Cyberster

INTERNSHIP REPORT

Week 2: Detection, Custom Rules, and Log Analysis

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 2 Days 8–9

File Integrity Monitoring (FIM)

Objective

The objective of Days 8 and 9 was to configure File Integrity Monitoring (FIM) using Wazuh's syscheck module on both the Linux (Kali) and Windows agent machines. FIM enables the SOC to detect any unauthorized creation, modification, or deletion of files in critical system directories, providing real-time alerting for potential intrusions or insider threats.

Tools Used

- Wazuh Manager (Ubuntu) — central SIEM receiving and processing FIM alerts
- Kali Linux Agent — Linux endpoint with syscheck configured for /etc and /var/log
- Windows Host Agent — Windows endpoint with syscheck configured for C:\Users\Public
- /var/ossec/etc/ossec.conf — Wazuh agent configuration file edited on Kali and Windows
- Wazuh Dashboard — for viewing and verifying FIM alerts in Integrity Monitoring module

Procedure

Step 1: Verify Wazuh Manager and Agents are Active

Before configuring FIM, the Wazuh Manager on Ubuntu was verified to be running and both agents (Kali Linux and Windows) were confirmed active using the agent_control utility.

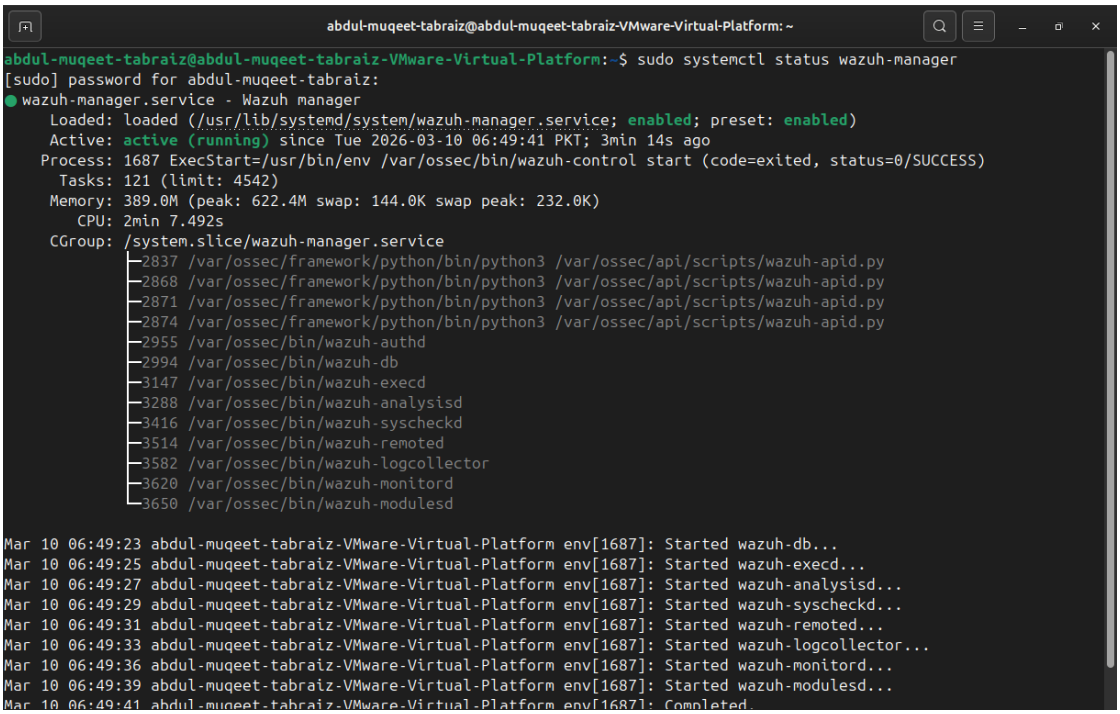

Fig: 8.1 Wazuh Manager service status on Ubuntu — all subprocesses active and running

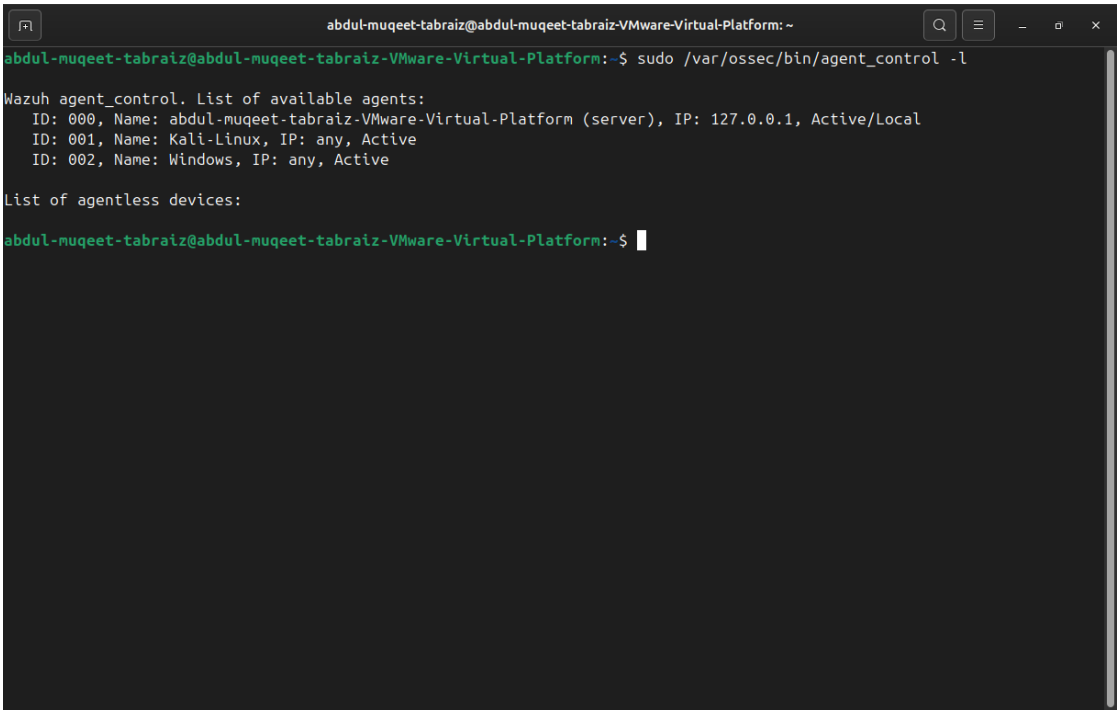

Fig: 8.2 agent_control -l output — Kali-Linux (ID 001) and Windows (ID 002) both listed as Active

Step 2: Open Kali Agent Configuration File

The Kali Linux agent's ossec.conf was opened to edit the syscheck block. This file controls which directories FIM monitors and how frequently scans run.

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

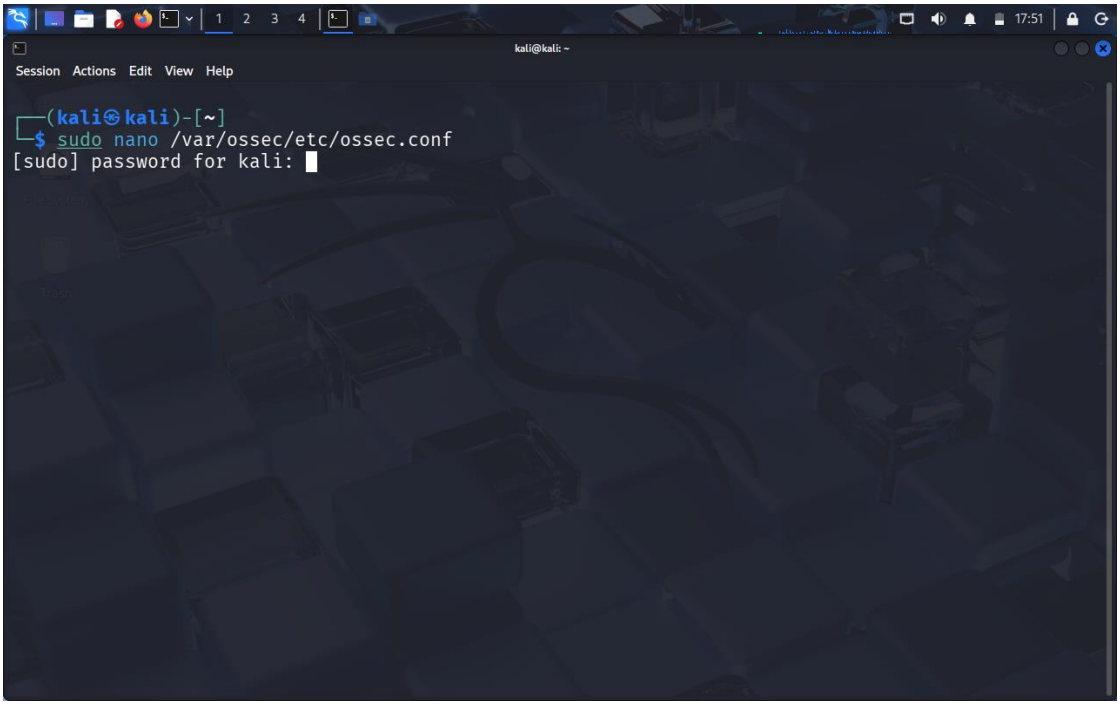

Fig: 8.3 Kali Linux terminal — opening /var/ossec/etc/ossec.conf for syscheck FIM configuration

Step 3: Configure syscheck Block on Kali Linux

The syscheck block was updated with the following changes to enable realtime monitoring on critical directories:

- /etc — realtime=yes, report_changes=yes (detect and show exact diff of file changes)
- /var/log — realtime=yes (monitor log directory for tampering or new log files)
- Scan frequency — changed from 43200 seconds (12 hours) to 600 seconds (10 minutes)

