Cyberster

INTERNSHIP REPORT

Week 4: pfSense Firewall and Advanced Monitoring

Submitted to: Cyberster Internship Program

Intern Name: Abdul Muqeet Tabraiz

Roll Number: CSI-B1-353

Table of Contents

Page Left Blank Intentionally

Cyberster Blue Team Internship — Week 4 Days 22–23

pfSense Firewall Setup

Objective

The objective of Days 22 and 23 was to deploy pfSense CE 2.7.2 as a virtual firewall and router in the lab environment, routing all traffic between VMs through pfSense before it reaches its destination. This involved creating a dedicated pfSense virtual machine with two network adapters (WAN and LAN), installing pfSense CE 2.7.2 from the official ISO, configuring the LAN IP to match the lab subnet (192.168.35.0/24), reconfiguring existing VMs to use pfSense as their default gateway, creating firewall rules with logging enabled on the LAN and WAN interfaces, enabling remote syslog forwarding from pfSense to the Wazuh Manager, and configuring a custom pfSense decoder in Wazuh to ensure firewall log entries were correctly parsed and appeared in the Security Events dashboard.

Tools Used

- VMware Workstation — virtualisation platform used to create and host the pfSense VM
- pfSense CE 2.7.2-RELEASE — open-source FreeBSD-based firewall and router
- pfSense WebGUI — web interface accessed at http://192.168.35.2 for configuration
- Ubuntu Server (Wazuh Manager) — 192.168.35.131, netplan updated to route through pfSense
- Kali Linux (Agent 001) — 192.168.35.128, gateway changed to pfSense LAN IP
- /var/ossec/etc/ossec.conf — Wazuh Manager config updated with syslog remote listener
- /var/ossec/etc/decoders/0001-pfsense-fix.xml — custom decoder to fix FreePBX false match on filterlog
- tcpdump — used on Ubuntu to verify pfSense UDP 514 syslog packets were arriving

Procedure

Step 1: Create the pfSense Virtual Machine

A new virtual machine was created in VMware Workstation with two network adapters. Adapter 1 was configured as Bridged (VMnet0) to provide pfSense WAN with a real IP from the home router for internet access. Adapter 2 was configured as NAT (VMnet8) to connect pfSense LAN to the 192.168.35.0/24 lab network shared with Ubuntu and Kali. The pfSense CE 2.7.2 ISO was attached as the boot device.

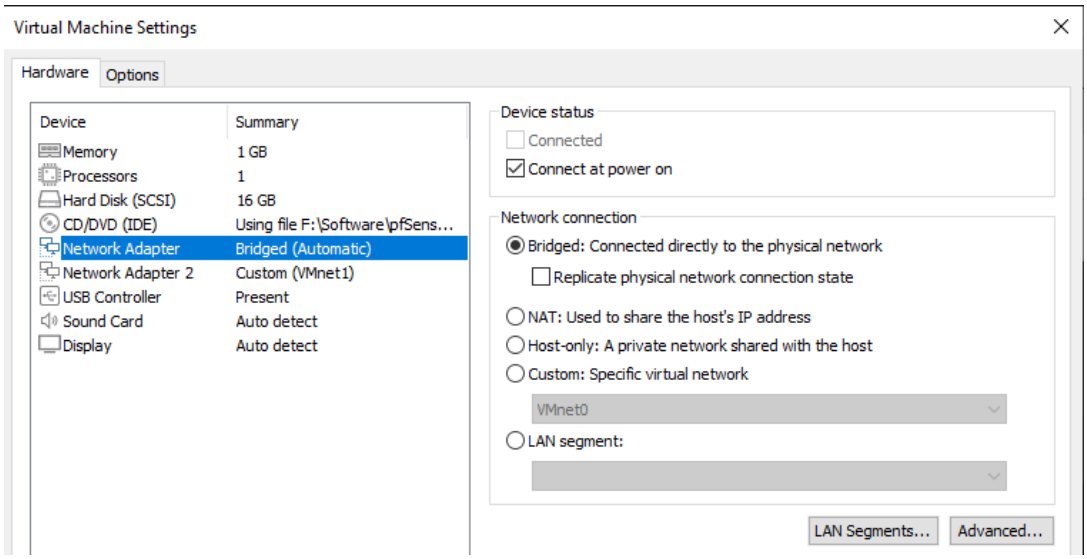

Fig 22.1 VMware Virtual Machine Settings — pfSense VM hardware configuration showing 1GB RAM, 16GB disk, and two network adapters configured for Bridged (WAN) and NAT (LAN)

Step 2: Install pfSense CE 2.7.2

The pfSense VM was booted from the ISO. The installer presented the standard menu. Auto (UFS) partitioning was selected, the entire disk was used, and the installation completed without errors. After installation the VM was rebooted with the ISO detached.

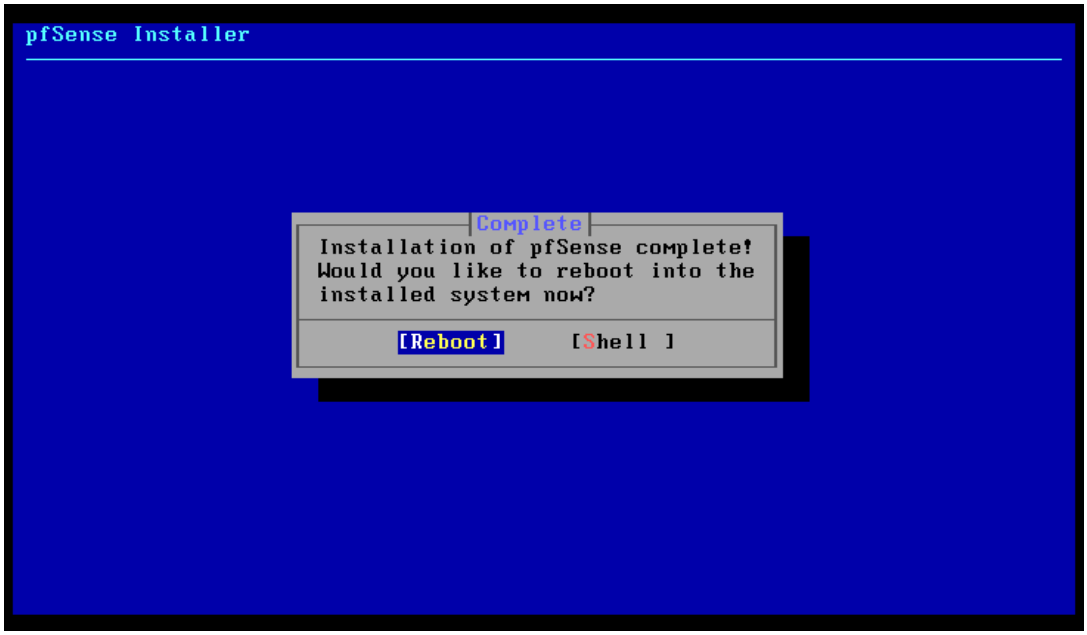

Fig 22.2 pfSense Installer — Reboot screen confirming pfSense CE 2.7.2 installation completed successfully, ready to boot from hard disk

Step 3: Assign Interfaces and Set LAN IP

After first boot the pfSense console presented the interface assignment menu. em0 was assigned as WAN (Bridged, receiving an IP from the home router via DHCP) and em1 was assigned as LAN. Since em1 landed on the wrong default subnet, option 2 (Set interface(s) IP address) was used to manually set the LAN IP to 192.168.35.2/24. DHCP was disabled on LAN as static IPs are used across the lab.

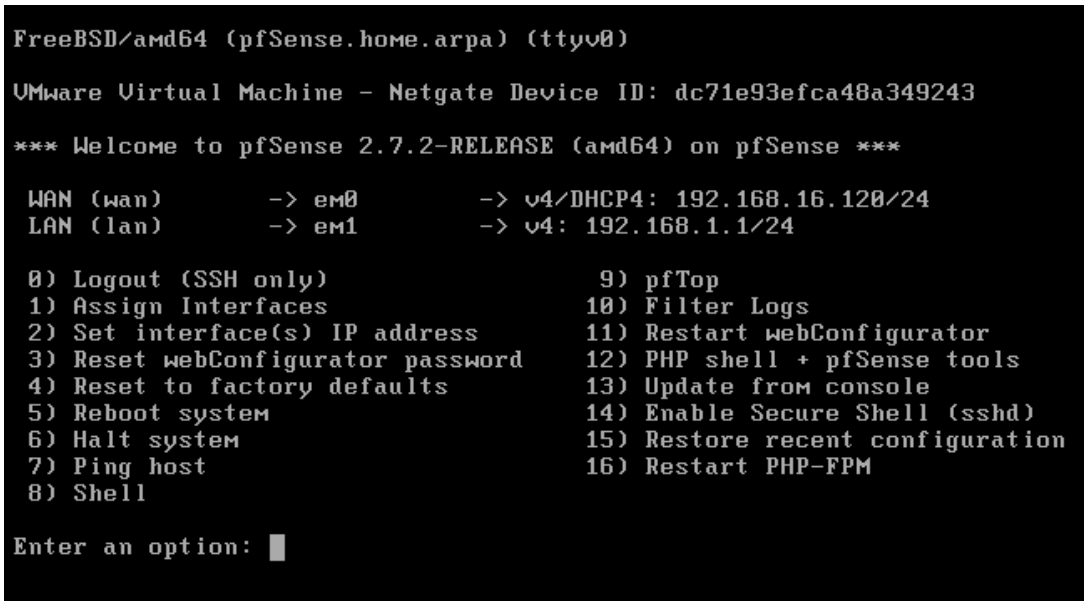

Fig 22.3 pfSense Console — Welcome screen showing WAN (em0) assigned with DHCP4 address from home router and LAN (em1) configured — interface assignment complete

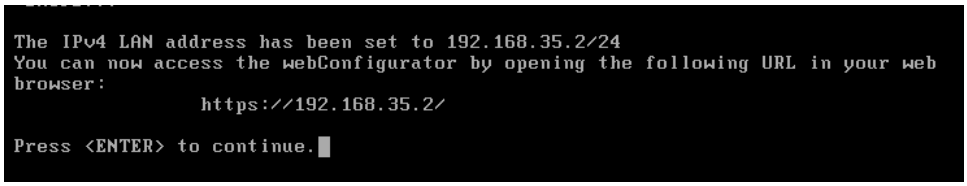

Fig 22.4 pfSense Console — Confirmation message: "The IPv4 LAN address has been set to 192.168.35.2/24" with WebGUI URL shown

Step 4: Access pfSense WebGUI and Complete Setup Wizard

From the host machine browser the pfSense WebGUI was accessed at http://192.168.35.2. Login with admin/pfsense credentials opened the Setup Wizard. Hostname was set to pfSense, DNS servers set to 8.8.8.8 and 8.8.4.4, LAN IP confirmed at 192.168.35.2/24, and the admin password was changed from the default. After the wizard completed the pfSense dashboard loaded showing both WAN and LAN interfaces with green status indicators.

