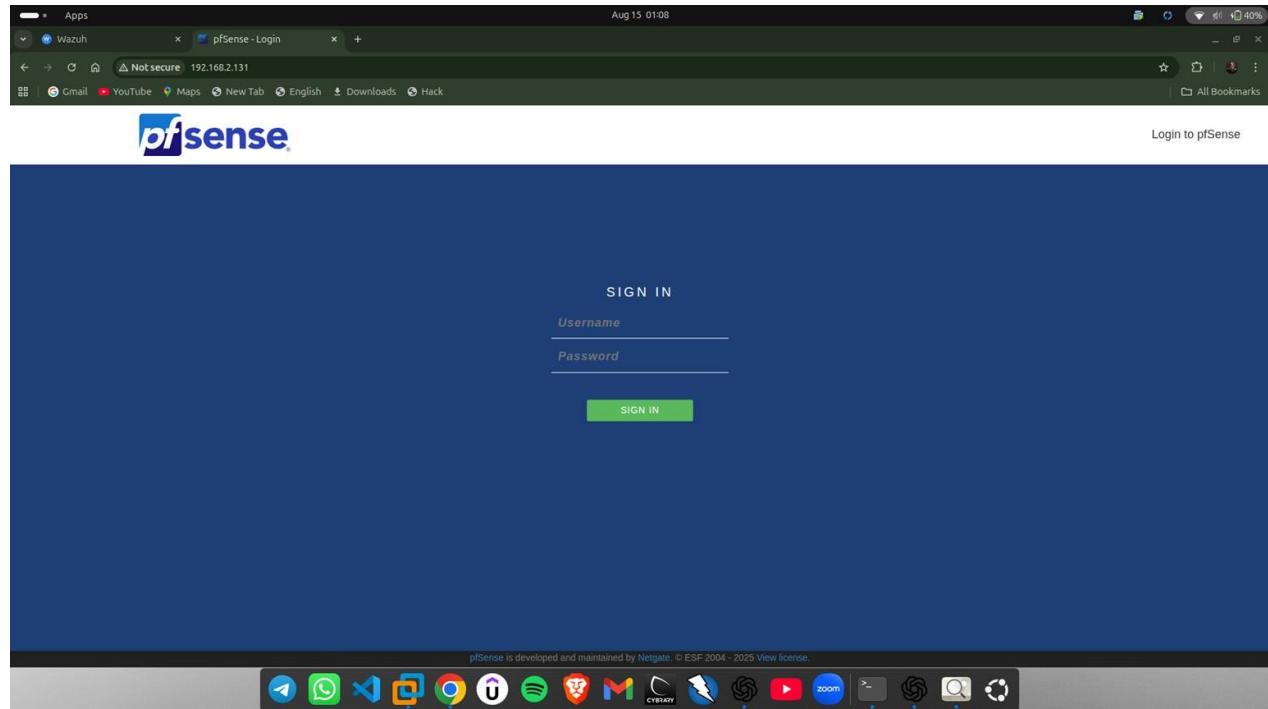


Figure 4.1.2: Web Graphical Interface of PfSense

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

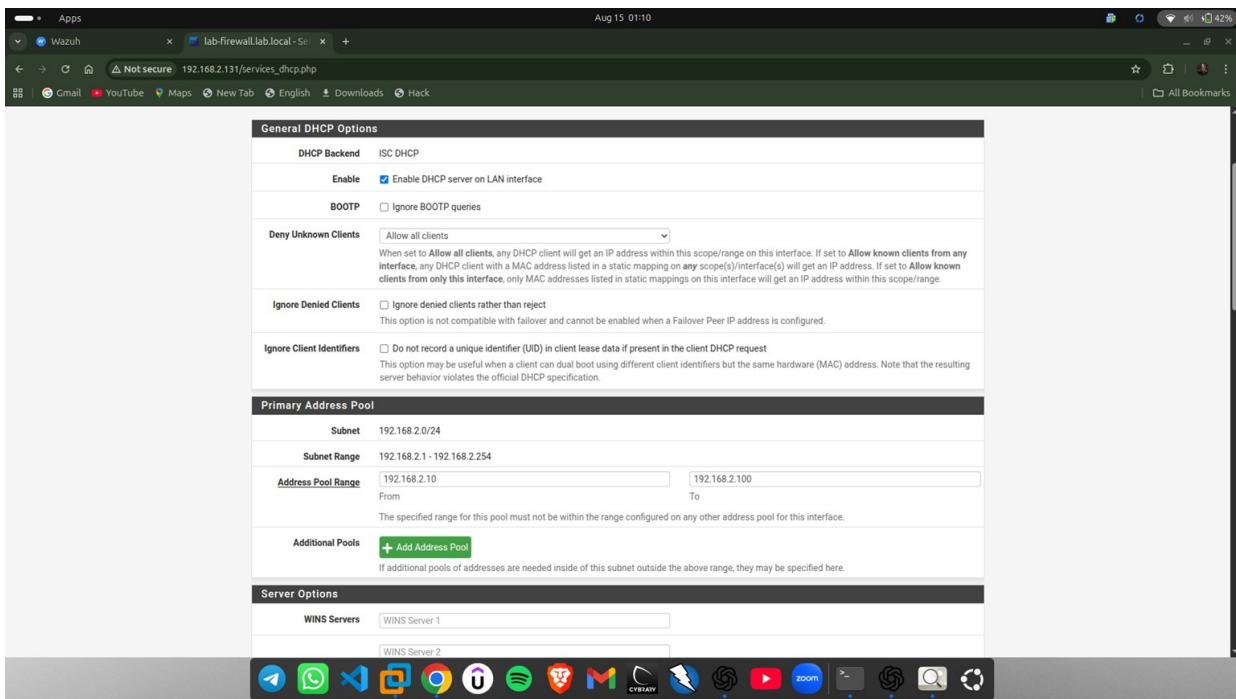


Figure 4.1.3: dhcp server on lan

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 LANRemote Syslog to SIEM. I configured Status → System Logs →

Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	6/946 Kib	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	0/0 B	IPv4 TCP	LAN subnets	*	*	4444	*		none		
<input type="checkbox"/>	0/0 B	IPv4 TCP	LAN subnets	*	*	23 (Telnet)	*		none		
<input checked="" type="checkbox"/>	0/0 B	IPv4 TCP	LAN subnets	*	*	443 (HTTPS)	*		none		
<input checked="" type="checkbox"/>	0/0 B	IPv4 TCP	LAN subnets	*	*	80 (HTTP)	*		none		
<input checked="" type="checkbox"/>	0/0 B	IPv4 *	LAN subnets	*	*	*	*		none	Default allow LAN to any rule	
<input checked="" type="checkbox"/>	0/0 B	IPv6 *	LAN subnets	*	*	*	*		none	Default allow LAN IPv6 to any rule	

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

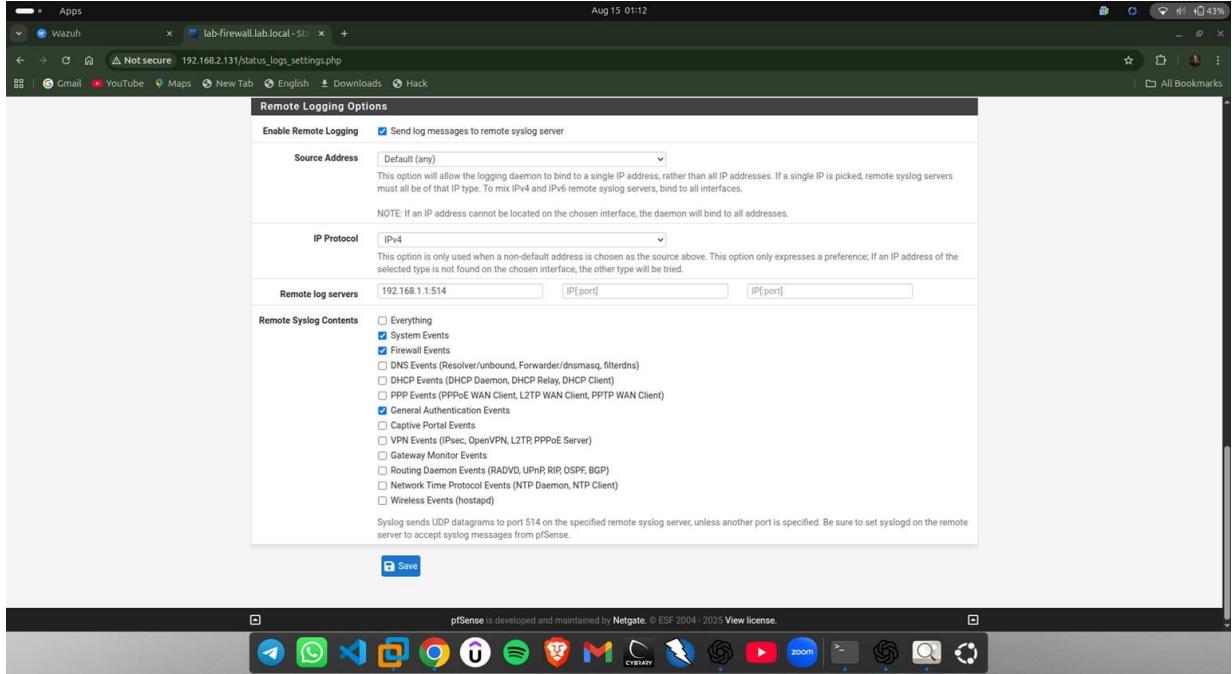


Figure 4.1.5:Remote Syslog to SIEM

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`.

The screenshot shows a terminal window titled "bash@bash-HP-ENVY-x360-Convertible-15m-es1xxx: ~". The user runs the command `sudo bash wazuh-install.sh -m`. The terminal displays the usage information for the `wazuh-install.sh` script, which includes options for installing the Wazuh server, indexer, and dashboard. The output is in white text on a dark background.

```
bash@bash-HP-ENVY-x360-Convertible-15m-es1xxx:~$ sudo bash wazuh-install.sh -m
[sudo] password for bash:
Unknown option: -m

NAME
      wazuh-install.sh - Install and configure Wazuh central components: Wazuh
server, Wazuh indexer, and Wazuh dashboard.

SYNOPSIS
      Here's the breakdown:
      wazuh-install.sh [OPTIONS] -a | -c | -s | -wi <indexer-node-name> | -wd
      dashboard-node-name> | -ws <server-node-name>

DESCRIPTION
      Alternatively, if you want to install it manually on Ubuntu/Debian:
      debug   -a, --all-in-one
              Install and configure Wazuh server, Wazuh indexer, Wazuh dashbo
      ard.
      generation
      sudo apt update
      -c, --config-file <path-to-config-yml>
      Path to the configuration file used to generate wazuh-install-fi
      les.tar file containing the files that will be needed for installation. By defau
      lt, the Wazuh installation assistant will search for a file named config.yml in
      the same path as the script.

      -dw, --download-wazuh <deb|rpm>
```

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:



A screenshot of a terminal window titled "bash@bash-HP-ENVY-x360-Convertible-15m-es1xxx:~". The window contains the following command and its output:

```
bash@bash-HP-ENVY-x360-Convertible-15m-es1xxx:~$ sudo ss -ltnp | grep 5601 || sudo ss -ltnp | grep node
[sudo] password for bash:
LISTEN 0 100 0.0.0.0:443 0.0.0.0:*
LISTEN 0 100 0.0.0.0:5601 0.0.0.0:*
users:(("node",pid=1608,fd=19))
```

Figure 4.2.2: Checking Wazuh statuses

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

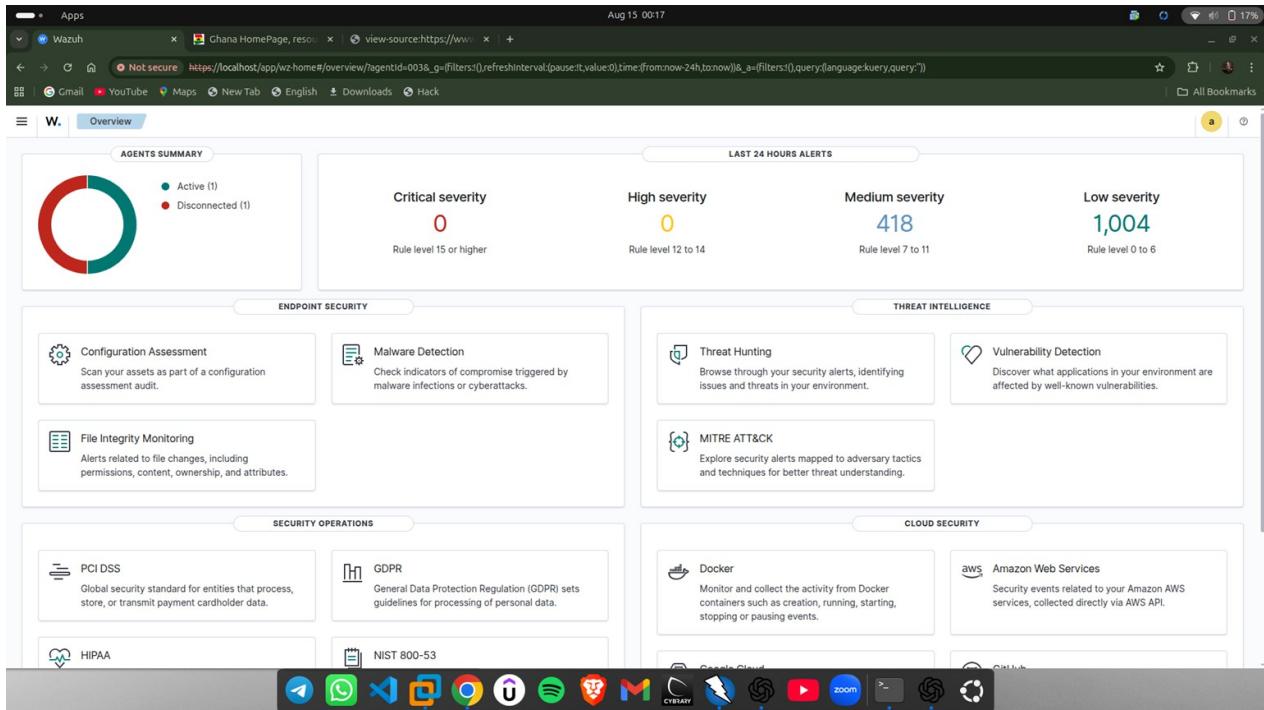


Figure 4.2.3: Wazuh dashboard showing all agent availability

### 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.

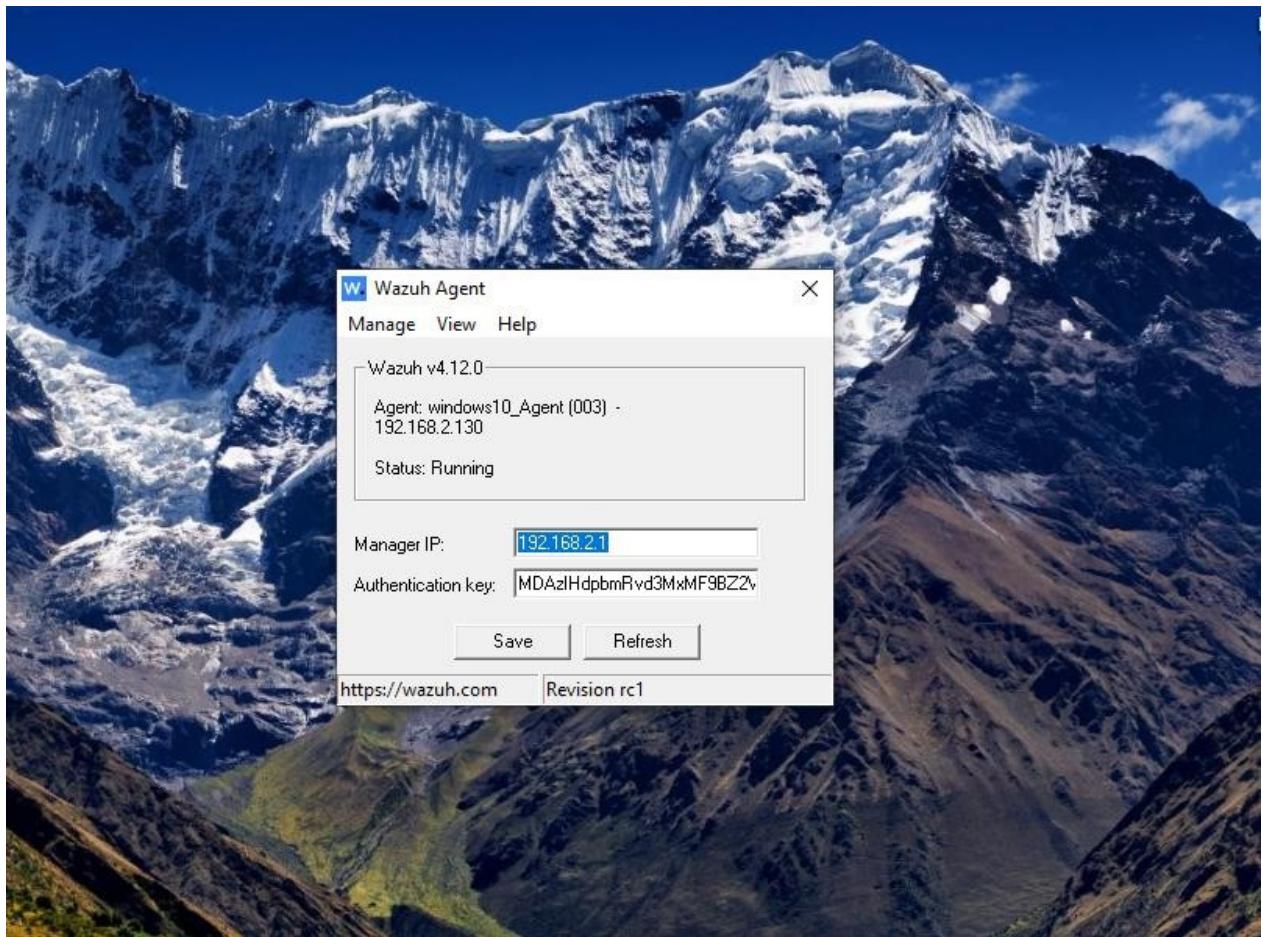
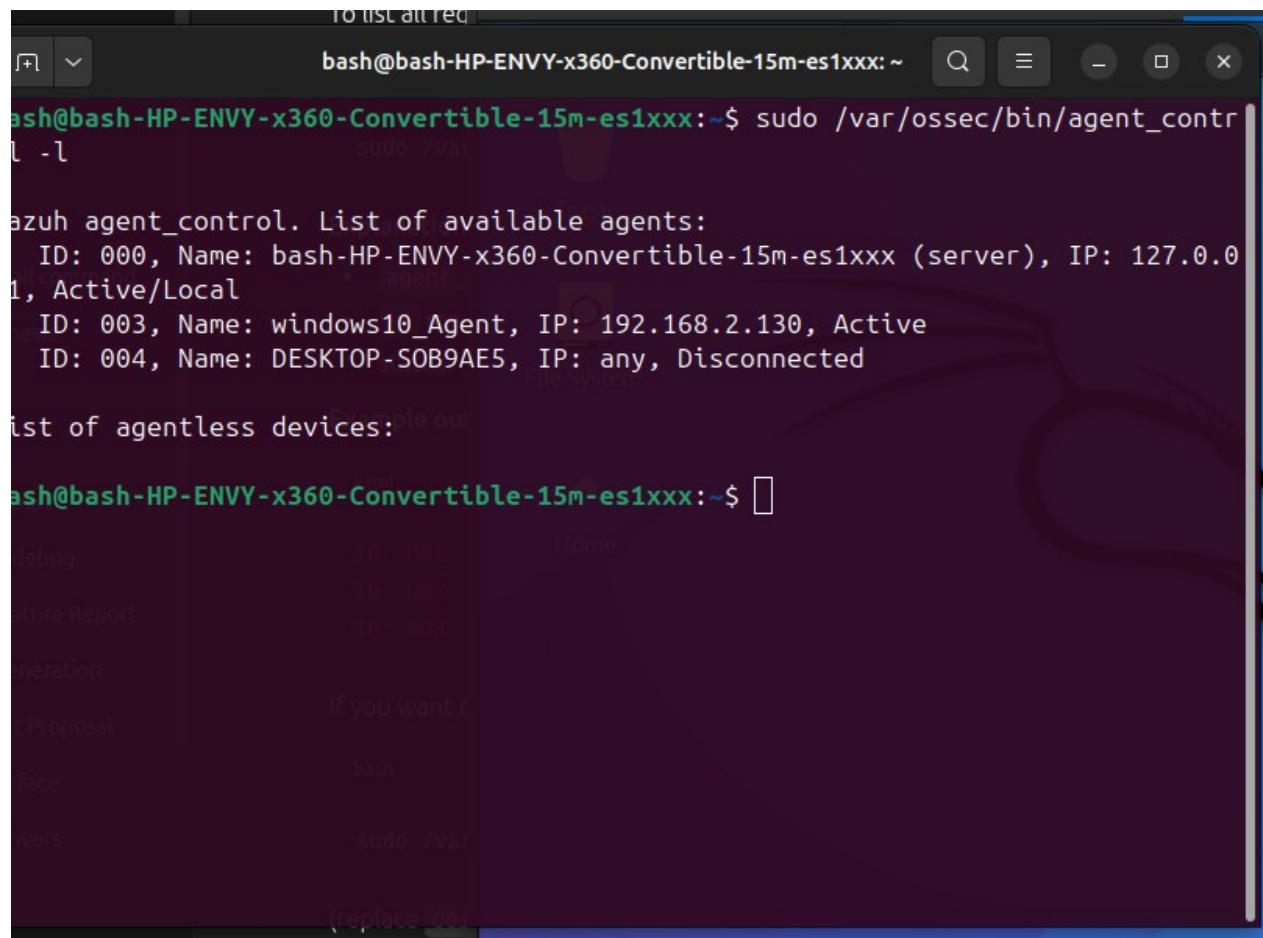