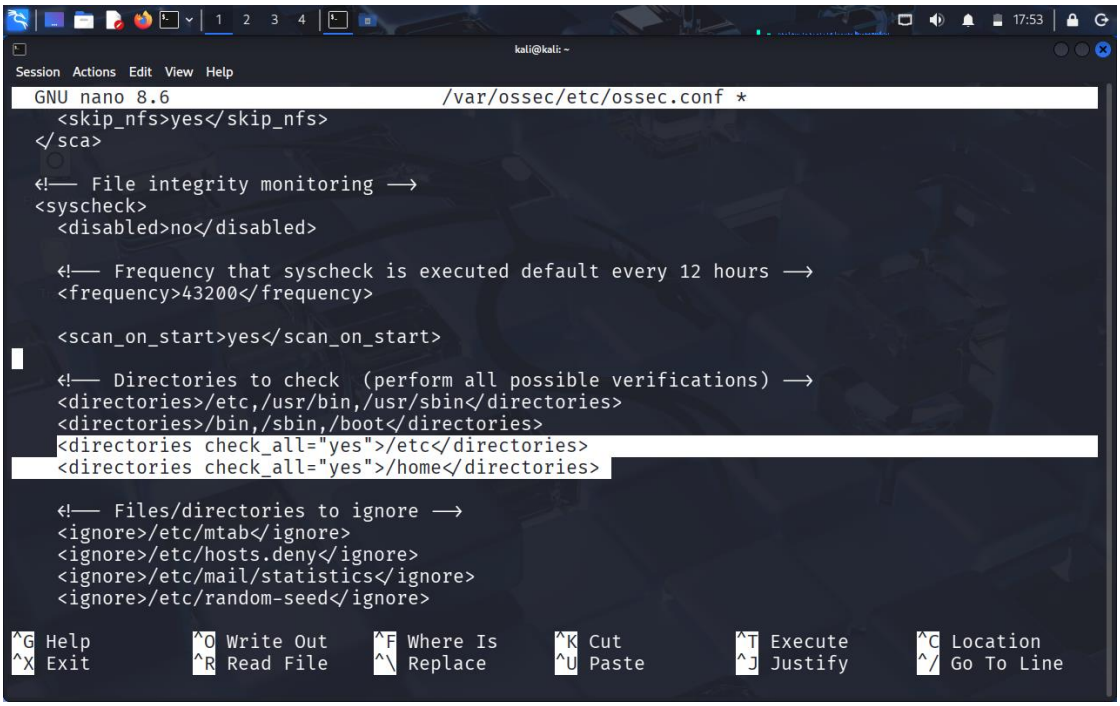

Fig: 8.4 Kali ossec.conf syscheck block — /etc and /home added with check_all=yes (initial view)

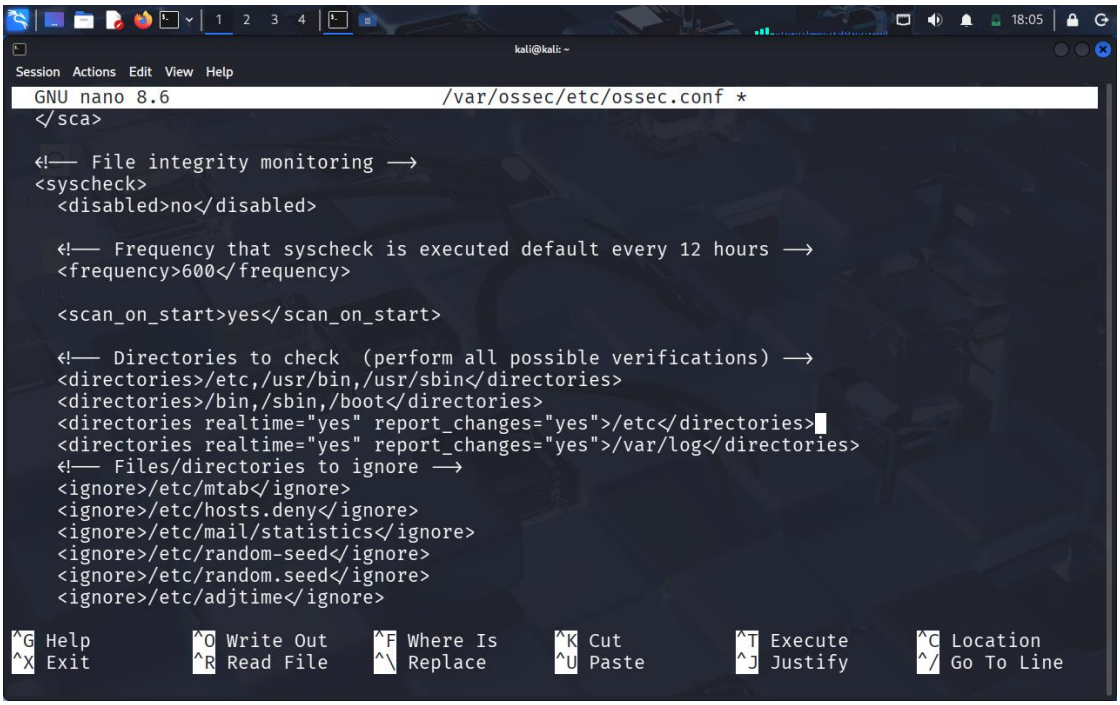

Fig: 8.5 Kali ossec.conf final syscheck config — realtime=yes and report_changes=yes on /etc and /var/log, frequency=600

Step 4: Restart Kali Agent After FIM Configuration

After saving the updated configuration, the Wazuh agent on Kali Linux was restarted to apply the new FIM settings. The agent status was verified after restart.

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

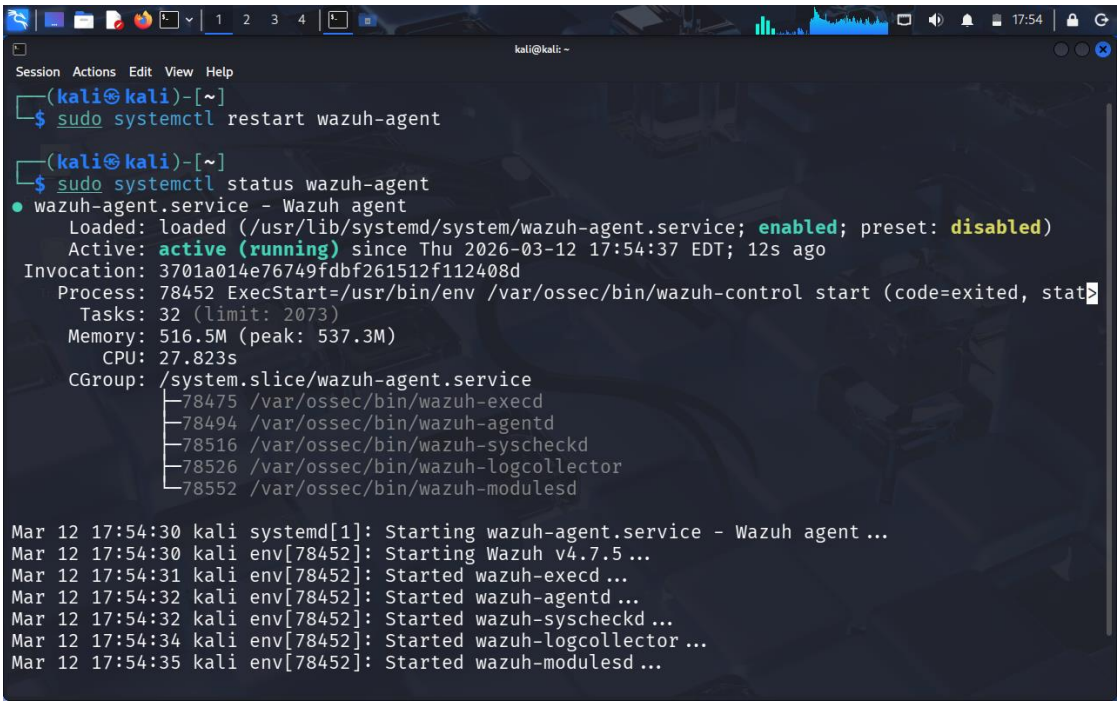

Fig: 8.6 Kali Wazuh agent restarted at 17:54 — active (running) with wazuh-syscheckd module running

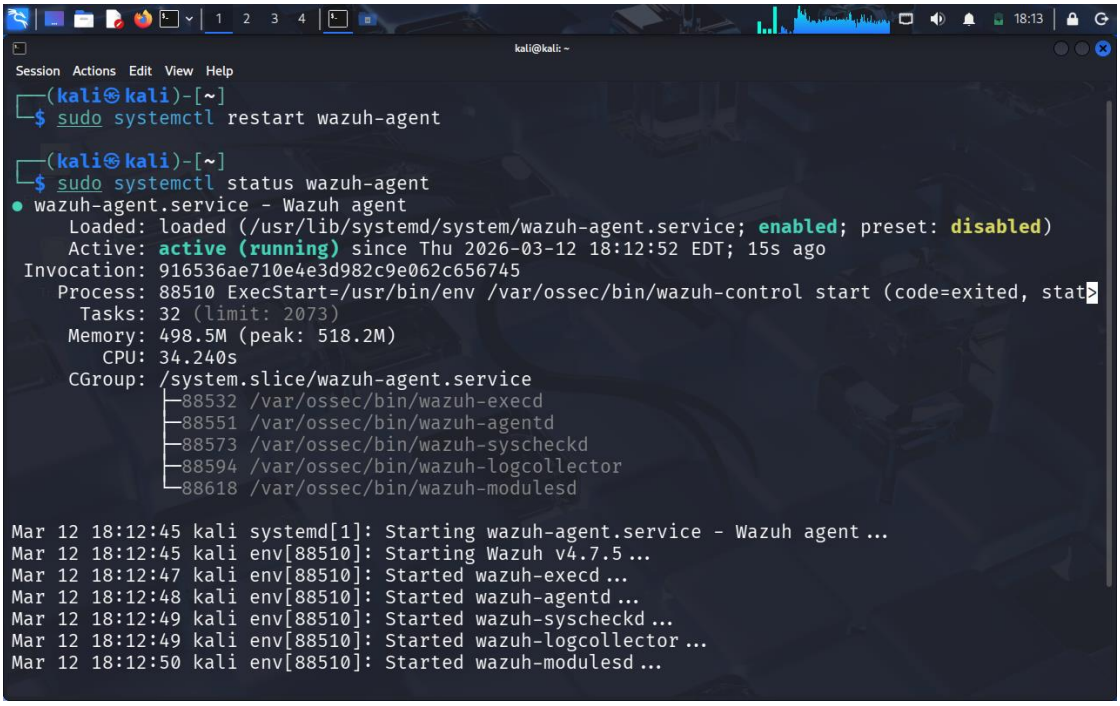

Fig: 8.7 Kali Wazuh agent restarted again at 18:12 after final config — all modules including syscheckd active