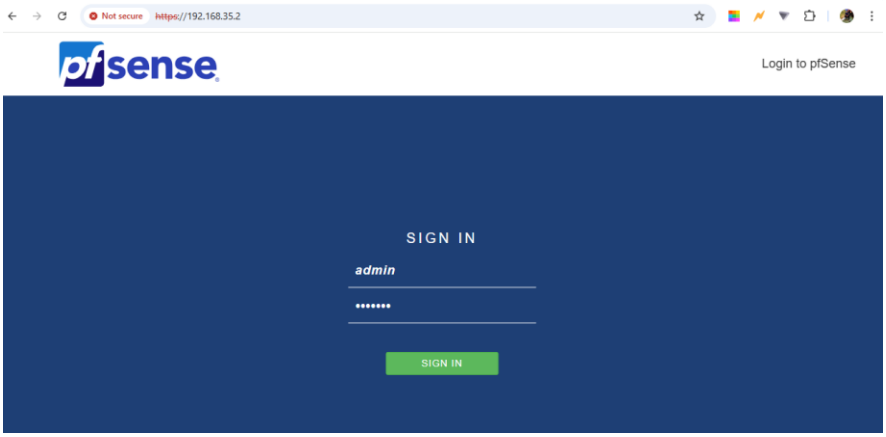

Fig 22.5 pfSense WebGUI — Login page at http://192.168.35.2 confirming the WebGUI is accessible from the host machine browser

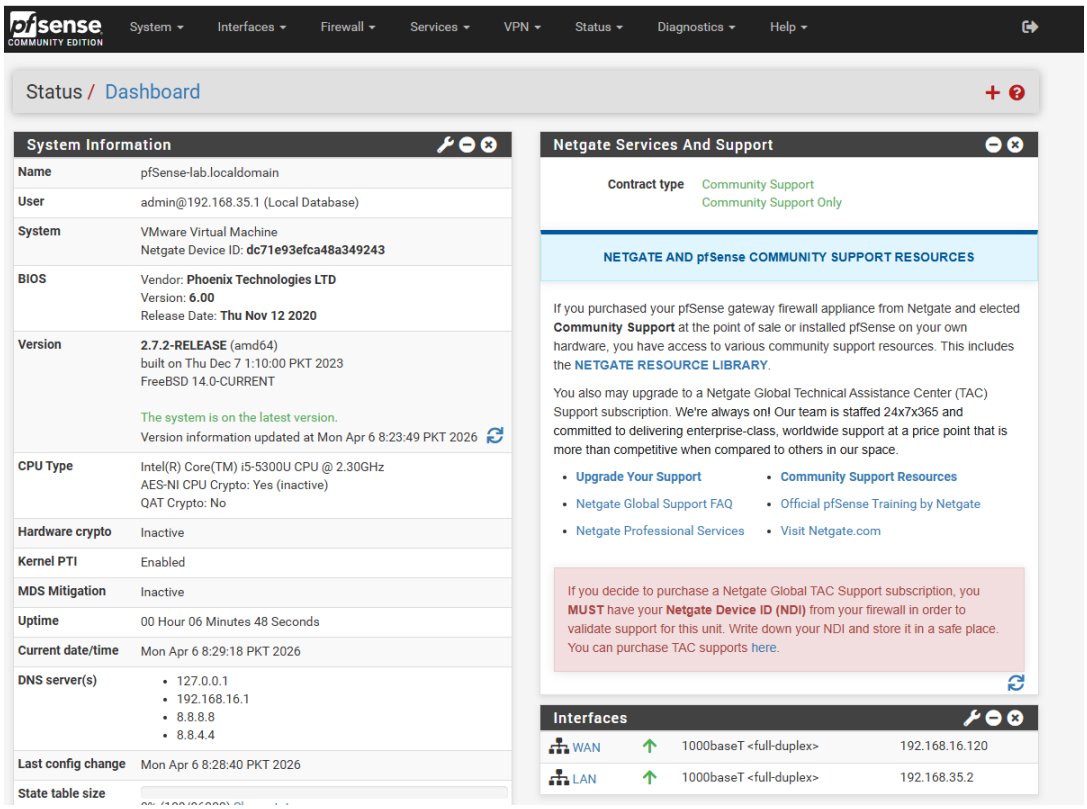

Fig 22.6 pfSense WebGUI Dashboard — Main dashboard showing System Information panel, WAN interface with home router IP, and LAN interface at 192.168.35.2 both with active status — primary Days 22–23 deliverable

Step 5: Reconfigure VM Gateways to Route Through pfSense

Ubuntu Server was reconfigured first. The netplan configuration file was updated to add a second interface (ens34) statically assigned to 192.168.35.131/24 with the default gateway pointing to pfSense at 192.168.35.2. The primary interface ens33 retained its existing NAT address. After applying netplan the routing table confirmed the default route via 192.168.35.2 on ens34 with metric 101, making it the preferred outbound path.

```
sudo nano /etc/netplan/00-installer-config.yaml
sudo netplan apply
```

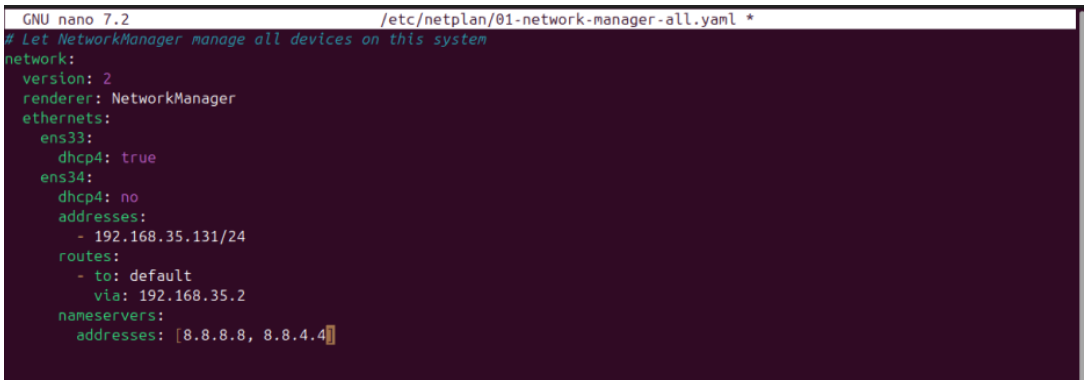

Fig 22.7 Ubuntu Server — Netplan configuration file showing ens34 statically assigned 192.168.35.131/24 with gateway 192.168.35.2 pointing to pfSense LAN

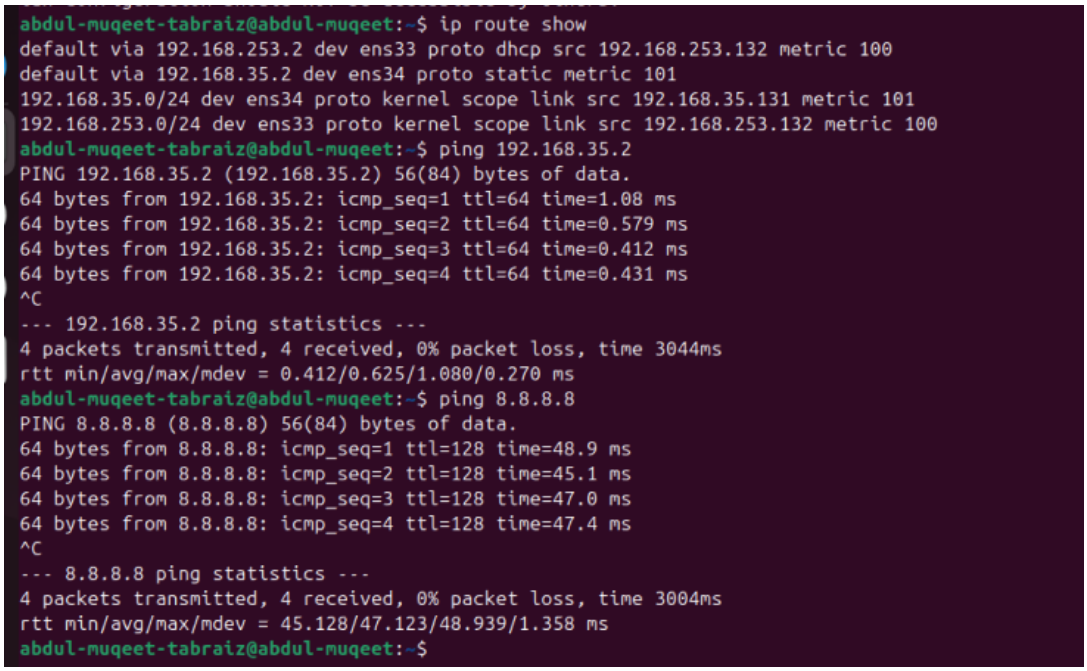

Fig 22.8 Ubuntu Server — ip route show output confirming default route via 192.168.35.2 on ens34 (metric 101) is the active gateway — routing through pfSense confirmed

Kali Linux routing was verified similarly. The ip route command confirmed Kali was also routing through pfSense via the lab subnet. Both VMs were then confirmed to have internet access by pinging 8.8.8.8, verifying that pfSense was correctly forwarding traffic from LAN to WAN.

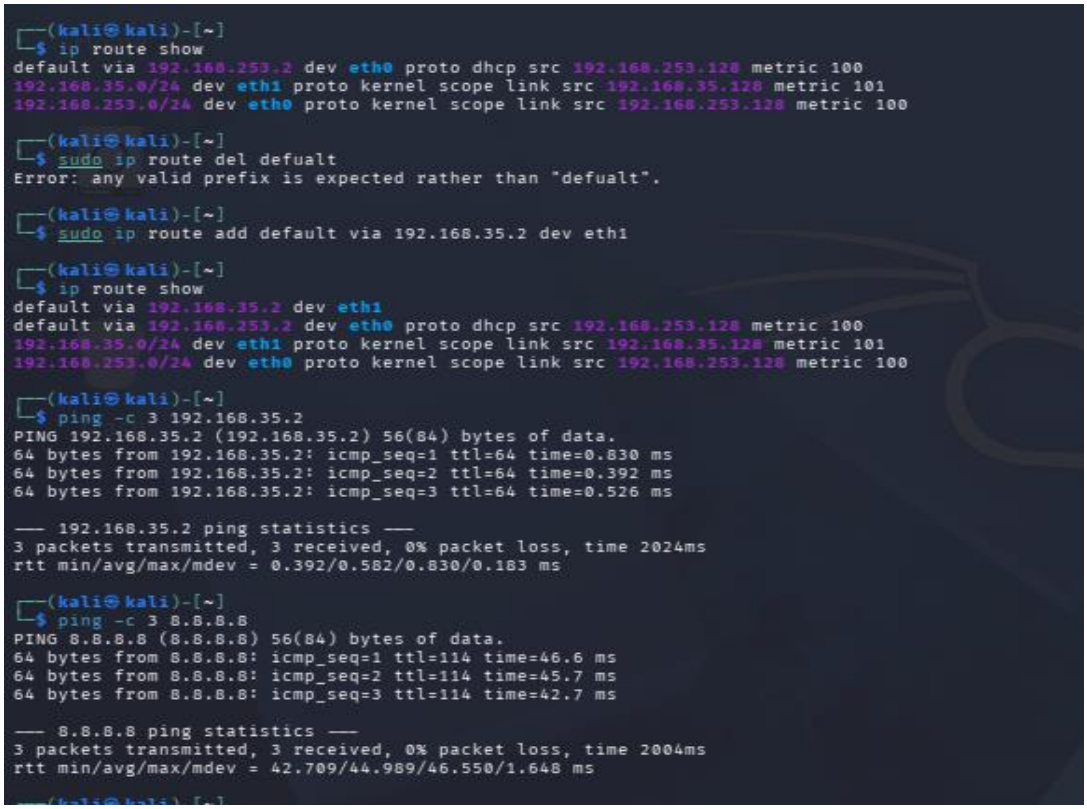

Fig 22.9 Kali Linux — ip route show output showing default route through 192.168.35.2 (pfSense LAN) confirming Kali traffic routes through pfSense

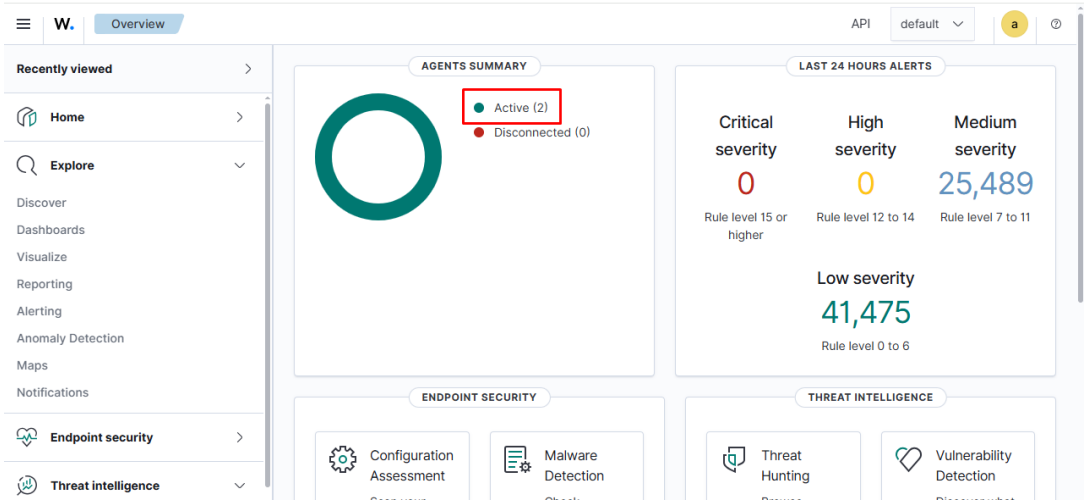

Fig 22.10 Wazuh Dashboard — Agents overview showing all agents Active with 0 disconnected after gateway changes, confirming Wazuh connectivity was maintained through pfSense

Step 6: Configure Firewall Rules with Logging

In the pfSense WebGUI under Firewall → Rules → LAN, two rules were created with logging enabled. Rule 1 passes all LAN traffic with logging to provide full visibility into inter-VM traffic. Rule 2 explicitly allows and logs ICMP between VMs. On the WAN tab a block rule was added to log all unsolicited inbound WAN traffic. The changes were applied and confirmed with the success notification.