Figure 4.3.1: Wazuh Agent installation

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
```



The screenshot shows a terminal window titled "To list all req" with the command "bash@bash-HP-ENVY-x360-Convertible-15m-es1xxx:~\$ sudo /var/ossec/bin/agent\_control -l". The output lists available agents:

```
azuh agent_control. List of available agents:  
ID: 000, Name: bash-HP-ENVY-x360-Convertible-15m-es1xxx (server), IP: 127.0.0.1, Active/Local  
ID: 003, Name: windows10_Agent, IP: 192.168.2.130, Active  
ID: 004, Name: DESKTOP-SOB9AE5, IP: any, Disconnected
```

It also lists agentless devices:

```
list of agentless devices:
```

At the bottom, there is a message: "If you want to replace [replace\_001] with your own value, run: sudo /var/ossec/bin/agent\_control -l [replace\_001]".

Figure 4.3.2:Wazuh agent verification

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

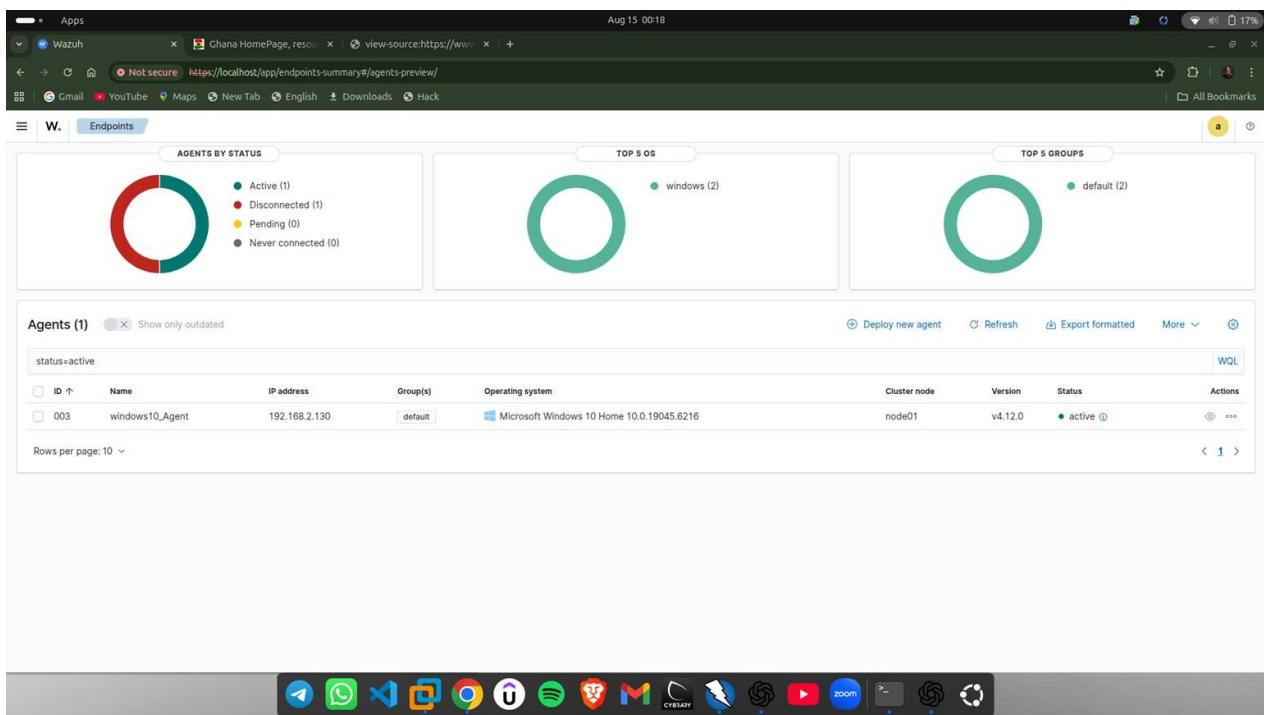


Figure 4.3.3: Dashboard view of agents

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.

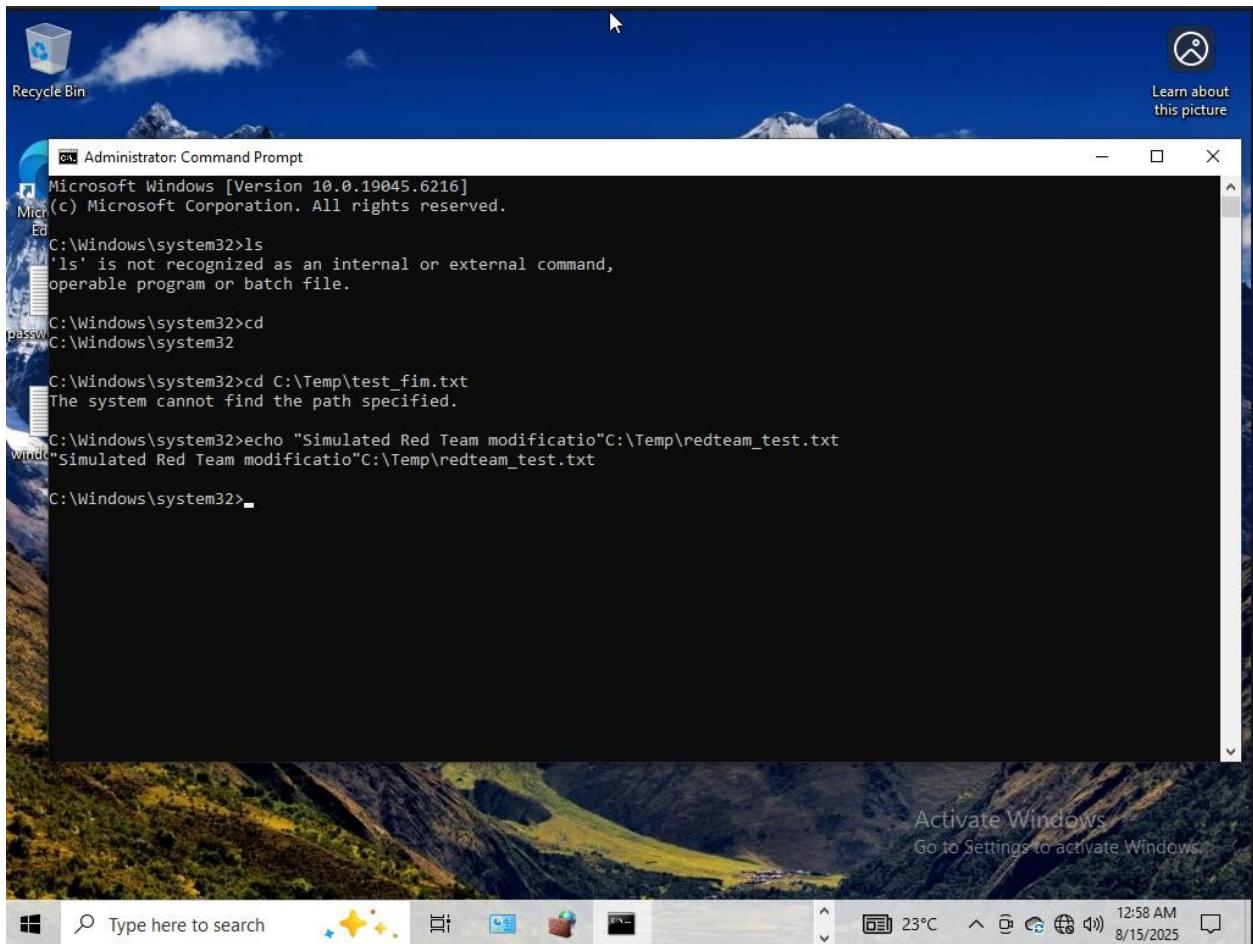


Figure 5.1.1: Initialising alerts with filepath and hashes

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).

The screenshot shows a web browser window with the following details:

- Header:** Apps, Wazuh, Not secure, https://localhost:app/endpoints-summary#/agents?tab=welcome&agent=003, Aug 15 01:04, 36% battery.
- Toolbar:** Gmail, YouTube, Maps, New Tab, English, Downloads, Hack, All Bookmarks.