Step 5: Generate FIM Test Events on Kali Linux

Test events were generated in the monitored /etc directory to verify FIM alerts were triggered for all three event types: file creation, modification, and deletion.

```
sudo touch /etc/soc_test.txt
```

```
echo "Tested by Abdul Muqeet" | sudo tee /etc/soc_test.txt
sudo rm /etc/soc_test.txt
```

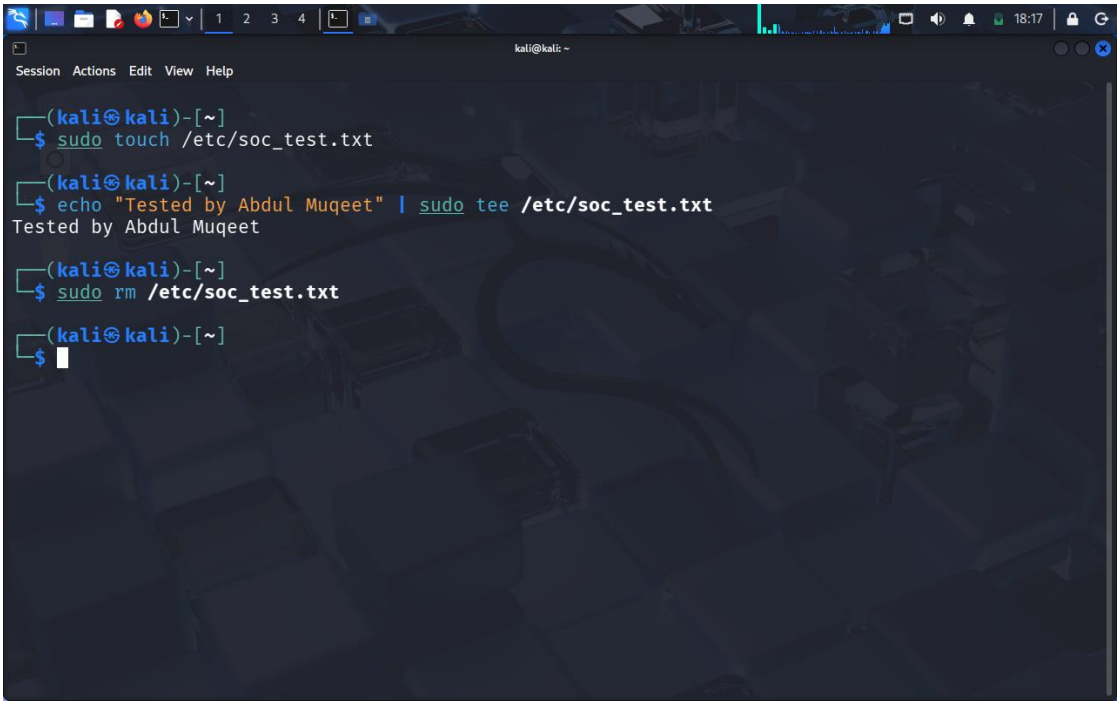

Fig: 8.8 Kali Linux — FIM test events: file created (touch), modified (echo+tee), and deleted (rm) in /etc

Step 6: Configure syscheck on Windows Agent

The Windows agent ossec.conf was also updated to include realtime FIM monitoring. C:\Users\Public was selected as the monitored path because C:\Windows\System32 restricts direct file creation during testing. The configuration was edited in Notepad.

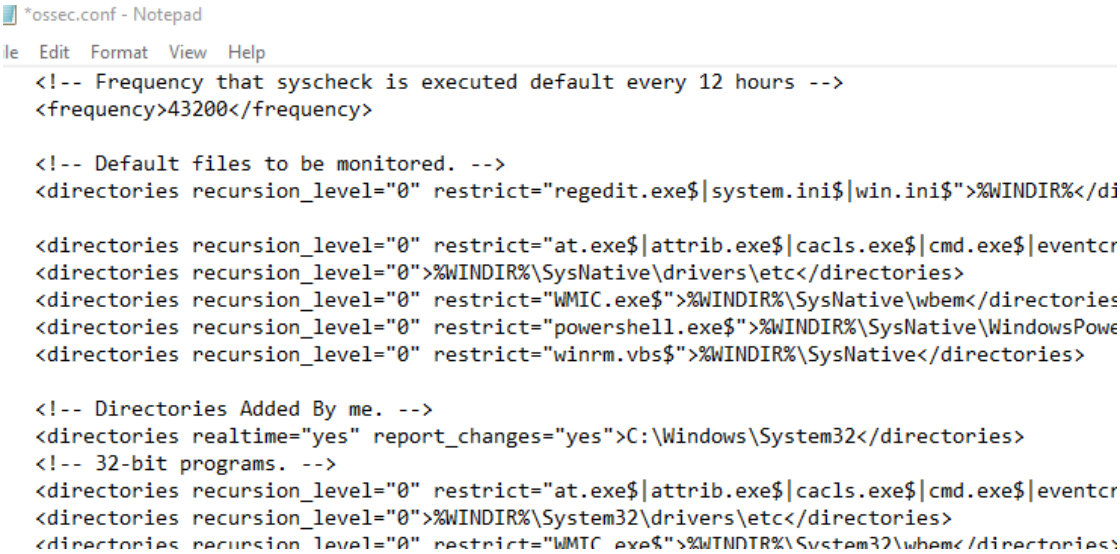

Fig: 8.9 Windows agent ossec.conf in Notepad — syscheck block showing FIM configuration with C:\Users\Public added

Step 7: Restart Windows Agent After FIM Configuration

The Windows Wazuh agent service was restarted via PowerShell with administrator privileges to apply the updated FIM configuration.

```
Restart-Service -Name "Wazuh"
```

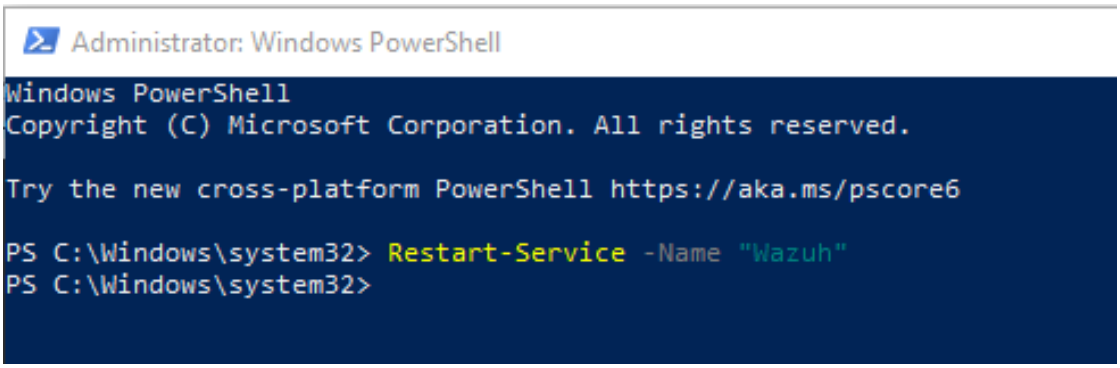

Fig: 8.10 Windows PowerShell — Restart-Service command executed, Wazuh service restarted successfully

Step 8: Verify Wazuh Manager Receiving FIM Alerts

The Wazuh Manager alerts log was monitored in real time on Ubuntu to confirm that FIM and authentication events from both agents were being received and processed.

```
sudo tail -f /var/ossec/logs/alerts/alerts.json
```

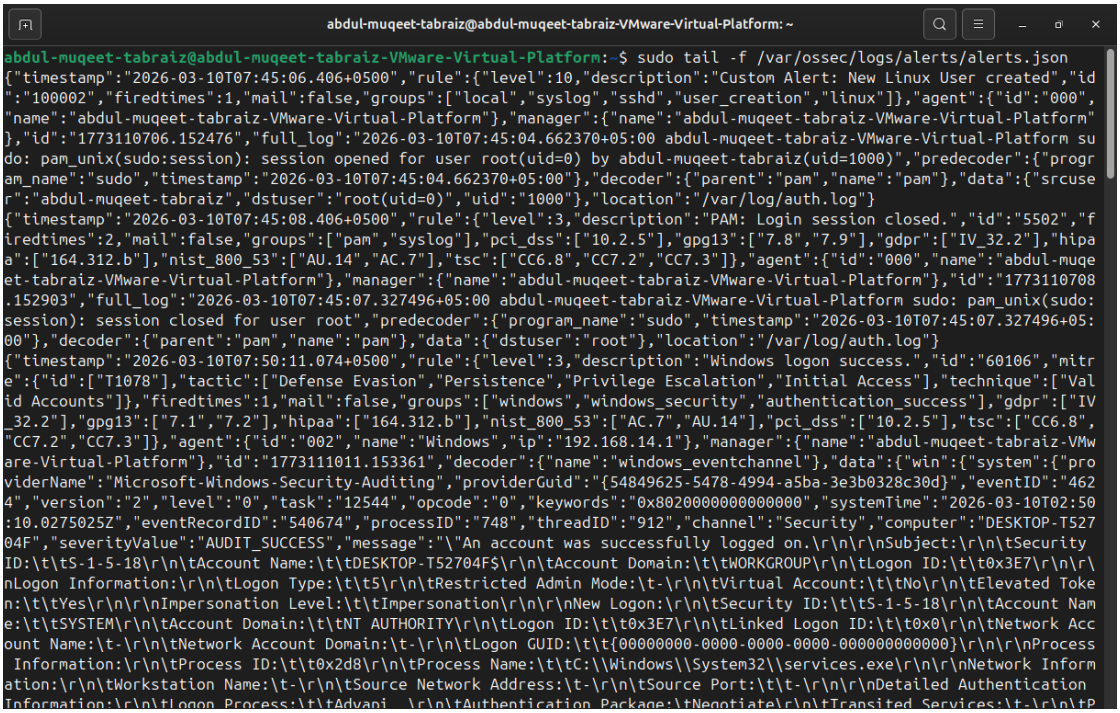

Fig: 8.11 Wazuh Manager alerts.json — FIM and authentication events received from both Kali and Windows agents

Result

File Integrity Monitoring was successfully configured on both Kali Linux and Windows agents. Realtime monitoring was enabled on /etc and /var/log on Kali with scan frequency reduced to 600 seconds. FIM test events (file creation, modification, and deletion) were generated and detected. All FIM alerts were visible in the Wazuh Manager alerts log and confirmed in the Wazuh Dashboard Integrity Monitoring module. The Windows agent was similarly configured and verified.