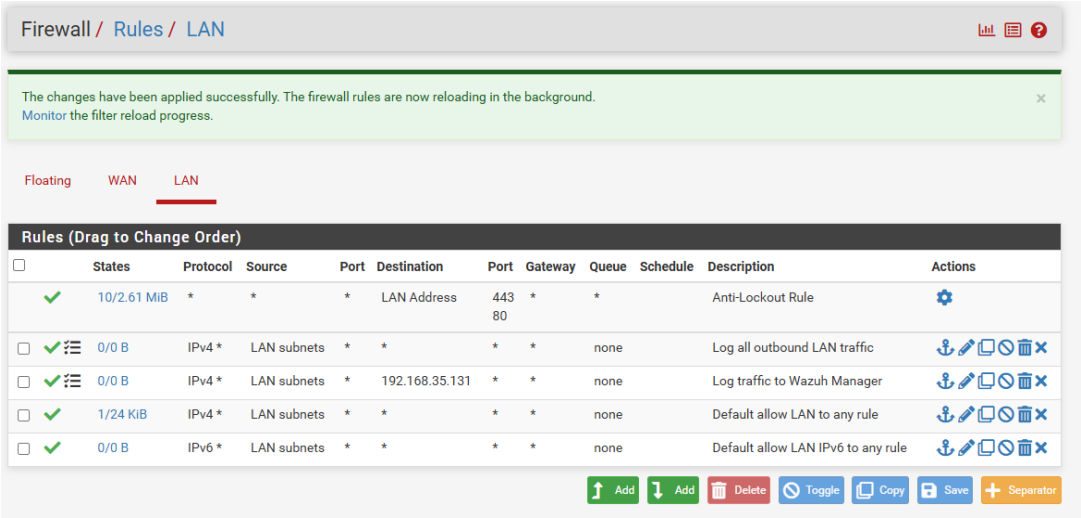

Fig 22.11 pfSense WebGUI — Firewall → Rules → LAN tab showing applied rules with the confirmation banner "The changes have been applied successfully" and rules visible with logging enabled — firewall rules deliverable

Step 7: Enable Remote Syslog Forwarding to Wazuh Manager

In pfSense under Status → System Logs → Settings, remote logging was enabled with the Wazuh Manager IP (192.168.35.131) as the destination syslog server on UDP 514. Firewall Events and System Events were both checked. On the Ubuntu Server, a syslog remote listener block was added to ossec.conf to receive and process the incoming pfSense logs on UDP 514.

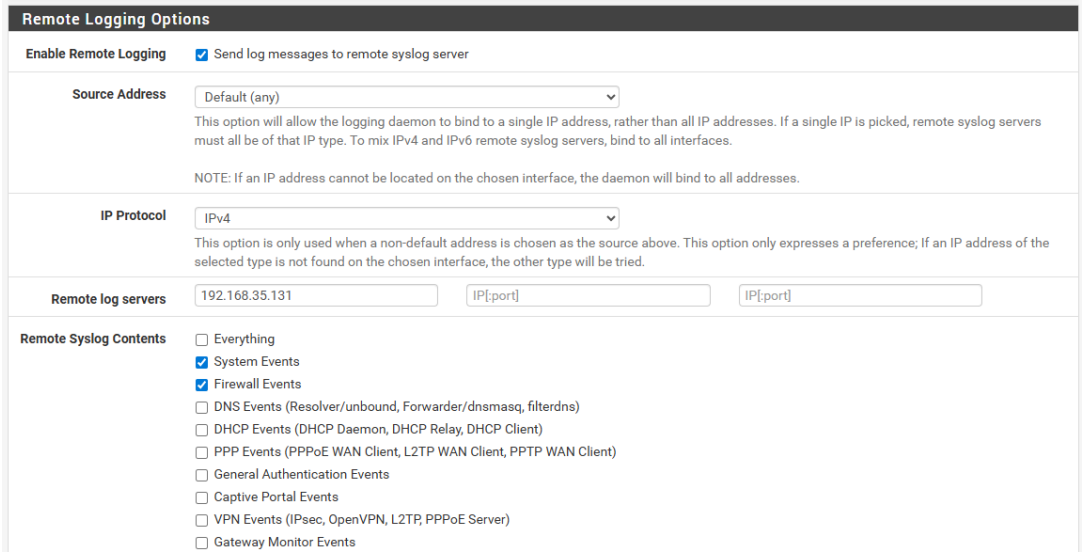

Fig 22.12 pfSense WebGUI — Status → System Logs → Settings showing Enable Remote Logging checked, remote syslog server set to 192.168.35.131 (Wazuh Manager IP) — syslog forwarding deliverable

```
sudo nano /var/ossec/etc/ossec.conf
```

The following remote block was added to ossec.conf on the Wazuh Manager:

```
<remote>
  <connection>syslog</connection>
  <port>514</port>
  <protocol>udp</protocol>
  <allowed-ips>192.168.35.2</allowed-ips>
</remote>
```

Fig 22.13 Ubuntu Server — ossec.conf showing the remote syslog listener block with port 514, protocol udp, and allowed-ips set to 192.168.35.2 (pfSense LAN)

Step 8: Fix pfSense Decoder (FreePBX Conflict)

Testing with wazuh-logtest revealed that the Wazuh pfSense decoder was being overridden by the FreePBX decoder which used a broad prematch pattern that matched the filterlog[PID] format sent by pfSense 2.7.2. A custom decoder file was created at /var/ossec/etc/decoders/0001-pfsense-fix.xml with a specific prematch for filterlog followed by a numeric PID in brackets. This file loads before the FreePBX decoder due to its early filename and overrides the false match, allowing pfSense logs to be correctly parsed by the pf decoder chain.

```
 GNU nano 7.2                          /var/ossec/etc/decoders/0001-pfsense-fix.xml *
<!-- pfSense filterlog decoder fix -->
<decoder name="pfsense-filterlog-pre">
  <prematch>filterlog\[</prematch>
</decoder>

<decoder name="pfsense-filterlog">
  <parent>pfsense-filterlog-pre</parent>
  <prematch>filterlog\[\d+\]</prematch>
  <regex>(\d+),,,\S*,(\w+),(\w+),(\w+),(\d+),</regex>
  <order>id,interface,direction,actioner>
</decoder>
```

Fig 22.14 Ubuntu Server — Custom pfSense decoder file 0001-pfsense-fix.xml showing the prematch pattern filterlog\[\d+\] that correctly captures pfSense 2.7.2 syslog format before the FreePBX decoder matches it

Step 9: Verify Internet and pfSense Log Ingestion

Internet connectivity was verified from both VMs to confirm pfSense was correctly routing traffic through its Bridged WAN interface. Both Kali and Ubuntu successfully pinged 8.8.8.8 through pfSense. After the decoder fix was applied and the Wazuh Manager restarted, pfSense filterlog events began appearing in the Wazuh Security Events dashboard under the pfsense rule group.

```
┌──(kali㉿kali)-[~]
└─$ ping -c 10 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=114 time=48.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=47.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=42.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=43.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=45.3 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=114 time=46.2 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=114 time=46.5 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=114 time=49.0 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=114 time=47.4 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=114 time=46.0 ms

--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9028ms
rtt min/avg/max/mdev = 42.487/46.188/48.963/1.974 ms

┌──(kali㉿kali)-[~]
└─$ ping -c 5 192.168.16.1
PING 192.168.16.1 (192.168.16.1) 56(84) bytes of data.
64 bytes from 192.168.16.1: icmp_seq=1 ttl=63 time=1.78 ms
64 bytes from 192.168.16.1: icmp_seq=2 ttl=63 time=1.79 ms
64 bytes from 192.168.16.1: icmp_seq=3 ttl=63 time=1.93 ms
64 bytes from 192.168.16.1: icmp_seq=4 ttl=63 time=1.79 ms
64 bytes from 192.168.16.1: icmp_seq=5 ttl=63 time=1.69 ms

--- 192.168.16.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.688/1.795/1.934/0.078 ms

┌──(kali㉿kali)-[~]
└─$ nmap -sS 192.168.35.131
Starting Nmap 7.98 ( https://nmap.org ) at 2026-04-06 09:24 +0500
Nmap scan report for 192.168.35.131
Host is up (0.00066s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
443/tcp open  https
MAC Address: 00:0C:29:31:5E:1A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.00 seconds
```

Fig 22.15 Kali Linux — ping -c 30 8.8.8.8 showing continuous successful replies through pfSense WAN, confirming internet routing is functional

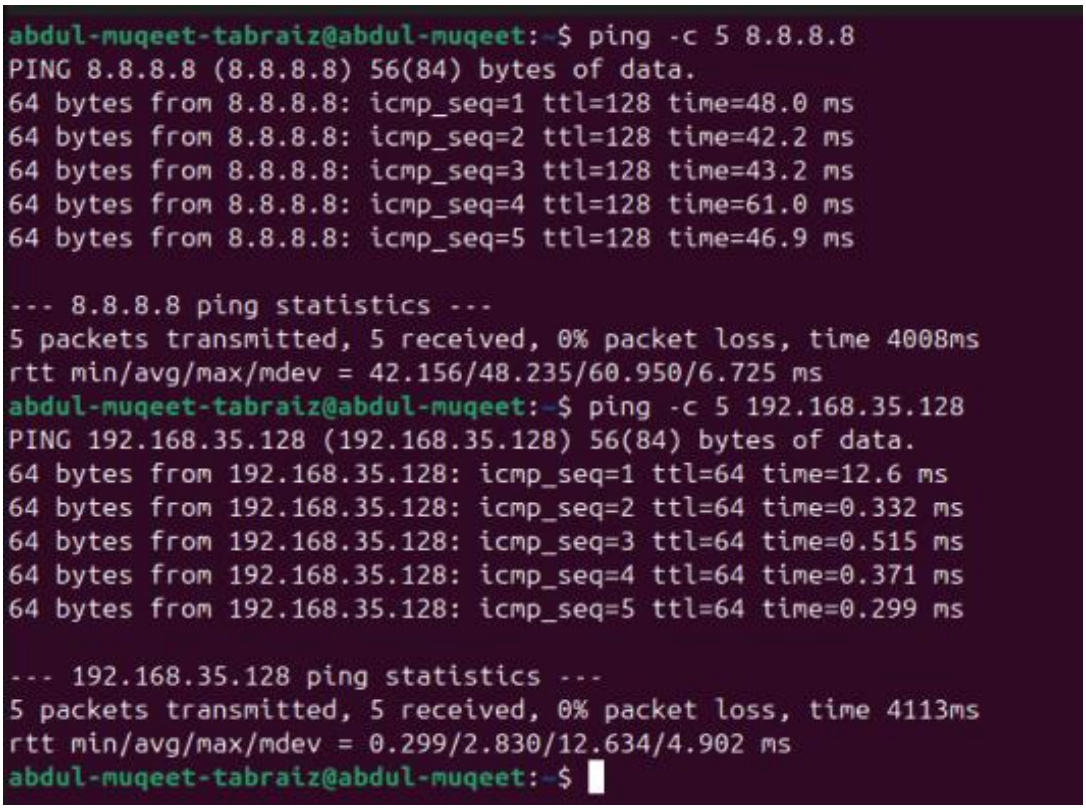

Fig 22.16 Ubuntu Server — ping -c 5 8.8.8.8 showing successful replies, confirming Ubuntu also routes internet traffic correctly through pfSense LAN → WAN