- Left Panel:** Shows event counts: 0 Critical, 0 High, 0 Medium, 0 Low.
- Middle Panel:** "Top 5 Packages" section showing "No recent events".
- Right Panel:** "CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0" report with a green status bar labeled "cis\_win10\_enterprise". The report table includes columns: Policy, End scan, Passed, Failed, Not applic..., Score. One row is listed: CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0, Aug 14, 2025 @ 21:02:26.000, 126, 264, 4, 32%.
- Bottom Panel:** "FIM: Recent events" table with columns: Time, Path, Action, Rule description, Rule Lev., Rule Id. Five entries are listed, all modified actions on registry keys.
- Bottom Bar:** A dark bar with various icons for messaging, file sharing, and system functions.

Figure 5.1.2: Event logs

What I looked for in the alerts (examples):

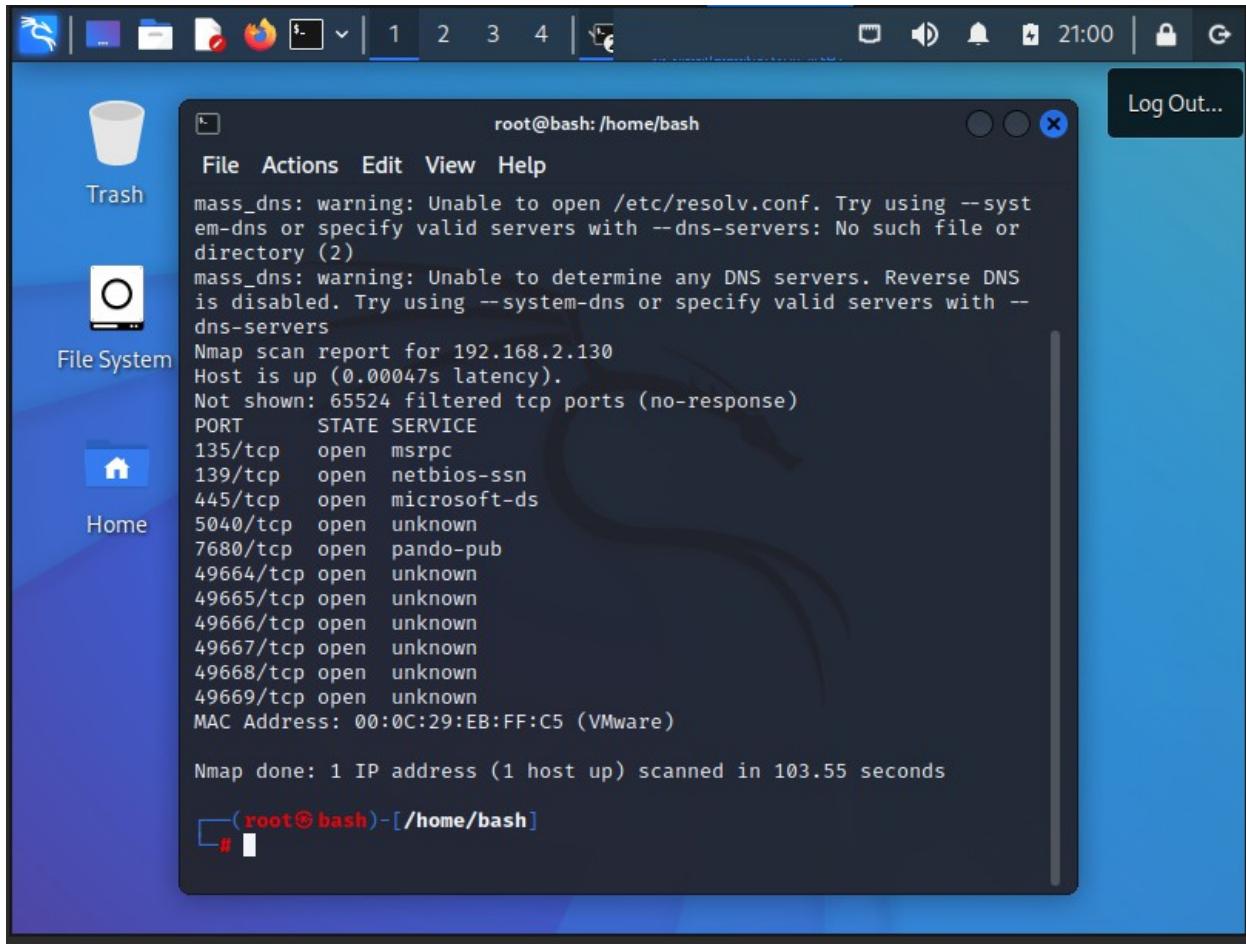
- Rule names/tags (e.g., fim, windows\_registry)
- Old vs. new hash for files (SHA1/SHA256)
- Registry hive/path and value type changes
- Agent name and IP (mapping to the endpoint)

### 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).



The screenshot shows a terminal window titled "root@bash:/home/bash". The terminal displays the results of an Nmap scan for host 192.168.2.130. The output includes messages about DNS resolution issues, the host being up, and a list of open TCP ports along with their corresponding services. The terminal also shows the MAC address of the interface used for the scan.