Summary of Days 8–9

Days 8 and 9 gave me hands-on experience with one of the most critical components of a SIEM deployment — File Integrity Monitoring. In a real SOC environment, FIM is the first line of defence against attackers who modify system files, plant backdoors, or tamper with log files to cover their tracks.

The key insight was understanding the difference between scheduled scans (syscheck_frequency) and realtime monitoring. Realtime monitoring uses inotify on Linux and the Windows kernel change notification API, generating alerts the moment a file changes rather than waiting for the next scan cycle. For critical directories like /etc, realtime is the only acceptable choice.

As a SOC analyst, knowing which directories to monitor is as important as knowing how to monitor them. Monitoring everything produces noise, monitoring nothing leaves you blind. The right selection — /etc for configuration changes, /var/log for log tampering — reflects real threat intelligence thinking.

Cyberster Blue Team Internship — Week 2 Days 10–11

Custom Wazuh Rules

Objective

The objective of Days 10 and 11 was to write and deploy three custom detection rules in Wazuh to alert on specific high-priority security events: new Linux user account creation, SSH brute force attacks, and USB device insertion on Windows. Each rule was mapped to a MITRE ATT&CK technique and confirmed by triggering real test events on the agent machines.

Tools Used

- Wazuh Manager (Ubuntu) — custom rules file at /var/ossec/etc/rules/local_rules.xml
- wazuh-analysisd -t — rule syntax validation tool (run before every manager restart)
- Kali Linux Agent — used to trigger user account creation and SSH brute force events
- Windows Agent — used to trigger USB device insertion via Event ID 6416
- Wazuh Dashboard — Security Events module for verifying rules fire correctly

Procedure

Step 1: Open the Custom Rules File on Ubuntu Manager

All custom rules are authored in the local_rules.xml file on the Wazuh Manager (Ubuntu). This file is loaded by wazuh-analysisd on startup and extends the default Wazuh ruleset without modifying core rule files.

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

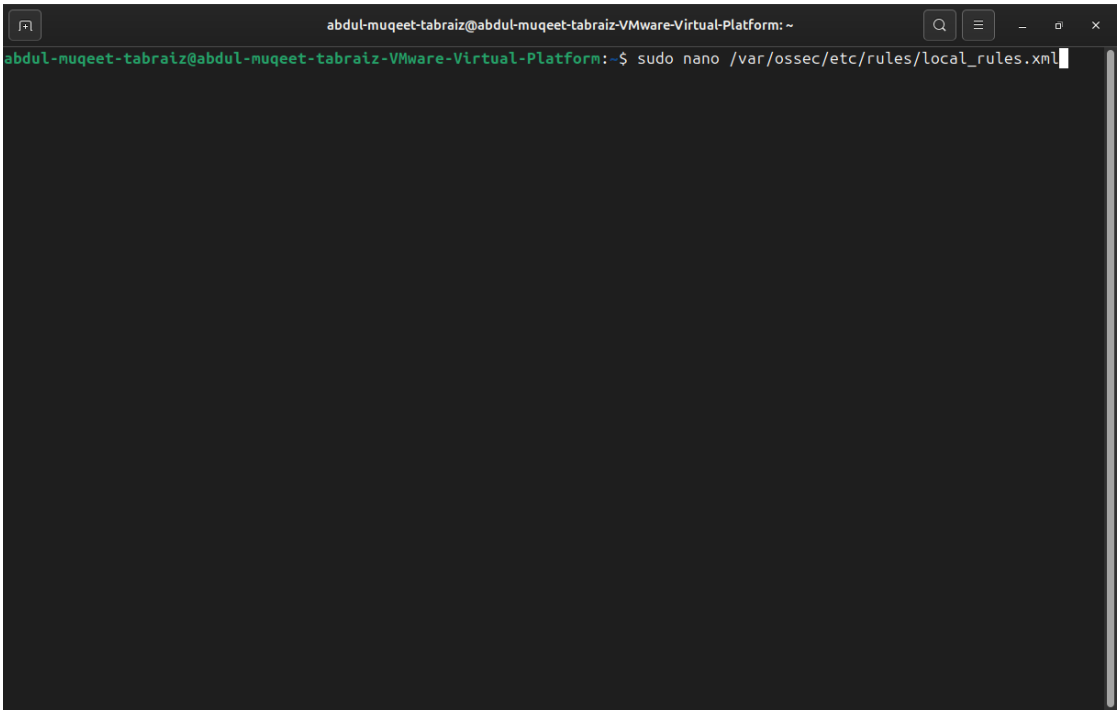

Fig: 10.1 Ubuntu Manager — opening local_rules.xml for custom rule authoring

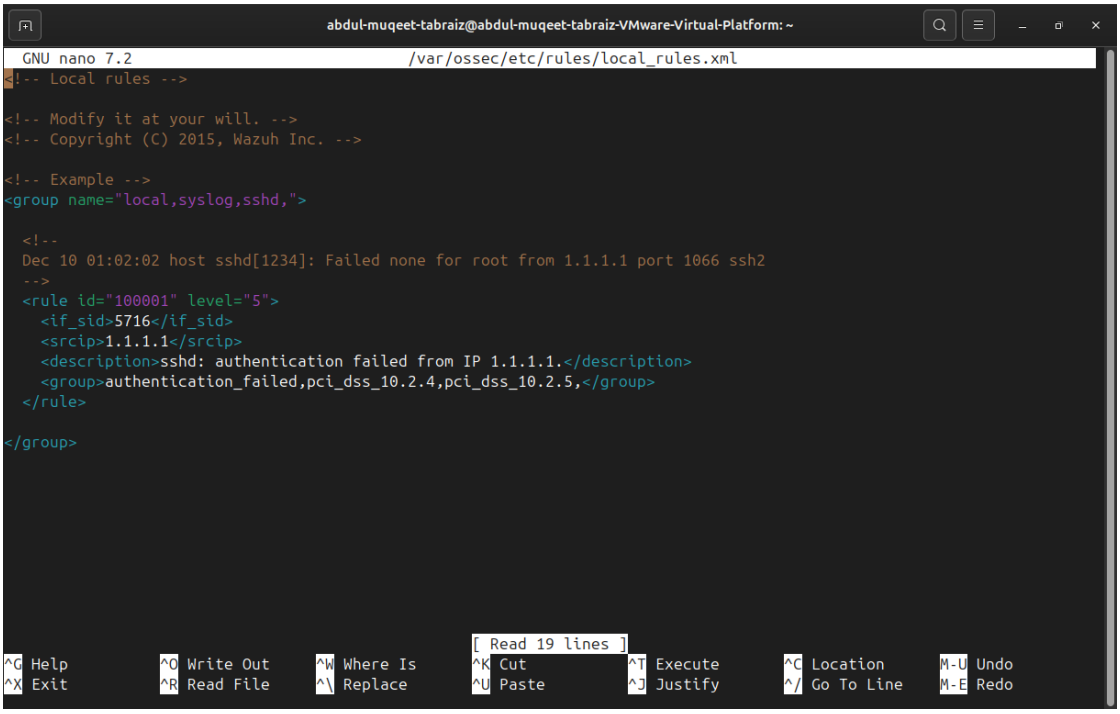

Fig: 10.2 local_rules.xml initial state — default example rule shown before custom rules added

Step 2: Write Rule 100100 — New Linux User Account Created

Rule 100100 was written to fire whenever a new Linux user account is created on an agent. It extends parent rule **5501** (PAM useradd) and is tagged with MITRE ATT&CK technique **T1136 — Create Account.**

```
<rule id="100100" level="10">
```

```
  <if_sid>5501</if_sid>
  <description>Custom Alert: New Linux user account created</description>
  <group>user_creation,linux,</group>
</rule>
```

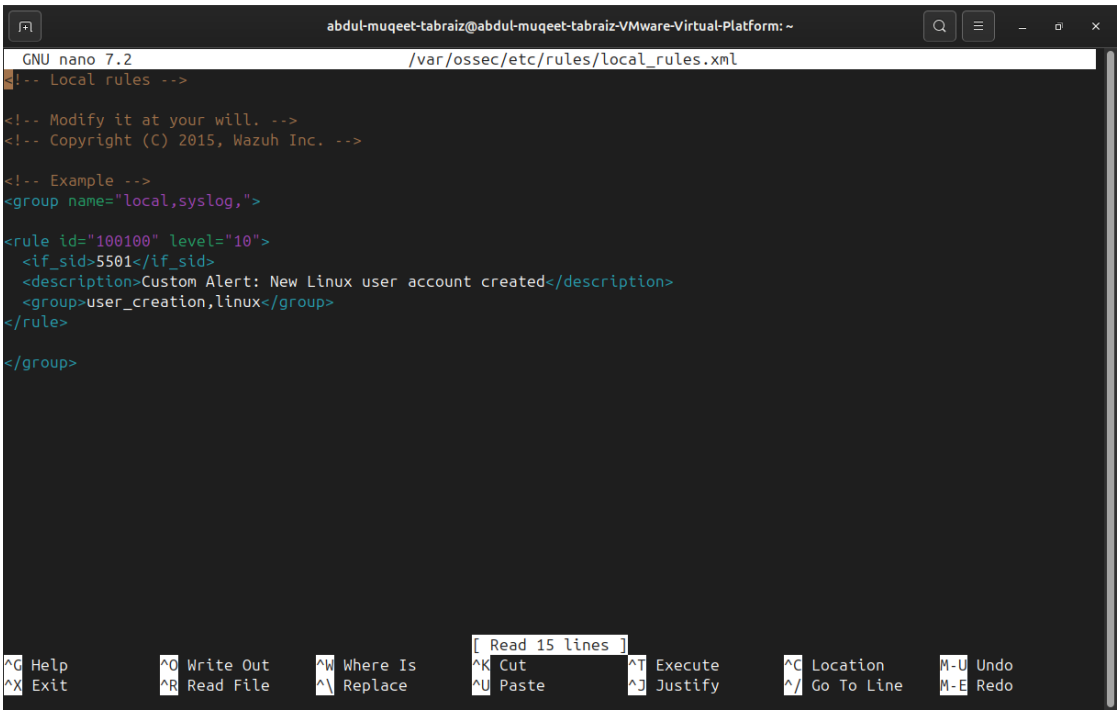

Fig: 10.3 local_rules.xml — Rule 100100 (user creation) written and saved to file