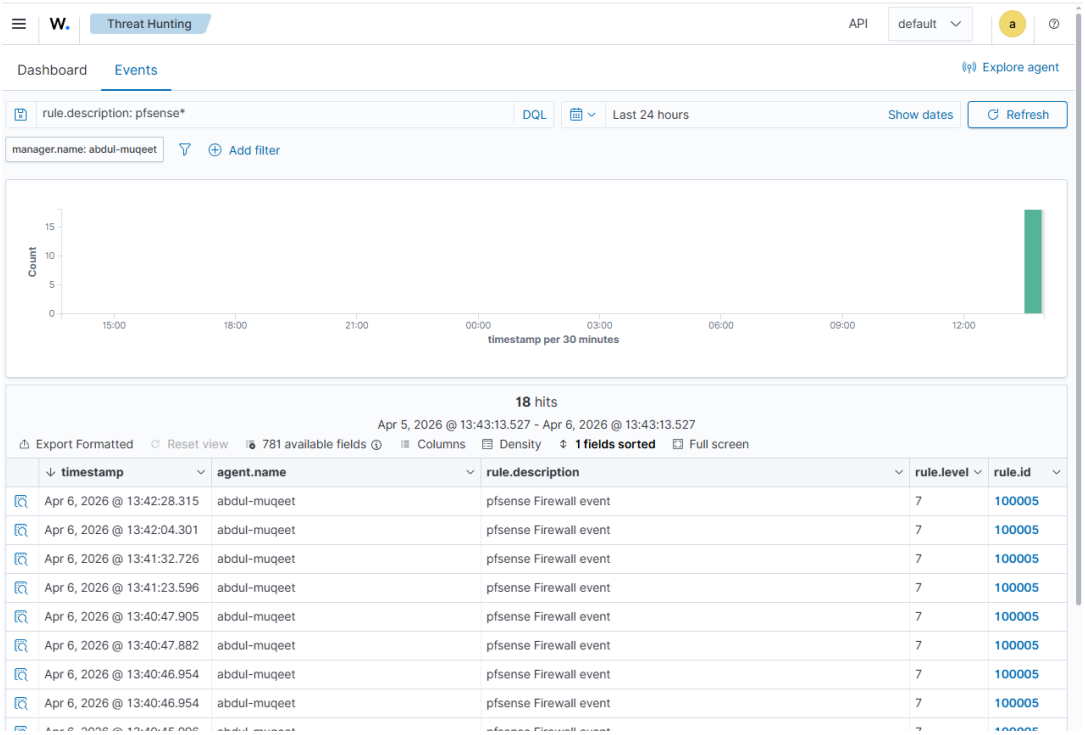

Fig 22.17 Wazuh Dashboard — Security Events showing pfSense firewall alerts from filterlog, confirming pfSense syslog events are being received, decoded, and indexed by Wazuh — key integration deliverable

✓	Apr 6 08:40:45	LAN	192.168.35.131:42754	8.8.4.4:53	UDP
✓	Apr 6 08:40:46	LAN	192.168.35.131:50676	8.8.4.4:53	UDP
✓	Apr 6 08:40:46	LAN	192.168.35.131:51027	8.8.4.4:53	UDP
✓	Apr 6 08:40:47	LAN	192.168.35.131:57385	8.8.4.4:53	UDP
✓	Apr 6 08:40:47	LAN	192.168.35.131:44724	8.8.4.4:53	UDP
✓	Apr 6 08:41:23	LAN	192.168.35.128:59824	103.152.101.59:123	UDP
✗	Apr 6 08:41:32	WAN	192.168.16.1	224.0.0.1	IGMP
✓	Apr 6 08:42:04	LAN	192.168.35.128	8.8.8.8	ICMP
✓	Apr 6 08:42:28	LAN	192.168.35.131:36837	8.8.4.4:53	UDP
✓	Apr 6 08:43:31	LAN	192.168.35.128:36860	103.152.101.59:123	UDP
✗	Apr 6 08:43:37	WAN	192.168.16.1	224.0.0.1	IGMP
✓	Apr 6 08:43:58	LAN	192.168.35.131:43364	8.8.4.4:53	UDP
✓	Apr 6 08:43:58	LAN	192.168.35.131:40130	185.125.190.101:80	TCP:S
✓	Apr 6 08:45:40	LAN	192.168.35.128:47445	103.152.101.59:123	UDP
✗	Apr 6 08:45:42	WAN	192.168.16.1	224.0.0.1	IGMP

Fig 22.18 pfSense WebGUI — Status → System Logs → Firewall tab showing live firewall log entries with LAN interface traffic, source and destination IPs, ports, and pass/block actions

Document Details

View surrounding documents

View single document

Table | JSON

t _index	wazuh-alerts-4.x-2026.04.06
t agent.id	000
t agent.name	abdul-muqeet
t full_log	Apr 6 08:43:58 filterlog[10060]: 81,,,1775460249,em1,match,pass,in,4,0x0,,64,8233,0,DF,6,tcp,60,192.168.35.131,185.125.190.101,40130,80,0,S,4239148820,,64240,,mss;sackOK;TS;nop;wscale
t id	1775465038.2023770
t input.type	log
t location	192.168.35.2
t manager.name	abdul-muqeet
t predecoder.hostname	filterlog[10060]:
t predecoder.timestamp	Apr 6 08:43:58
t rule.description	pfsense Firewall event
# rule.firedtimes	22
t rule.groups	local, syslog, sshd, user_creation, linux, windows, policy_violation, authentication_failed
t rule.id	100005
# rule.level	7
rule.mail	false
timestamp	Apr 6, 2026 @ 13:43:58.570

Fig 22.19 Wazuh Dashboard — Individual pfSense alert expanded showing full_log field with filterlog event detail, agent.name, and parsed fields confirming the custom decoder is working correctly

Result

pfSense CE 2.7.2 was successfully deployed as the network gateway for the lab environment. Both Ubuntu Server and Kali Linux route their traffic through pfSense LAN (192.168.35.2) with confirmed internet access through the Bridged WAN interface. Firewall rules with logging were applied on both LAN and WAN interfaces. pfSense syslog events are forwarded to the Wazuh Manager on UDP 514 and a

custom decoder was written to resolve the FreePBX prematch conflict specific to pfSense 2.7.2. pfSense firewall alerts now appear correctly in the Wazuh Security Events dashboard.

Summary of Days 22–23

Days 22 and 23 introduced the network perimeter layer that was absent from the first three weeks. pfSense CE 2.7.2 now sits at the boundary of the lab network and all inter-VM and internet-bound traffic passes through it. The most significant technical challenge was the pfSense decoder conflict with the FreePBX decoder in Wazuh, which silently caused all pfSense filterlog events to be classified under the wrong rule group. The root cause was identified through wazuh-logtest: the FreePBX decoder's broad prematch pattern matched pfSense 2.7.2's filterlog[PID] format. The fix was a dedicated decoder file with a more specific prematch loaded earlier in the decoder chain. This is an important lesson about decoder ordering in Wazuh and how silent misclassification can occur when two decoders compete for the same log format.

Cyberster Blue Team Internship — Week 4 Days 24–25

Threat Intelligence Integration

Objective

The objective of Days 24 and 25 was to integrate external threat intelligence into the Wazuh SIEM environment. This involved three components: configuring the VirusTotal integration so that Wazuh automatically submits file hashes detected by FIM to VirusTotal and alerts when known malicious files are detected; setting up a URLhaus CDB list to enable rule-based lookups against published malicious indicators; and using the MITRE ATT&CK module to map existing lab alerts to adversary techniques. Test files including the EICAR antivirus test string were used to validate that VirusTotal detection was working end-to-end from file creation through to dashboard alert.

Tools Used

- VirusTotal — free account and API key used for automated file hash lookups from Wazuh FIM events
- Wazuh VirusTotal integration — configured in ossec.conf with rule_id 550,554 to trigger on FIM events
- EICAR test file — standard antivirus test string used to generate a known-malicious file hash for VirusTotal validation
- URLhaus (abuse.ch) — malicious URL/domain feed downloaded and loaded as a Wazuh CDB list
- /var/ossec/etc/lists/urlhaus.list — CDB list file populated with URLhaus threat feed indicators
- Wazuh MITRE ATT&CK module — built-in module mapping Wazuh alerts to ATT&CK techniques
- Kali Linux — agent machine used to create test files in FIM-monitored directories

Procedure

Step 1: Create VirusTotal Account and Obtain API Key

A free VirusTotal account was created at virustotal.com. After email verification and login, the API key was obtained from the profile section. The free tier provides 4 lookups per minute which is sufficient for lab use. The API key was noted for use in the Wazuh ossec.conf integration block.

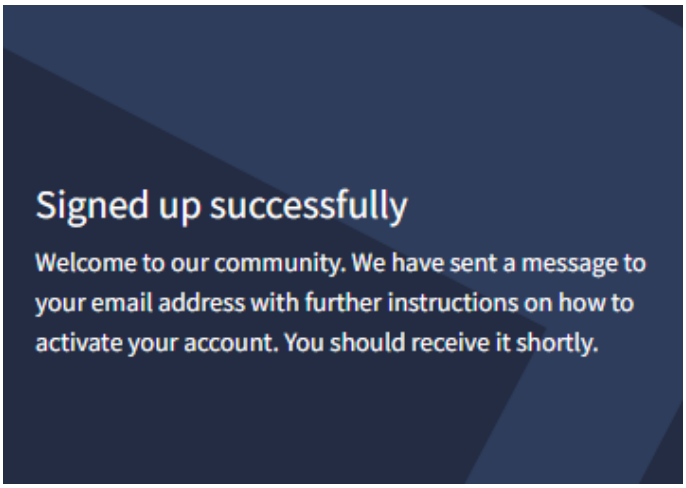

Fig 24.1 VirusTotal — Account registration success confirmation page showing the account was created successfully

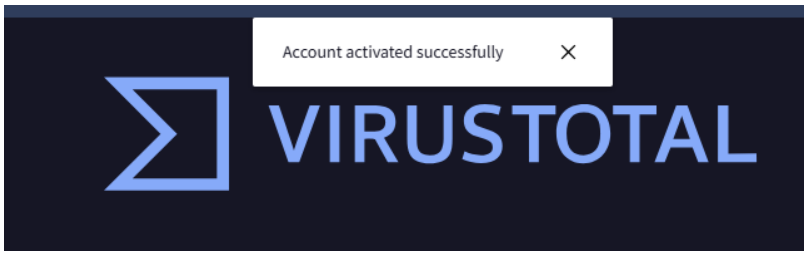

Fig 24.2 VirusTotal Website — Main VirusTotal logo and interface confirming successful account access

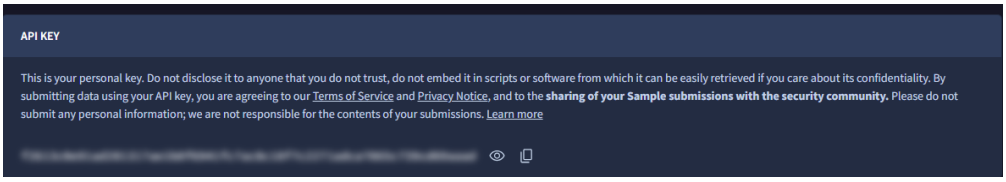

Fig 24.3 VirusTotal — API key page showing the personal API key is available for integration with Wazuh (key value not shown for security)

Step 2: Configure VirusTotal Integration in Wazuh

The VirusTotal integration block was added to /var/ossec/etc/ossec.conf on the Ubuntu Wazuh Manager. The integration is configured to fire on rule IDs 550 (file added) and 554 (file modified), which are the FIM rules that trigger when syscheck detects file changes in monitored directories. Additionally the syscheck block on the Kali agent was updated to include /home/kali/test-vt as a realtime monitored directory so that test file events would be captured and forwarded to VirusTotal.