```
mass_dns: warning: Unable to open /etc/resolv.conf. Try using --system-dns or specify valid servers with --dns-servers: No such file or directory (2)
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.2.130
Host is up (0.00047s latency).
Not shown: 65524 filtered tcp ports (no-response)
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
5040/tcp   open  unknown
7680/tcp   open  pando-pub
49664/tcp  open  unknown
49665/tcp  open  unknown
49666/tcp  open  unknown
49667/tcp  open  unknown
49668/tcp  open  unknown
49669/tcp  open  unknown
MAC Address: 00:0C:29:EB:FF:C5 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 103.55 seconds

(root@bash)-[~/home/bash]
#
```

Figure 5.2.1: Scanning windows endpoints

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

The screenshot shows the Wazuh web interface with the following details:

- Endpoints** tab selected.
- windows10\_Agent** agent selected.
- Vulnerability Detection** section:
  - 0 Critical
  - 0 High
  - 0 Medium
  - 0 Low
- Top 5 FIM: Recent events** table:

Time	Path
Aug 15, 2025 @ 01:25:23.035	HKEY_LOCAL_MACHINE\
Aug 15, 2025 @ 01:25:23.019	HKEY_LOCAL_MACHINE\
Aug 15, 2025 @ 01:25:23.004	HKEY_LOCAL_MACHINE\
Aug 15, 2025 @ 01:25:22.993	HKEY_LOCAL_MACHINE\
Aug 15, 2025 @ 01:25:22.993	HKEY_LOCAL_MACHINE\
- Details** section for HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{ea88619b-5be1-4f27-a14d-0ec38b5010b4}:

  - Last analysis: Aug 15, 2025 @ 01:24:55.000
  - Last modified: Aug 15, 2025 @ 01:24:54.000

- Registry values** table:

Date	Value name	Value type	sha1