Step 3: Write Rule 100200 — SSH Brute Force Attack Detected

Rule 100200 detects SSH brute force by tracking repeated failures from the same source. It uses **if_matched_sid: 5710** (SSH auth failure) with frequency=6 over a 60-second timeframe. MITRE technique: **T1110 — Brute Force.**

```
<rule id="100200" level="12" frequency="6" timeframe="60">
  <if_matched_sid>5710</if_matched_sid>
  <description>Custom Alert: SSH Brute Force Attack Detected</description>
  <group>authentication_failures,bruteforce,</group>
</rule>
```

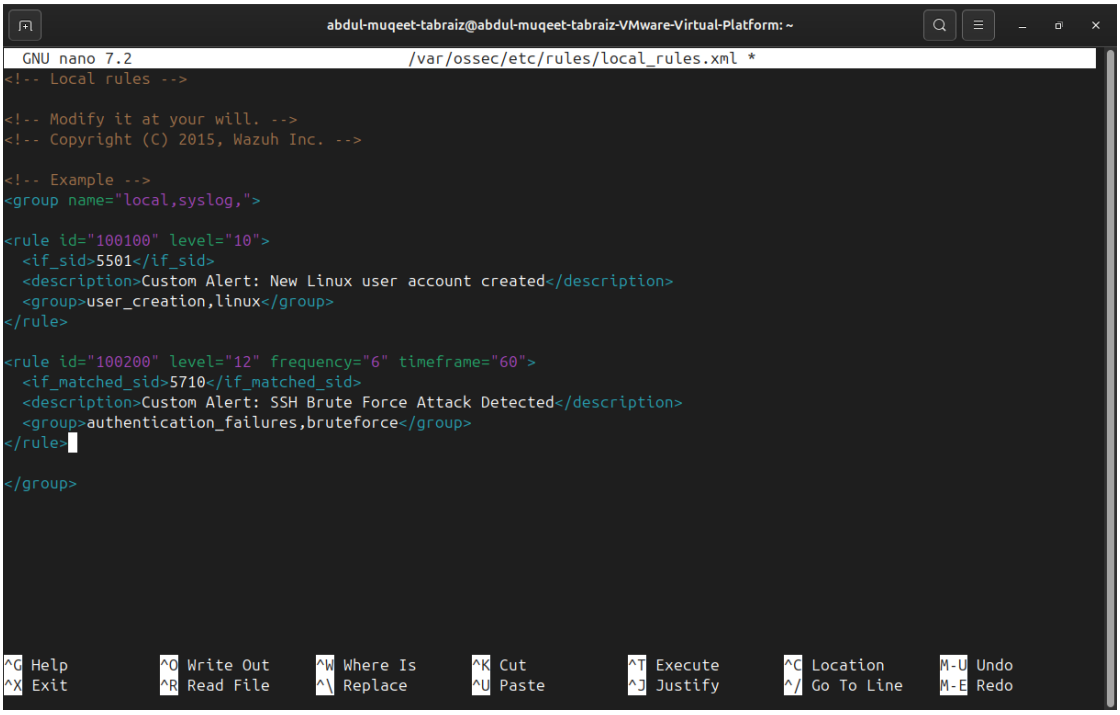

Fig: 10.4 local_rules.xml — Rule 100200 (SSH brute force) added below Rule 100100 in same group block

Step 4: Write Rule 100003 — USB Device Insertion on Windows

Rule 100003 detects USB storage device insertion on Windows via Windows Security Event ID 6416 (New external device recognized). MITRE technique: T1091.

An initial attempt used Event ID 2003 from the DriverFrameworks-UserMode channel (shown in Fig 10.5) but that channel is not monitored by default. The final working version uses Event ID 6416 from the Security log after enabling Plug and Play audit policy on Windows.

```
# Enable on Windows first:
auditpol /set /subcategory:"Plug and Play Events" /success:enable

<rule id="100003" level="8">
  <if_group>windows</if_group>
  <field name="win.system.eventID">^6416$</field>
  <description>SOC Alert: New external USB device recognized on Windows
endpoint</description>
  <group>usb_insertion,removable_media,</group>
  <mitre><id>T1091</id></mitre>
</rule>
```

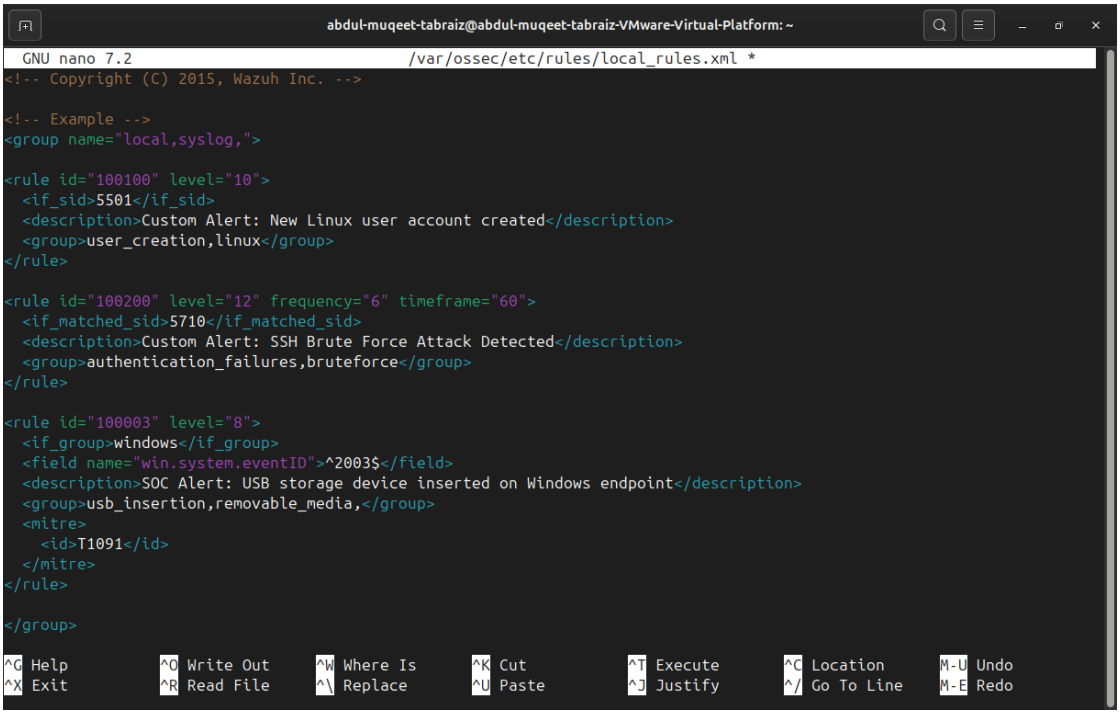

Fig: 10.5 local_rules.xml — first USB rule version using Event ID 2003 (later updated to 6416)

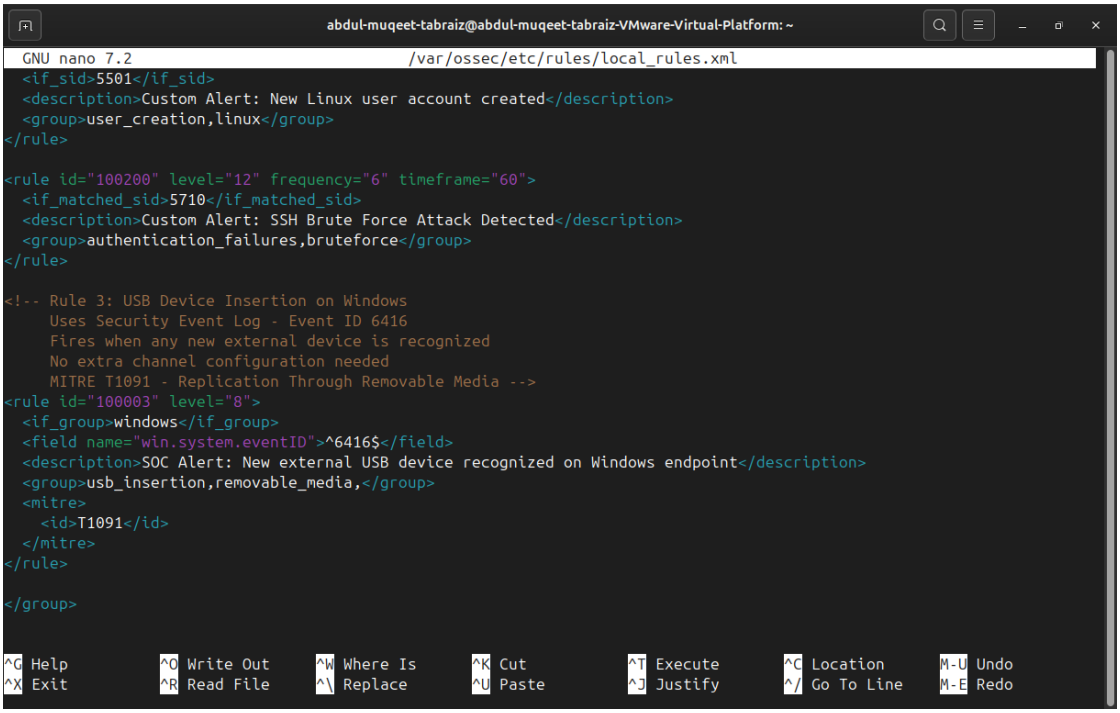

Fig: 10.6 local_rules.xml — final version with all 3 rules; USB rule using Event ID 6416 with full MITRE mapping

Step 5: Validate Rule Syntax Before Restarting

Before restarting the Wazuh Manager, the rule file was validated using the wazuh-analysisd test flag. An early version of the file had a typo ('timefrane' instead of 'timeframe') that caused the Manager to fail to start — this was caught by the validator and corrected.

```
sudo /var/ossec/bin/wazuh-analysisd -t
```