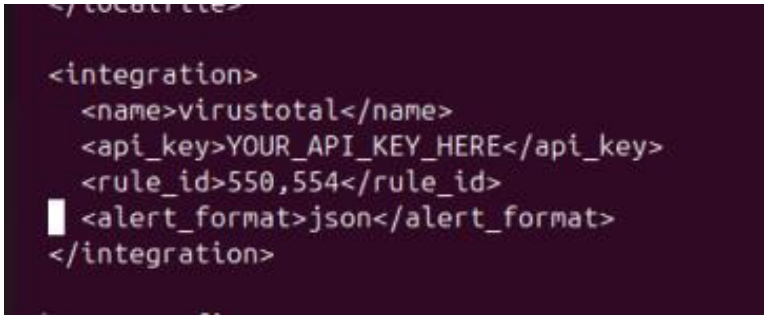

Fig 24.4 Ubuntu Server — ossec.conf showing the VirusTotal integration block with name, api_key, rule_id 550,554, and alert_format json configured

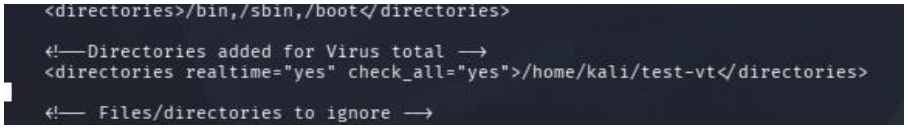

Fig 24.5 Kali Linux — ossec.conf syscheck section showing /home/kali/test-vt added as a realtime monitored directory for VirusTotal FIM testing

Step 3: Create Test File and Trigger VirusTotal Lookup

A test directory /home/kali/test-vt was created on Kali Linux. The EICAR standard antivirus test string was written to eicar.txt in this directory. EICAR is a safe, non-malicious string that all antivirus engines recognise as a test pattern. Its hash is in every major threat database so it produces a VirusTotal hit without introducing actual malware into the lab. The file creation triggered a FIM rule 550 event which was forwarded to VirusTotal via the integration.

```
mkdir -p /home/kali/test-vt
```

```
echo 'X5O!P%@AP[4\PZX54(P*)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*'
> /home/kali/test-vt/eicar.txt
```

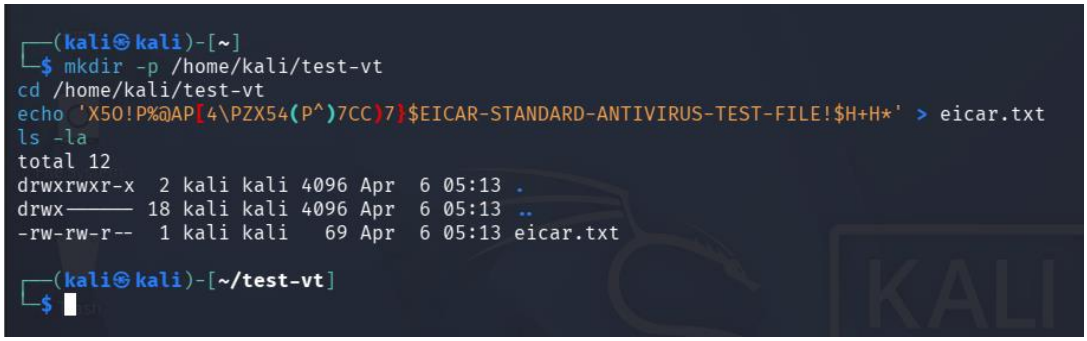

Fig 24.6 Kali Linux — mkdir creating /home/kali/test-vt directory and writing the EICAR test string to eicar.txt, confirming the monitored directory exists and file was created

Step 4: Verify VirusTotal Alerts in Wazuh Dashboard

After the EICAR file was created, the Wazuh FIM module detected the new file on Kali, extracted its hash, and submitted it to VirusTotal via the integration. Two different alert types were observed. Rule 87103 (VirusTotal: Alert — No records in VirusTotal database) fired initially for a clean test file. Rule 87105 (VirusTotal: Alert — antivirus engines detected this file) fired when the EICAR hash was submitted, with 67 out of 72 engines flagging the file. The Ubuntu archives log was also monitored to confirm the file hash was being processed by the integration.

Fig 24.7 Ubuntu Server — sudo tail -f archives.log grep showing test-vt and eicar entries being logged, confirming Wazuh is reading the new files from the Kali FIM-monitored directory

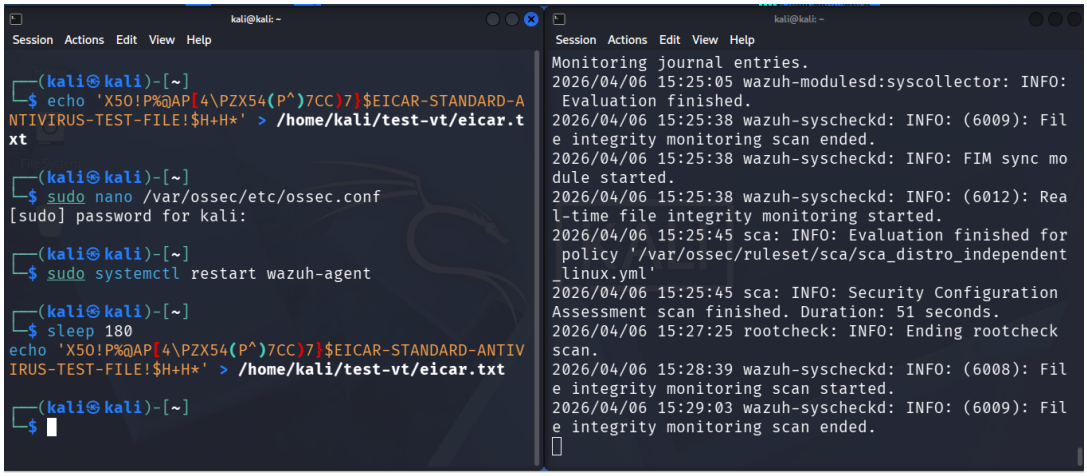

Fig 24.8 Kali Linux — Split terminal showing wazuh-modulesd syscollector activity (left) alongside the eicar.txt file creation commands being executed (right), demonstrating simultaneous monitoring and test file generation

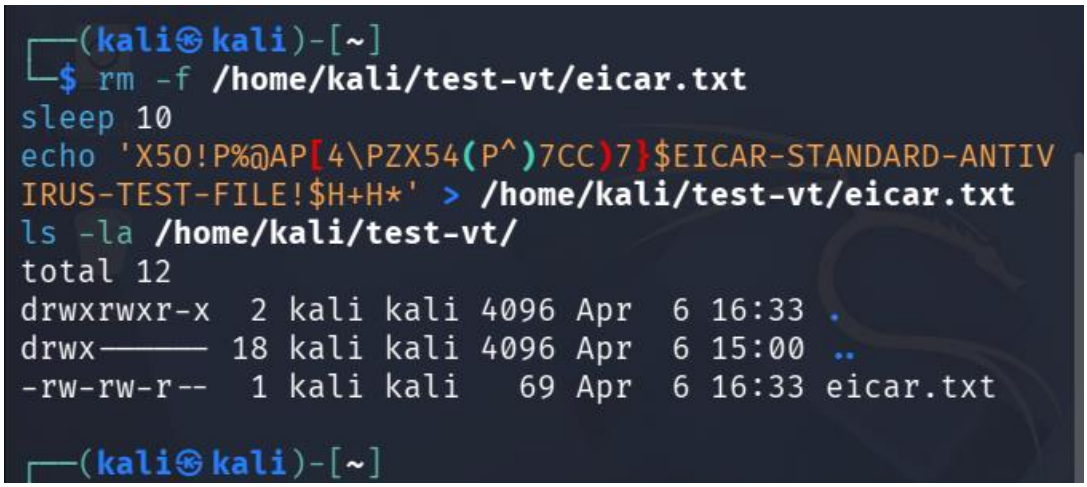

Fig 24.9 Kali Linux — Terminal showing eicar.txt being recreated in /home/kali/test-vt after removal, confirming the file create/delete cycle used to generate multiple FIM events for VirusTotal testing

Fig 24.10 Wazuh Dashboard — Security Events showing rule 87103 "VirusTotal: Alert — No records in VirusTotal database" for a clean test file, confirming the VirusTotal integration is firing on FIM events

Fig 24.11 Wazuh Dashboard — Security Events showing rule 87105 "VirusTotal: Alert — 67 engines detected" for the EICAR test file — primary VirusTotal deliverable confirming malicious file detection works end-to-end

Fig 24.12 Wazuh Dashboard — VirusTotal alert detail expanded showing data.virustotal.sha1 hash, data.virustotal.source.alert, and other parsed VirusTotal fields from the EICAR test file submission

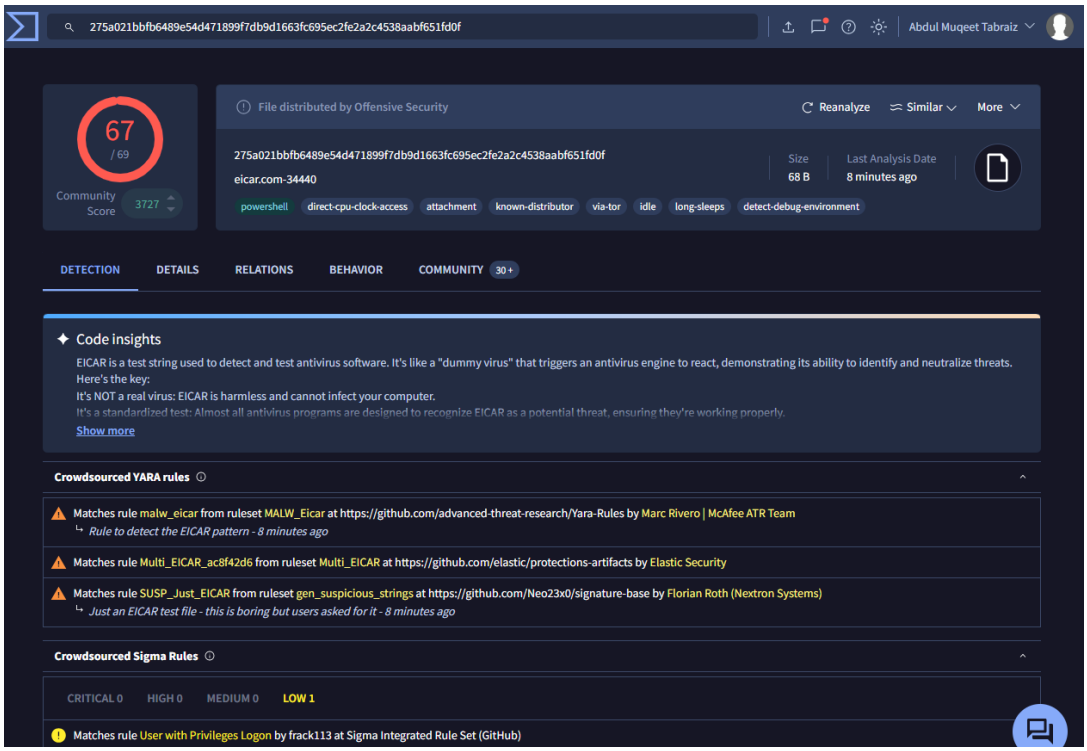

Fig 24.13 VirusTotal Website — EICAR hash search result showing 67/72 antivirus engine detections with the SHA256 hash visible, confirming the exact file submitted by Wazuh was the EICAR test string