Aug 15, 2025 @ 01:24:56.000	DhcpInterfaceOptions	REG_BINARY	1da2c566e2650ea3a10b65 1aad29b36da22c9bb6
Aug 15, 2025 @ 01:24:56.000	LeaseTerminatesTime	REG_DWORD	a3501b954513ff89fe2aa6d ec749b941c8e819a4
Aug 15, 2025 @ 01:24:56.000	T2	REG_DWORD	7abff0b053ded76c949a12 1fb685f1c2af1e970d
Aug 15, 2025 @ 01:24:56.000	T1	REG_DWORD	0102e2325db4d6153851fe 8a59fdf6645454107f
Aug 15, 2025 @ 01:24:56.000	LeaseObtainedTime	REG_DWORD	f43cd79aa4586abe016ac08 fdef9e244db034e1a
- Buttons at the bottom: Search, DQL, Last 24 hours, Show dates, Refresh.

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.

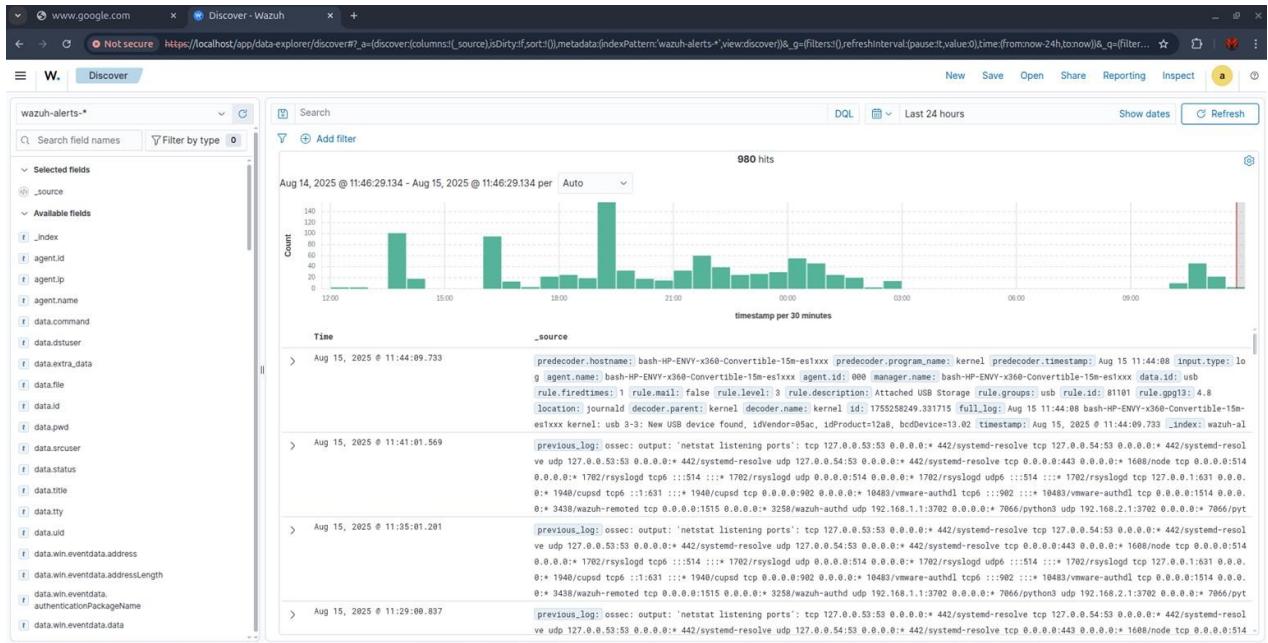


Figure 5.2.3: compliance and event status dashboard

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.

The screenshot shows the Wazuh Agents page. At the top, there are tabs for Threat Hunting, File Integrity Monitoring, Configuration Assessment, MITRE ATT&CK, Vulnerability Detection, and More... Below these, a search bar contains '(w) windows10\_Agent (003)'. To the right are links for Inventory data, Stats, and Configuration. The main table displays the following information for the windows10\_Agent endpoint:

ID	Status	IP address	Version	Group	Operating system	Cluster node	Registration date	Last seen
003	active	192.168.2.130	Wazuh v4.12.0	default	Microsoft Windows 10 Home 10.0.19045.6216	node01	Aug 13, 2025 @ 16:39:56.000	Aug 15, 2025 @ 01:03:47.000

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.

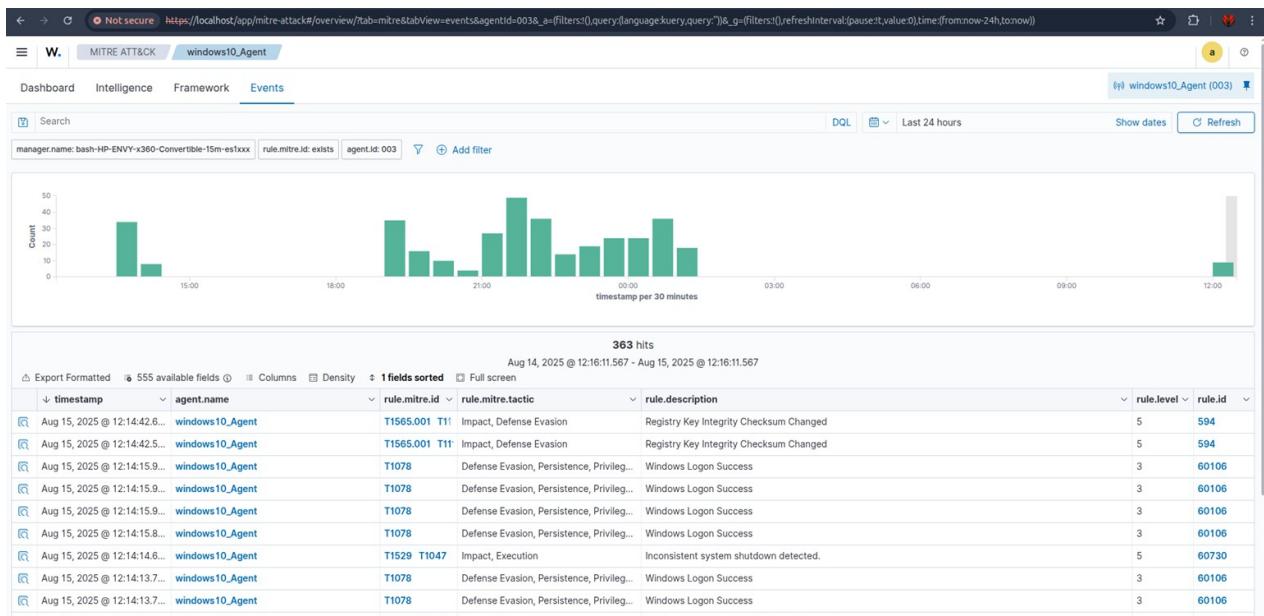


Table 2. Test Scenarios and Expected vs. Observed Alerts

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.