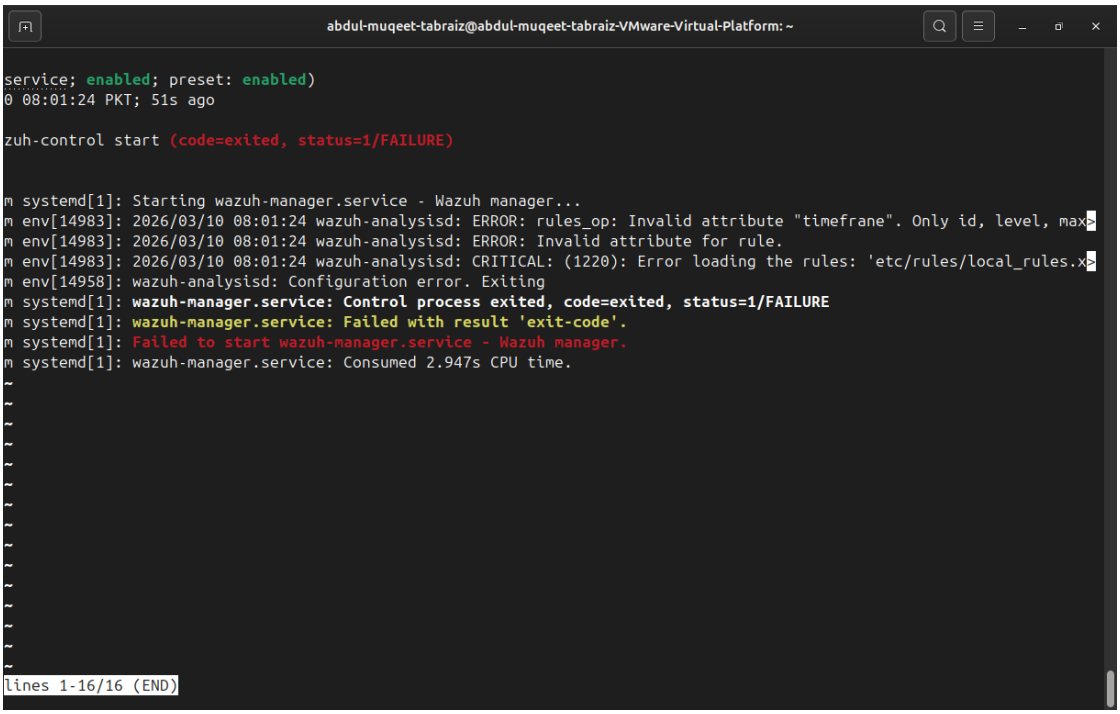

Fig: 10.7 Wazuh Manager startup FAILURE — 'timefrane' typo in rule 100003 caught by analysisd error log

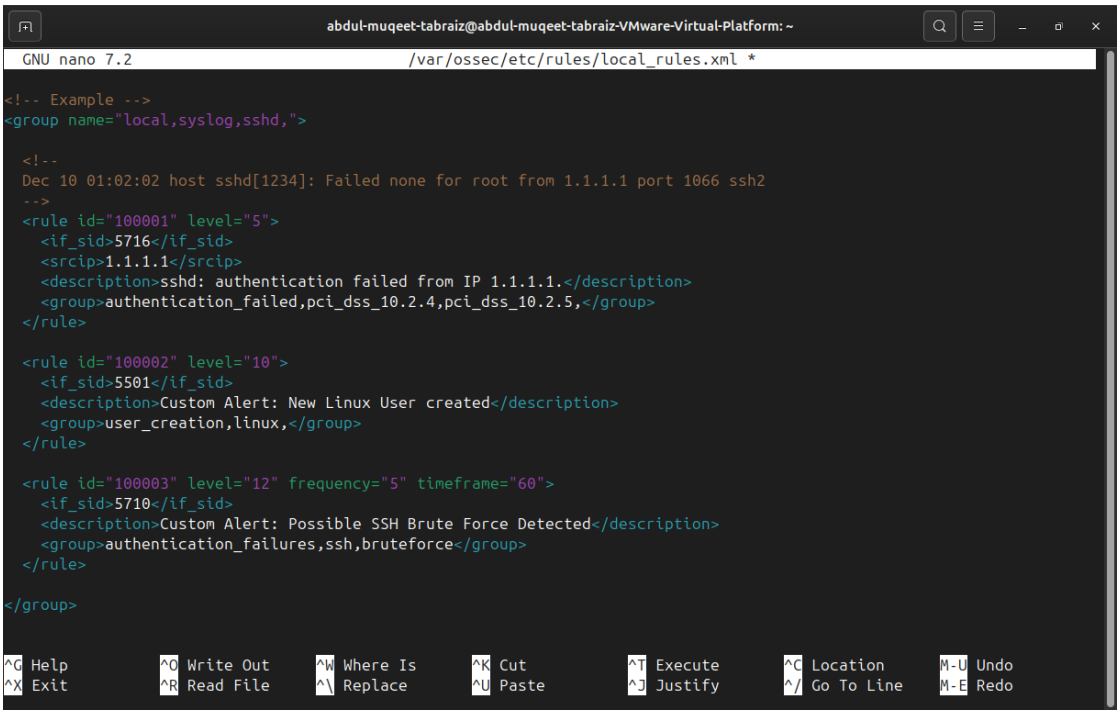

Fig: 10.8 local_rules.xml — typo corrected, 'timeframe' attribute properly spelled in Rule 100200

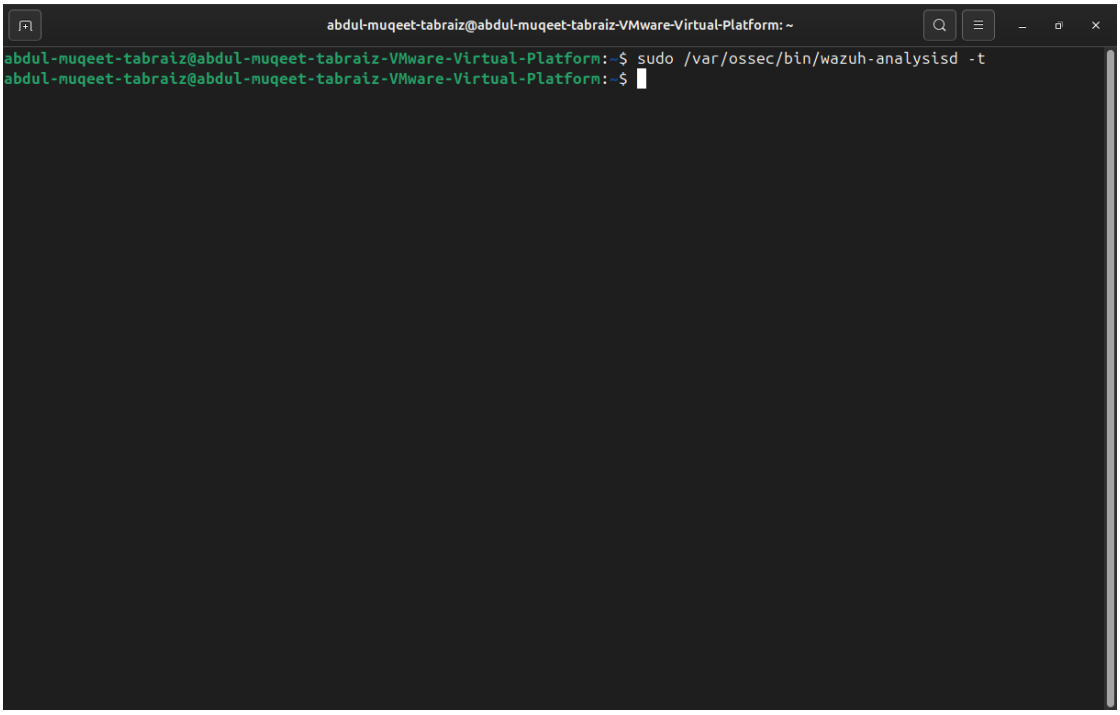

Fig: 10.9 wazuh-analysisd -t exits cleanly with no errors — all rules syntax valid and ready to load

Step 6: Restart Wazuh Manager to Load Custom Rules

After validation passed, the Wazuh Manager was restarted to load the three custom rules into the detection engine.

```
sudo systemctl restart wazuh-manager
sudo systemctl status wazuh-manager
```

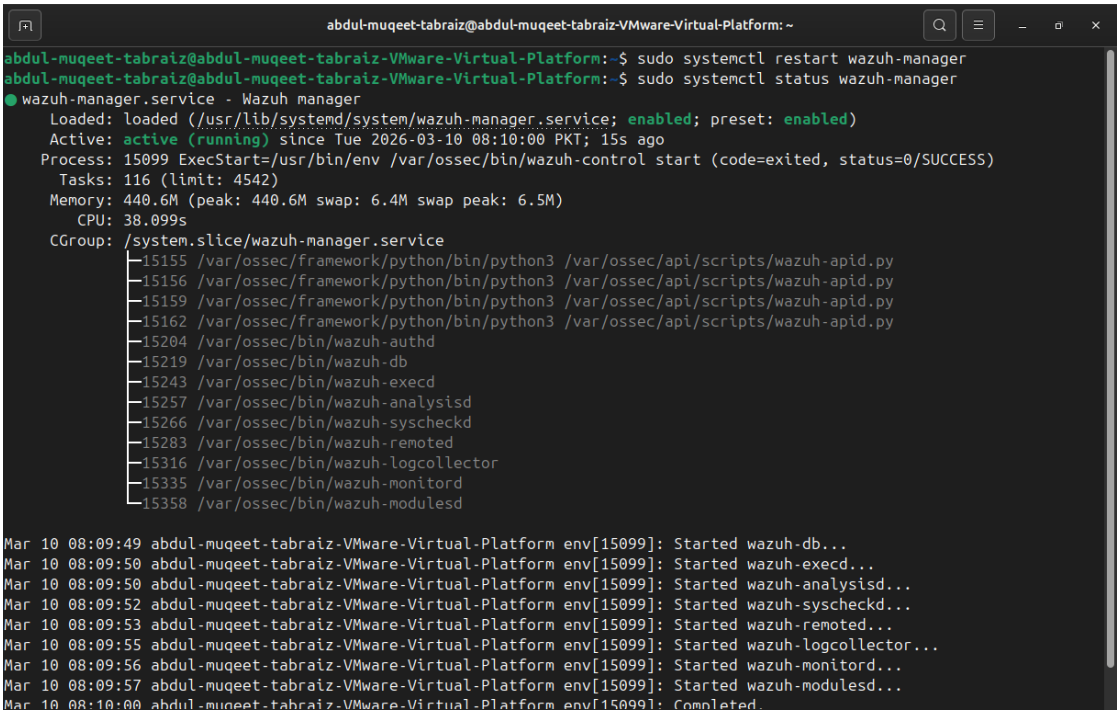

Fig: 10.10 Wazuh Manager restarted and active — custom rules loaded into wazuh-analysisd (6776 total rules enabled)