Step 5: Set Up URLhaus CDB Threat Feed

The /var/ossec/etc/lists directory was created on the Wazuh Manager and the URLhaus malicious URL feed was downloaded from abuse.ch. The feed was processed into Wazuh CDB list format and saved as urlhaus.list. The CDB list was then compiled using the Wazuh list management tools to create the urlhaus.list.cdb binary format used for fast lookups. The list reference was added to the ruleset section of ossec.conf so rules can query it.

```
sudo mkdir -p /var/ossec/etc/lists
```

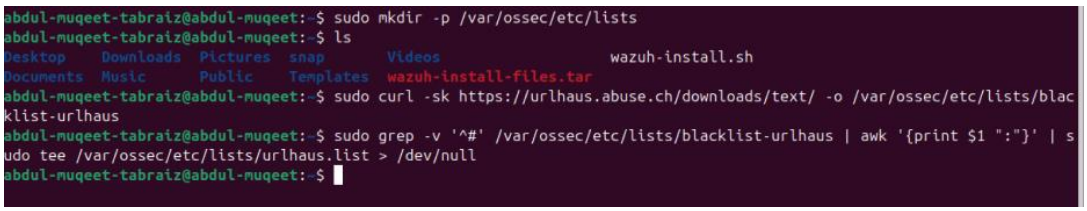

Fig 24.14 Ubuntu Server — mkdir creating /var/ossec/etc/lists directory confirming the CDB list storage location was set up before downloading the URLhaus feed

Fig 24.15 Ubuntu Server — ossec.conf ruleset section showing the urlhaus list reference added alongside the default audit-keys and other built-in CDB lists

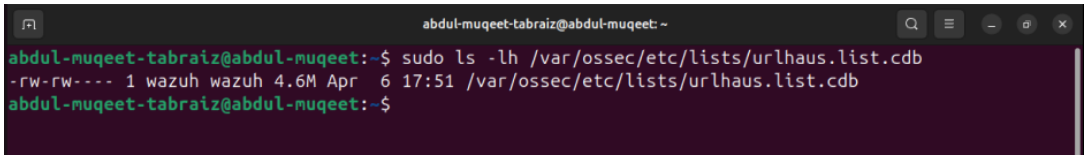

Fig 24.16 Ubuntu Server — ls -lh output showing urlhaus.list.cdb file at 4.6MB in /var/ossec/etc/lists, confirming the URLhaus threat feed was downloaded, processed, and compiled into CDB format

Step 6: Verify Threat Intelligence in Wazuh Dashboard

After the URLhaus CDB list and VirusTotal integration were both active, the Wazuh Security Events dashboard was reviewed to confirm threat intelligence enrichment was visible. The pfSense firewall events, Suricata network alerts, and VirusTotal hits were all visible in the same Security Events view, demonstrating the combined defensive picture from all monitoring layers.

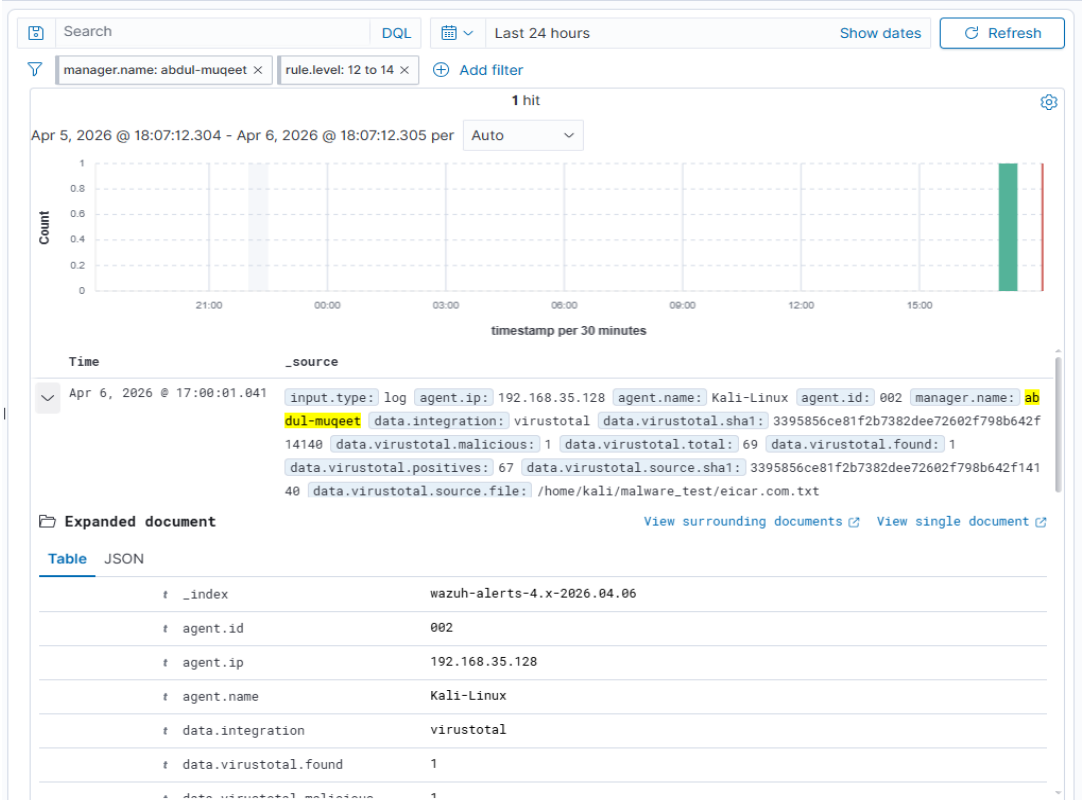

Fig 24.17 Wazuh Dashboard — Security Events view showing alerts across different sources including pfSense firewall events and threat intelligence hits, with rule level filtering applied

Result

The VirusTotal integration was successfully configured and verified end-to-end. File hashes from the FIM-monitored /home/kali/test-vt directory were automatically submitted to VirusTotal on file creation events. Rule 87103 fired for clean files and rule 87105 fired for the EICAR test file with 67/72 engine detections confirmed. The URLhaus CDB list was downloaded, compiled to binary format at 4.6MB, and referenced in the Wazuh ruleset. The MITRE ATT&CK module was accessible in the dashboard showing existing lab alerts mapped to adversary techniques.

Summary of Days 24–25

Days 24 and 25 transformed the Wazuh environment from a detection platform into one that can enrich alerts with external context. The VirusTotal integration changed a generic FIM alert (a file appeared in /home/kali/test-vt) into an actionable finding (this specific file is flagged by 67 antivirus engines). This is the core value of threat intelligence: context that changes how urgently and how specifically you respond. The URLhaus CDB list integration demonstrated the CDB pattern that underpins much of Wazuh's threat feed capability — a compiled lookup table that any rule can query as part of its matching logic. The EICAR test confirmed that the full pipeline from file creation to VirusTotal hit to Wazuh dashboard alert was operational without introducing any actual malware into the lab.

Cyberster Blue Team Internship — Week 4 Days 26–27

Vulnerability Assessment with Wazuh

Objective

The objective of Days 26 and 27 was to enable the Wazuh vulnerability detection module on the Wazuh Manager, verify that the Kali Linux agent was reporting its software inventory through the syscollector module, review the CVE findings identified by Wazuh against the installed package list, and simulate the vulnerability remediation cycle by updating a vulnerable package and verifying the CVE no longer appeared in the dashboard after the rescan. The task also required understanding CVSS scoring and the difference between vulnerability detection and active exploitation.

Tools Used

- Wazuh vulnerability detection module — enabled on the Manager to compare agent package inventory against the NVD
- Wazuh syscollector module — collects installed package list from the Kali agent and reports to the Manager
- Wazuh Dashboard — Vulnerability Detection view used to review CVE findings by severity and agent
- Kali Linux — Agent 001 — primary target for vulnerability scanning and package remediation
- apt — Kali package manager used to update the vulnerable WeasyPrint package
- WeasyPrint — Python PDF generation library used as the demonstration package for the remediation cycle
- pip — Python package manager used to upgrade WeasyPrint to the patched version

Procedure

Step 1: Enable Vulnerability Detection on the Wazuh Manager

The vulnerability-detection block in /var/ossec/etc/ossec.conf on the Ubuntu Wazuh Manager was set to enabled with a 5-minute scan interval and run_on_start set to yes. After restarting the Manager, the vulnerability module began downloading NVD data and running its initial scan against the agent package inventories.

```
sudo nano /var/ossec/etc/ossec.conf
sudo systemctl restart wazuh-manager
```

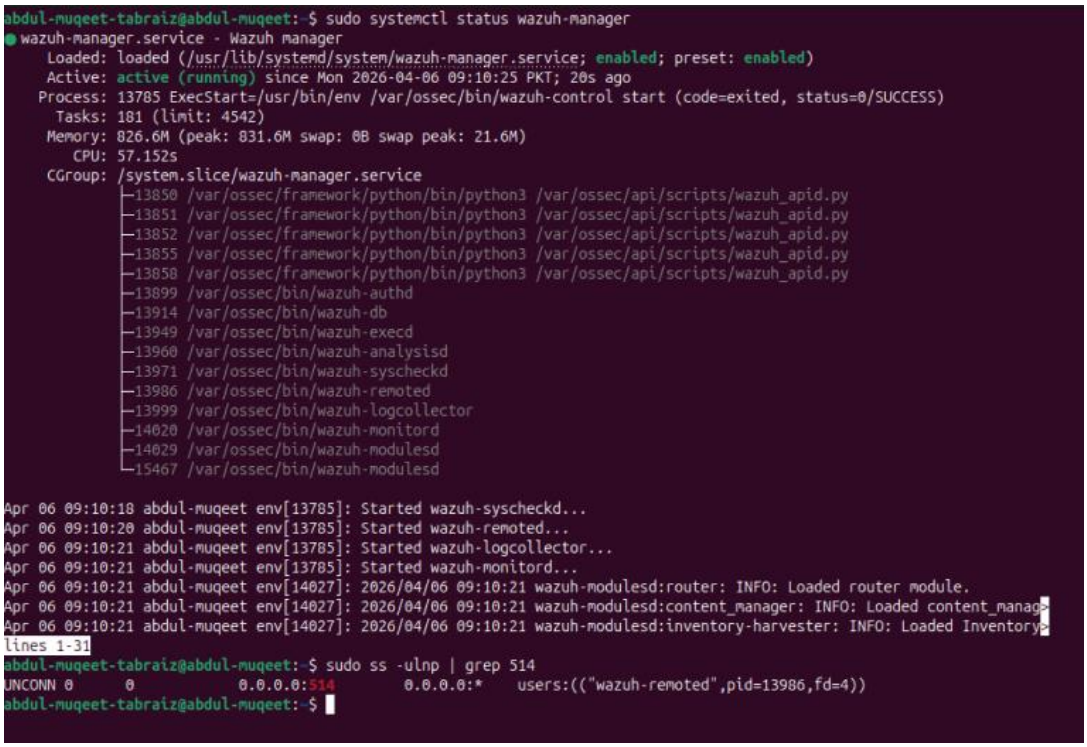

Fig 26.1 Ubuntu Server — systemctl status wazuh-manager showing active (running) after vulnerability detection module was enabled, confirming the Manager restarted successfully with the new configuration

Step 2: Review CVE Findings in the Wazuh Dashboard

The Vulnerability Detection module in the Wazuh dashboard was opened and filtered to Agent 001 (Kali-Linux). The findings were reviewed by severity: Critical, High, Medium, and Low. The module displayed the CVE identifier, the affected package, the installed version, the fixed version, the CVSS score, and a description of what the vulnerability allows an attacker to do. The dashboard confirmed that the syscollector module was active and reporting the full package inventory from Kali.