Step 7: Test Rule 100100 — Trigger User Creation on Kali

A new user 'soc' was created on the Kali Linux agent using adduser to trigger Rule 100100 and verify it appears in the Wazuh Dashboard.

```
sudo adduser soc
```

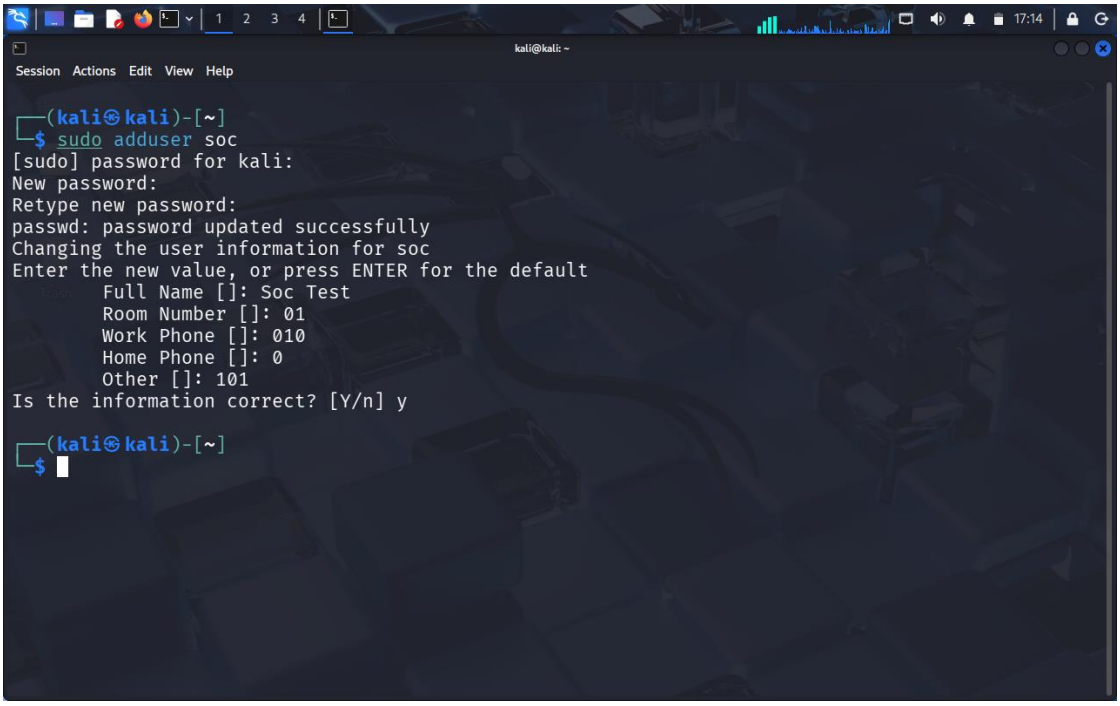

Fig: 10.11 Kali Linux — sudo adduser soc executed to trigger Rule 100100 user creation alert

Step 8: Test Rule 100200 — SSH Brute Force from Kali

Multiple failed SSH login attempts were made from Kali Linux targeting Ubuntu to trigger the brute force correlation rule.

```
ssh abdul-muqeet-tabraiz@192.168.14.140   # repeated with wrong passwords
```

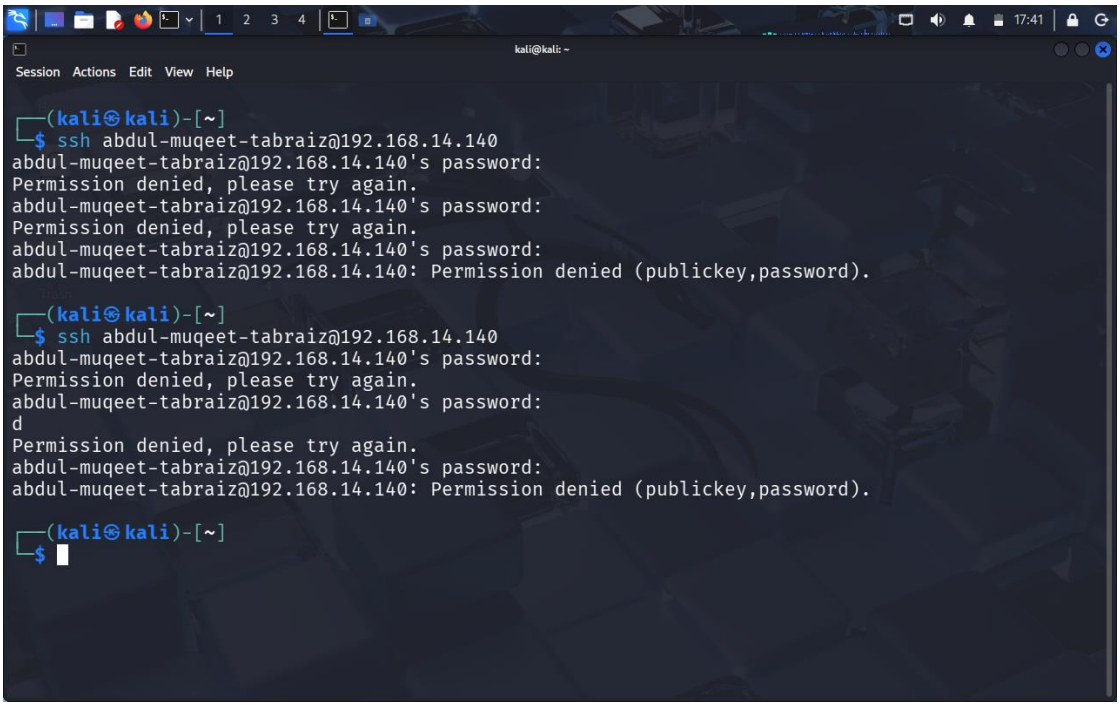

Fig: 10.12 Kali Linux — repeated SSH login failures to Ubuntu Manager triggering Rule 100200 brute force detection

Step 9: Verify All Rules Fired in Wazuh Dashboard

The Wazuh Dashboard Security Events module was used to confirm all three custom rules fired correctly and were attributed to the correct agents.

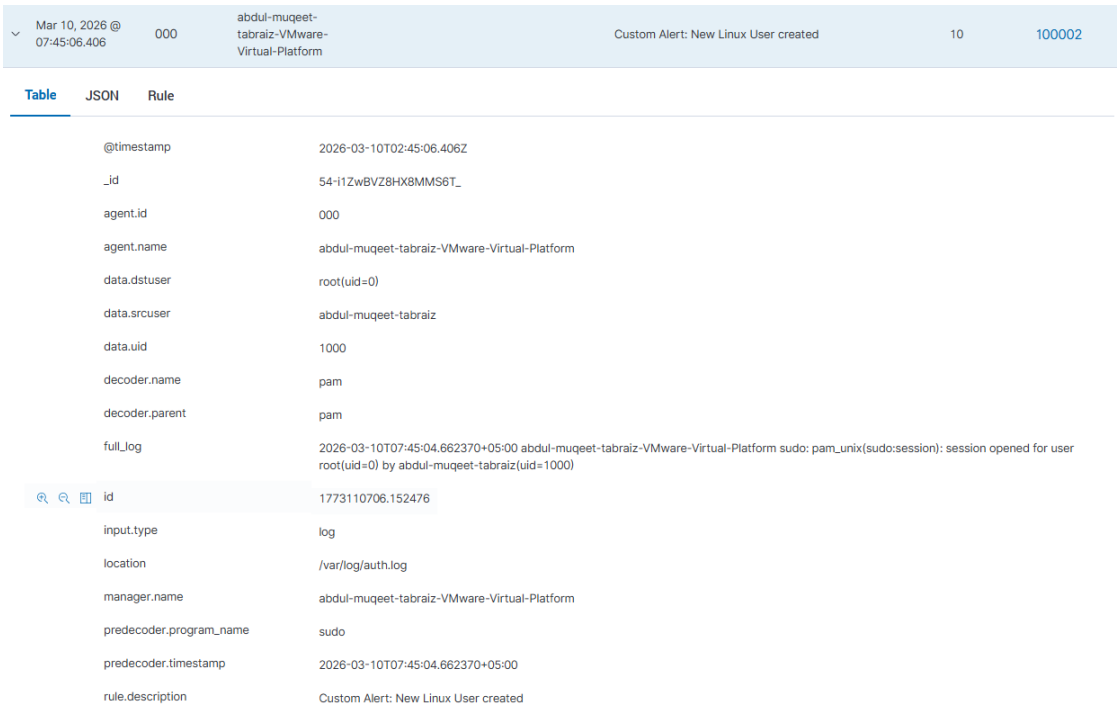

Fig: 10.13 Wazuh Dashboard alert detail — Custom Alert: New Linux User created, Rule 100002, Level 10, agent 000

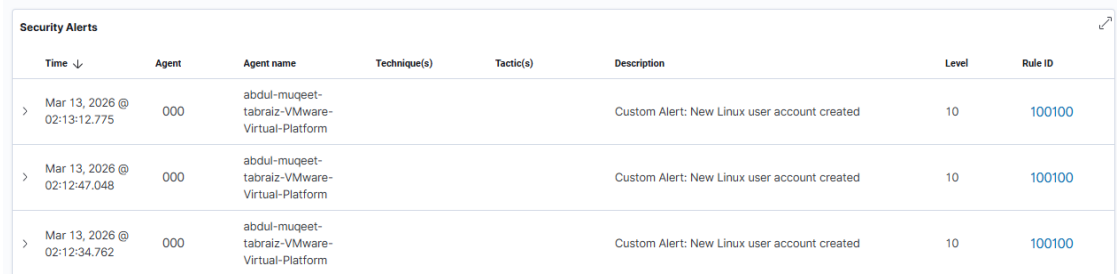

Fig: 10.14 Wazuh Dashboard Security Alerts — multiple Rule 100100 alerts visible across the event timeline

Result

All three custom detection rules were successfully authored, validated, deployed, and confirmed firing in the Wazuh Dashboard. Rule 100100 detected new Linux user account creation. Rule 100200 detected SSH brute force patterns via frequency correlation. Rule 100003 was configured and ready to detect Windows USB insertion via Event ID 6416. Each rule was mapped to its MITRE ATT&CK technique.