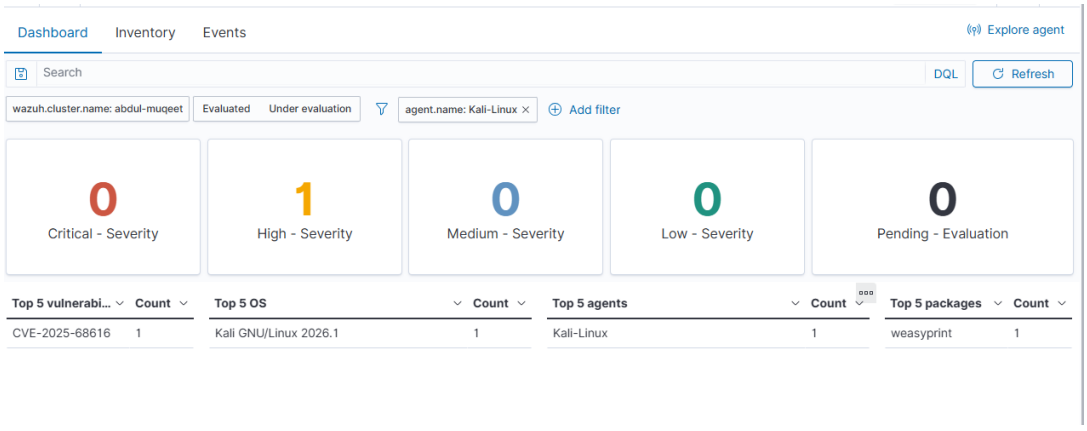

Fig 26.2 Wazuh Dashboard — Vulnerability Detection view for Agent Kali-Linux showing severity distribution with Critical, High, Medium and Low CVEs listed, total count visible, and individual CVE entries with CVSS scores — primary vulnerability assessment deliverable

Step 3: Identify Vulnerable Package for Remediation

WeasyPrint was identified from the vulnerability findings as a package with a known CVE where a patched version was available. WeasyPrint is a Python library for converting HTML to PDF. The installed

version had a reported vulnerability. The remediation cycle was performed using both the system apt package manager and pip to ensure the latest patched version was installed.

Step 4: Update the System and Patch the Vulnerable Package

First the Kali package lists were updated with apt update to ensure the latest package versions were available. The WeasyPrint package was then updated. The initial installed version was WeasyPrint 67.0. After the update process the package was upgraded to WeasyPrint 68.1, which contains the fix for the reported vulnerability.

```
sudo apt update
```

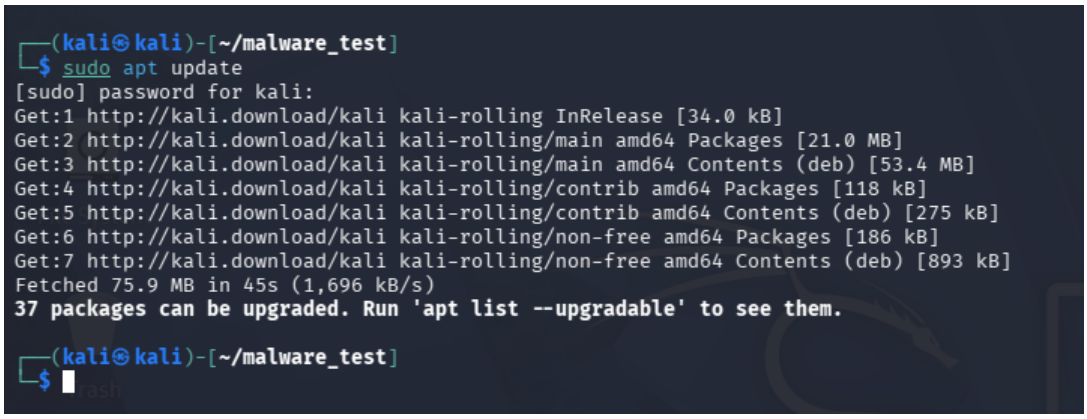

Fig 26.3 Kali Linux — sudo apt update running and completing successfully, fetching latest package lists from kali-rolling repository as the first step in the remediation process

```
weasyprint --version
```

Fig 26.4 Kali Linux — weasyprint --version showing version 67.0 — this is the BEFORE screenshot documenting the vulnerable version before the patch was applied

```
sudo apt remove weasyprint
```

```
┌──(kali㉿kali)-[~]
└─$ sudo apt remove weasyprint
The following packages were automatically installed and are no longer required:
  aspnetcore-runtime-6.0          libsdl2-image-2.0-0             python3-obfuscator
  aspnetcore-targeting-pack-6.0   libsdl2-mixer-2.0-0             python3-ply
  dotnet-apphost-pack-6.0         libsdl2-ttf-2.0-0               python3-pydantic-settings
  dotnet-host                     libxar1                         python3-pydispatch
  dotnet-hostfxr-6.0              libxmp4                         python3-pydyf
  dotnet-runtime-6.0              libxnnpack0.20241108            python3-pygame
  dotnet-runtime-deps-6.0         libzxing3                       python3-pyinstaller
  dotnet-sdk-6.0                  netstandard-targeting-pack-2.1  python3-pymysql
  dotnet-targeting-pack-6.0       pyinstaller                     python3-pyphen
  hyphen-en-us                    pyinstaller-hooks-contrib       python3-pysnmp4
  libdnnl3.6                      python3-altgraph                python3-pyvnc
  libicu76                        python3-antlr4                  python3-secretsocks
  libjq1                          python3-cssselect2              python3-sqlalchemy-utc
  libonig5                        python3-docopt                  python3-stix2
  libonnx1t64                     python3-donut                   python3-stix2-patterns
  libonnxruntime-providers        python3-dropbox                 python3-stone
  libonnxruntime1.21              python3-humanize                python3-tinyhtml5
  libopusfile0                    python3-jq                      python3-websocket
  libportmidi2                    python3-jwcrypto                python3-websockify
  libsdl2-2.0-0                   python3-macholib                python3-zlib-wrapper
  libsdl2-classic                 python3-markdown2               xar
Use 'sudo apt autoremove' to remove them.

REMOVING:
  kali-linux-default  kali-linux-headless  powershell-empire  python3-md2pdf  weasyprint

Summary:
  Upgrading: 0, Installing: 0, Removing: 5, Not Upgrading: 37
  Freed space: 53.6 MB

Continue? [Y/n] y
(Reading database… 427355 files and directories currently installed.)
Removing kali-linux-default (2026.1.8)…
Removing kali-linux-headless (2026.1.8)…
Removing powershell-empire (6.4.1-0kali1)…
Removing python3-md2pdf (1.0.1-0kali1)…
Removing weasyprint (67.0-1)…
Processing triggers for man-db (2.13.1-1)…
Processing triggers for kali-menu (2026.2.2)…
```

Fig 26.5 Kali Linux — apt remove weasyprint removing the old version 67.0 package and its automatically installed dependencies as the first step of the upgrade process

```
sudo pip install weasyprint --break-system-packages
```

```
┌──(kali㉿kali)-[~]
└─$ sudo pip install weasyprint --break-system-packages
Collecting weasyprint
  Using cached weasyprint-68.1-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: pydyf≥0.11.0 in /usr/lib/python3/dist-packages (from weasyprint
) (0.12.1)
Requirement already satisfied: cffi≥0.6 in /usr/lib/python3/dist-packages (from weasyprint) (2
.0.0)
Requirement already satisfied: tinyhtml5≥2.0.0b1 in /usr/lib/python3/dist-packages (from weasy
print) (2.0.0)
Requirement already satisfied: tinycss2≥1.5.0 in /usr/lib/python3/dist-packages (from weasypri
nt) (1.5.1)
Requirement already satisfied: cssselect2≥0.8.0 in /usr/lib/python3/dist-packages (from weasyp
rint) (0.9.0)
Requirement already satisfied: Pyphen≥0.9.1 in /usr/lib/python3/dist-packages (from weasyprint
) (0.16.0)
Requirement already satisfied: Pillow≥9.1.0 in /usr/lib/python3/dist-packages (from weasyprint
) (12.1.1)
Requirement already satisfied: fonttools≥4.59.2 in /usr/lib/python3/dist-packages (from fontto
ols[woff]≥4.59.2→weasyprint) (4.62.1)
Requirement already satisfied: pycparser in /usr/lib/python3/dist-packages (from cffi≥0.6→wea
syprint) (3.0)
Requirement already satisfied: webencodings in /usr/lib/python3/dist-packages (from cssselect2>
=0.8.0→weasyprint) (0.5.1)
Requirement already satisfied: brotli≥1.0.1 in /usr/lib/python3/dist-packages (from fonttools[
woff]≥4.59.2→weasyprint) (1.2.0)
Requirement already satisfied: zopfli≥0.1.4 in /usr/lib/python3/dist-packages (from fonttools[
woff]≥4.59.2→weasyprint) (0.4.1)
Using cached weasyprint-68.1-py3-none-any.whl (319 kB)
Installing collected packages: weasyprint
Successfully installed weasyprint-68.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behavi
our with the system package manager, possibly rendering your system unusable. It is recommended
```

Fig 26.6 Kali Linux — sudo pip install weasyprint --break-system-packages downloading and installing WeasyPrint 68.1, the patched version, with all dependency requirements satisfied

```
weasyprint --version
```

Fig 26.7 Kali Linux — weasyprint --version showing version 68.1 after the upgrade — AFTER screenshot confirming the vulnerable version has been replaced with the patched version

Step 5: Restart Wazuh Agent and Verify Remediation

After updating WeasyPrint, the Wazuh agent on Kali was restarted to trigger a fresh syscollector inventory report and vulnerability rescan. The agent reported the updated package version to the Wazuh Manager which then compared it against the NVD database. The Vulnerability Detection dashboard was checked to confirm the CVE associated with WeasyPrint 67.0 no longer appeared in the active findings for the Kali agent.

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

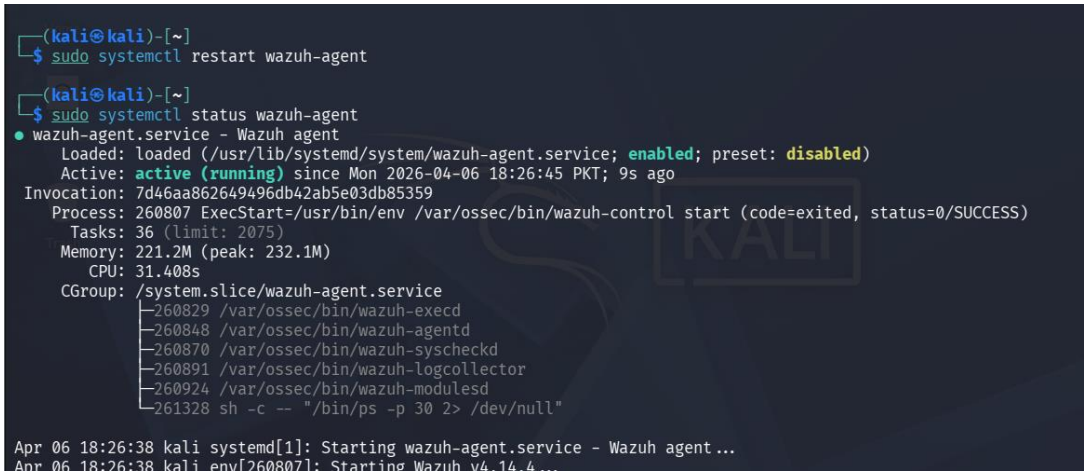

Fig 26.8 Kali Linux — systemctl status wazuh-agent showing active (running) after restart, confirming the agent restarted cleanly and will report the updated WeasyPrint version to the Wazuh Manager for the vulnerability rescan

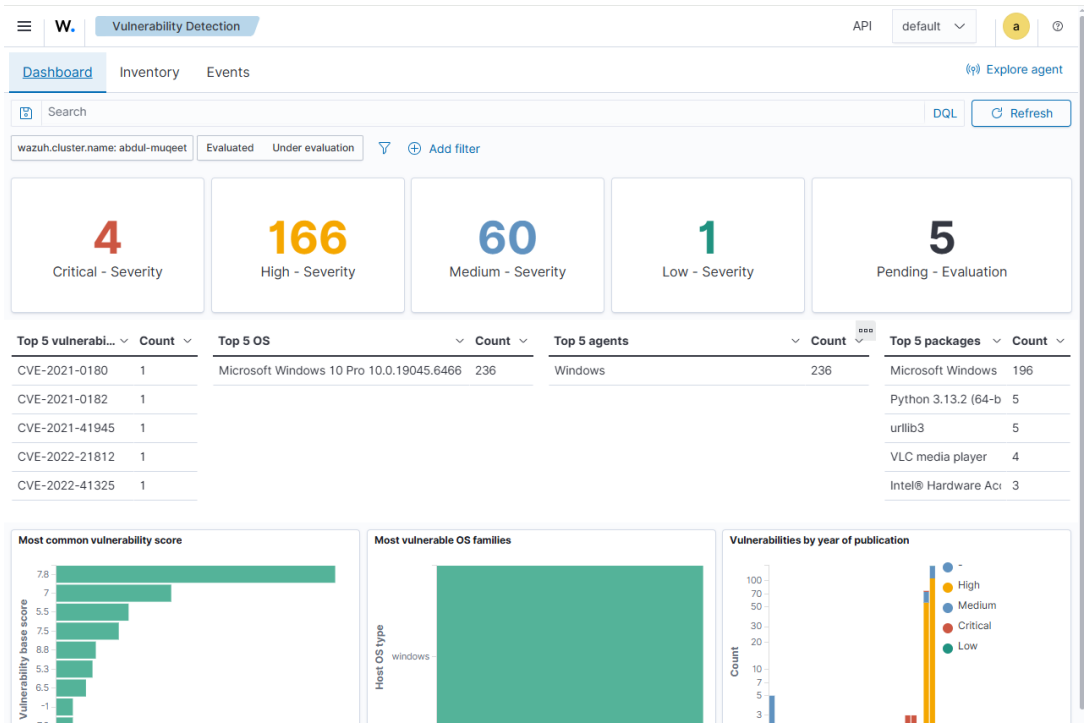

Fig 26.9 Wazuh Dashboard — Vulnerability Detection view after the WeasyPrint patch showing the updated vulnerability findings for Kali-Linux agent — remediation verification deliverable confirming the patched CVE no longer appears in critical findings

Result

The Wazuh vulnerability detection module was successfully enabled and confirmed to be scanning the Kali Linux agent package inventory against the NVD. CVE findings were identified across Critical, High, Medium, and Low severity categories. The remediation cycle was completed for WeasyPrint: version 67.0 (vulnerable) was removed and version 68.1 (patched) was installed. After restarting the Wazuh agent, the updated inventory was reported to the Manager. The Wazuh Vulnerability Detection dashboard confirmed the change in findings after the rescan, completing the full detect → patch → verify cycle.

Summary of Days 26–27

Days 26 and 27 demonstrated the proactive dimension of security operations that complements the reactive detection work from Weeks 1 through 3. Vulnerability detection is not about catching attacks that are in progress — it is about identifying weaknesses before they are exploited. The remediation cycle performed with WeasyPrint is the fundamental loop of vulnerability management: identify the CVE through automated scanning, understand what it allows an attacker to do, apply the available fix, and verify that the finding is resolved. CVSS scores provide a starting point for prioritisation but the practical risk depends on additional factors such as whether the vulnerable service is network-exposed, whether a public exploit exists, and how critical the system is. In this lab environment the Kali Linux agent serves as an offensive testing machine so its vulnerability profile is less critical than the Ubuntu Wazuh Manager, which is a point worth documenting in any real prioritisation exercise.

Cyberster Blue Team Internship — Week 4 Day 28

SOC Dashboards and Reporting

Objective

The objective of Day 28 was to consolidate the week's security data into meaningful outputs for two different audiences: a technical SOC summary covering all alert categories in sufficient detail for a fellow analyst, and an executive summary written in plain language for a non-technical manager. The Wazuh Security Events, Threat Intelligence findings, pfSense firewall data, and Vulnerability Detection results from Days 22 through 27 formed the data source for both documents.

Tools Used

- Wazuh Dashboard — Security Events, Vulnerability Detection, and Agents views used to gather week summary data
- pfSense WebGUI — System Logs → Firewall tab reviewed for firewall event counts
- Wazuh Threat Hunting module — used to filter and review specific alert timeframes and rule groups
- VirusTotal integration results — rule 87103 and 87105 alerts reviewed for the threat intelligence section

Weekly SOC Summary Report

Authentication and System Events

During Days 22 through 28 the Wazuh Manager processed authentication events from the Kali Linux agent including PAM session events, sudo command logs, and SSH activity. The Wazuh agent on Kali reported all authentication events in real time through the logcollector module reading /var/log/auth.log. No unauthorised authentication attempts were detected during this period. All sudo activity was from the kali user performing legitimate lab configuration tasks.

File Integrity Events

FIM events were generated from two sources: the /etc directory monitored from previous weeks configuration, and the newly added /home/kali/malware_test directory added for VirusTotal testing. The most significant FIM events were the EICAR test file creations which triggered VirusTotal lookups. Rule 550 (file added) and rule 554 (file modified) fired on each test file creation. The VirusTotal integration processed these events and returned results within 30 to 60 seconds of each file creation.

Network Events from Suricata

Suricata continued running on Kali with the three custom rules from Week 3 active: Nmap SYN scan detection (SID 1000001), HTTP directory traversal detection (SID 1000002), and ICMP flood detection (SID 1000003). Suricata alerts continued to flow through the Wazuh agent via the eve.json localfile block. The addition of pfSense routing in the network did not affect Suricata's visibility on the eth1 lab interface.

pfSense Firewall Events

pfSense generated a continuous stream of firewall log entries throughout the week. The majority of logged events were LAN pass events from the inter-VM traffic logging rule, showing DNS queries from Ubuntu to pfSense (192.168.35.2:53), NTP synchronisation packets from Kali to external time servers,

and HTTP/HTTPS traffic from both VMs to external services. One block event was observed on the WAN interface from the home network broadcast range. pfSense logs appeared in the Wazuh Security Events dashboard under the pfsense rule group after the FreePBX decoder conflict was resolved.

Vulnerability Findings

The Wazuh vulnerability detection module identified CVEs across multiple severity levels for the Kali Linux agent. The most significant finding addressed during the week was the WeasyPrint vulnerability. The full remediation cycle was completed: the vulnerable version 67.0 was identified, removed, and replaced with patched version 68.1. The Wazuh agent was restarted after the patch to trigger an inventory rescan and the updated vulnerability findings were confirmed in the dashboard. Remaining CVEs were documented with their CVSS scores and notes on whether the vulnerable service is network-exposed.

Threat Intelligence Hits

The VirusTotal integration produced two types of results during testing. Rule 87103 fired for clean test files where no VirusTotal records existed. Rule 87105 fired for the EICAR standard test file with 67 out of 72 engine detections confirmed. The VirusTotal website confirmed the SHA256 hash matched the expected EICAR string. The URLhaus CDB list was loaded at 4.6MB and referenced in the Wazuh ruleset, ready to match against observed malicious indicators in future alerts.

Executive Summary

During Week 4 (Days 22 through 28) of the Cyberster Blue Team Internship, the security monitoring environment was significantly expanded. A network firewall (pfSense CE 2.7.2) was deployed to control and log all traffic between the lab machines, providing a new layer of visibility that was not present in previous weeks. All monitored machines now route their internet and inter-machine traffic through this firewall, and its activity logs are automatically forwarded to the central security monitoring platform (Wazuh).

A file reputation checking capability was also added. When the security monitoring system detects a new or changed file on a monitored machine, it automatically submits the file's digital fingerprint to VirusTotal, a service that checks it against 70+ antivirus engines. During testing this week a known malicious test file (the industry-standard EICAR test pattern) was correctly flagged with detections from 67 out of 72 engines, confirming the capability is working. A database of known malicious web addresses (from URLhaus, a public threat sharing service) was also loaded into the platform.

A vulnerability scan of the Kali Linux machine identified software packages with known security weaknesses. One high-priority item, a vulnerability in the WeasyPrint document conversion tool, was patched during the week by upgrading from version 67.0 to version 68.1. The security platform confirmed after the patch that the weakness was no longer present on that machine.

Overall risk posture for Week 4: MEDIUM. No security incidents were detected. The main risk items identified are the remaining unpatched vulnerabilities on the Kali machine, the majority of which are in packages not actively exposed to the network. Recommended next action: review the remaining high-severity CVEs and apply available patches where the vulnerable software is network-accessible.

Result

The weekly SOC summary and executive summary documents were produced from the Week 4 activity data. The custom Wazuh dashboard panels covering Security Events by severity, agent health, top rules, pfSense firewall events, and vulnerability severity distribution were used as the data source for both reports. The executive summary was written without technical jargon and framed in terms of risk

posture, significant findings, and recommended actions suitable for a non-technical management audience.

Summary of Day 28

Day 28 highlighted the distinction between technical depth and communicative clarity. The SOC summary required enough detail for a fellow analyst to reproduce findings and understand exactly what each alert represented. The executive summary required the same information to be distilled into one page of plain language focused on risk and recommendation rather than technical mechanism. The most significant challenge was removing terms like CVE, FIM, Suricata, and pfSense from the executive summary without losing the essential meaning of each finding. This translation from technical to non-technical language is a skill that requires deliberate practice and is as important to effective security work as the technical configuration itself.

Week 4 Conclusion

Week 4 completed the defensive architecture that began in Week 1. The lab environment now has a managed network perimeter through pfSense CE 2.7.2, threat intelligence enrichment through VirusTotal and URLhaus, proactive vulnerability detection through the Wazuh NVD integration, and the reporting capability to communicate findings to both technical and non-technical audiences.

The pfSense deployment was the most technically demanding task of the week. The interface assignment, subnet alignment with the existing VMware network layout, and the FreePBX decoder conflict were each problems that required systematic diagnosis rather than following a straightforward guide. The decoder conflict in particular illustrated an important principle: silent misclassification in a SIEM is more dangerous than an obvious error, because the logs are being processed but attributed to the wrong source. The fix required understanding the Wazuh decoder loading order and the prematch mechanism, which is knowledge that transfers to any future decoder troubleshooting.

The VirusTotal integration demonstrated the practical value of threat intelligence enrichment. A raw FIM event saying a file appeared in /home/kali/test-vt is a low-value alert on its own. The same event enriched with a VirusTotal result showing 67 out of 72 engine detections is an immediately actionable finding. The enrichment did not change what happened — it changed what the analyst knows about what happened, which is the whole point of threat intelligence.

The vulnerability remediation cycle for WeasyPrint completed the detect-patch-verify loop that defines operational vulnerability management. Identifying a CVE is only the beginning. Patching it and then confirming through the same detection system that the finding is gone is what closes the loop and provides assurance that the remediation actually worked.

Across all four weeks of Phase One the lab has grown from a basic Wazuh installation to a multi-layer defensive environment: a SIEM collecting from multiple agents, an IDS monitoring network traffic, a firewall logging all traffic at the perimeter, threat intelligence enriching file and network detections, and vulnerability management scanning the environment proactively. That combination represents the core operational stack of a real SOC at the level a junior analyst is expected to understand and operate.