Summary of Days 10–11

Writing custom Wazuh rules is where a SOC analyst transitions from using a tool to truly understanding it. The default ruleset covers generic events, but custom rules allow the team to detect the specific threats relevant to their environment and threat model.

The most valuable lesson was the mandatory use of wazuh-analysisd -t before every manager restart. A single typo ('timefrane') caused the entire Wazuh Manager to fail to start in a production environment this would mean zero detection capability. Rule validation must be part of every deployment workflow.

The USB rule also taught me the importance of log source verification. Not every Windows event channel is monitored by default. As a detection engineer, understanding what data is actually being collected before writing rules that depend on it is fundamental a rule that references a log source that isn't monitored will never fire.

Cyberster Blue Team Internship — Week 2 Days 12–13

Log Analysis & Threat Hunting

Objective

The objective of Days 12 and 13 was to perform comprehensive log analysis and threat hunting using the Wazuh Dashboard. This involved generating a range of suspicious events across both Linux and Windows agents, then using the dashboard's search and filtering capabilities to identify, correlate, and investigate events the way a SOC analyst would during an active incident or threat hunt.

Tools Used

- Wazuh Manager (Ubuntu) — aggregating logs from all agents into the indexer
- Wazuh Dashboard — Security Events, Threat Hunting, and MITRE ATT&CK modules
- Kali Linux Agent — source of SSH authentication failure and FIM events
- Windows Agent — source of Windows Security Event logs (4625, 4720, 4672)
- /var/log/auth.log — SSH authentication log monitored on Ubuntu Manager
- /var/ossec/logs/ossec.log — Wazuh Manager operational and FIM sync log

Procedure

Step 1: Confirm All Systems Ready for Log Analysis

The Wazuh Manager was verified running on March 13 and all agents confirmed active before beginning the log analysis session.

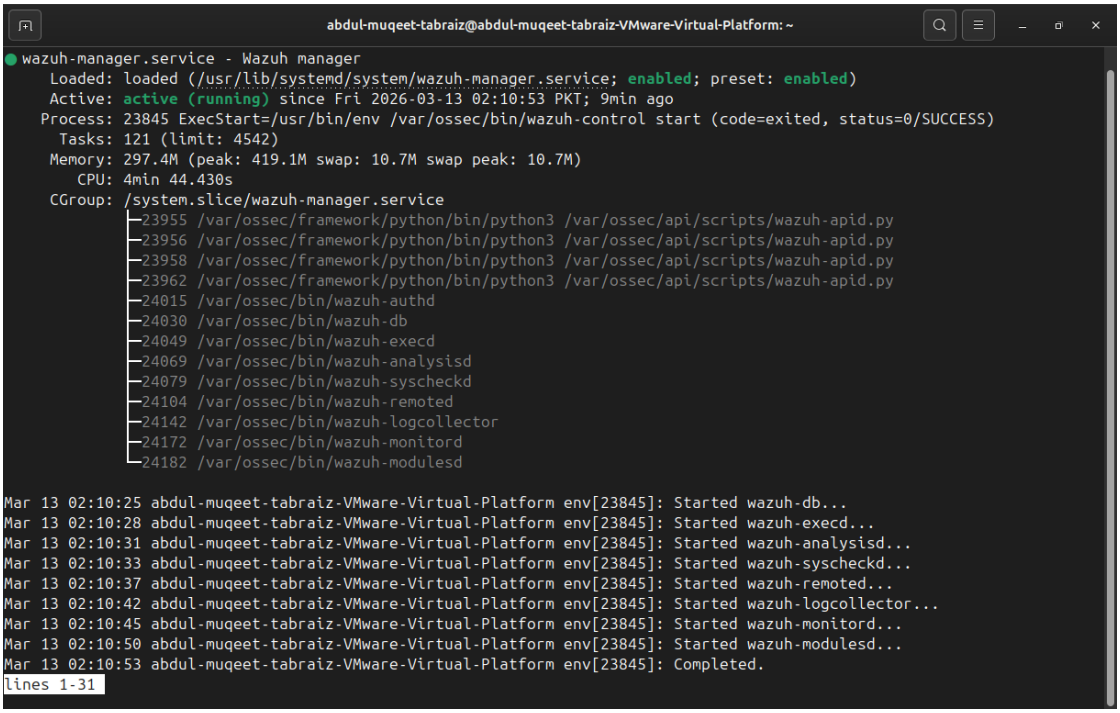

Fig: 12.1 Wazuh Manager running on Mar 13, 2026 at 02:10 — ready for log analysis

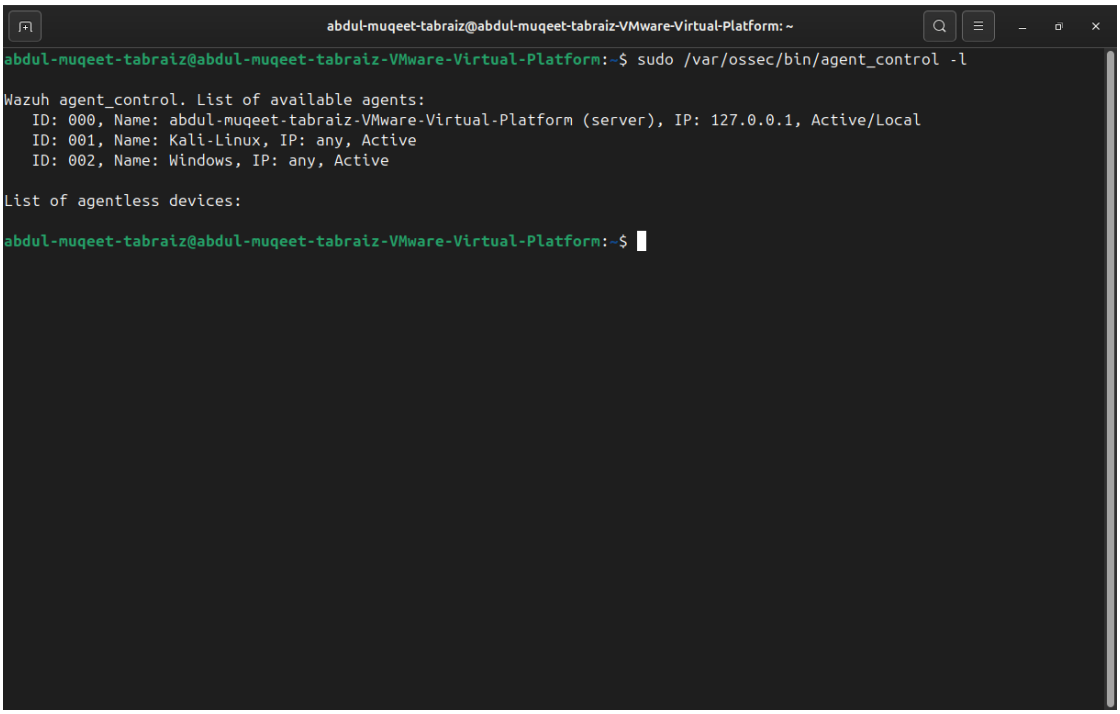

Fig: 12.2 agent_control -l — Kali-Linux (001) and Windows (002) both showing as Active

Step 2: Verify FIM Sync Active in Manager Log

The Wazuh Manager's ossec.log was checked to confirm the FIM syscheck module was actively running and syncing from both agents.

```
sudo tail -f /var/ossec/logs/ossec.log
```

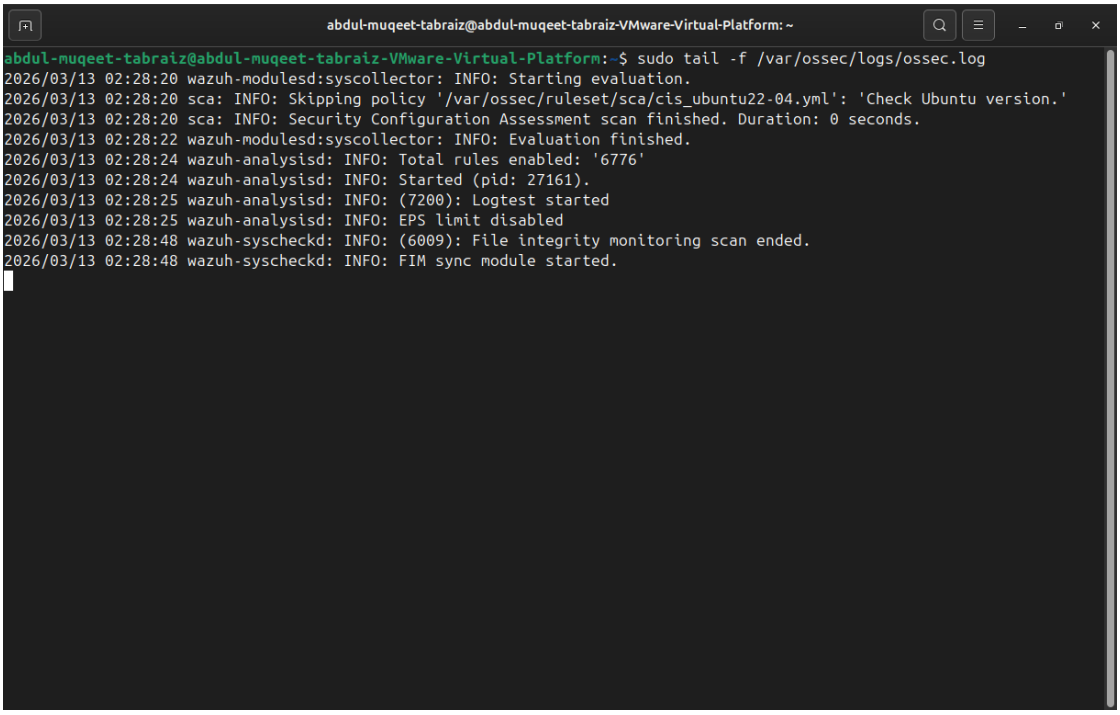

Fig: 12.3 ossec.log — FIM sync module started, syscheck scan complete (6009), 6776 rules enabled

Step 3: Generate SSH Brute Force Events from Kali

Failed SSH login attempts were generated from the Kali Linux agent to populate the authentication_failed and brute_force alert groups in the dashboard.

```
ssh wronguser@localhost   # repeated multiple times
```

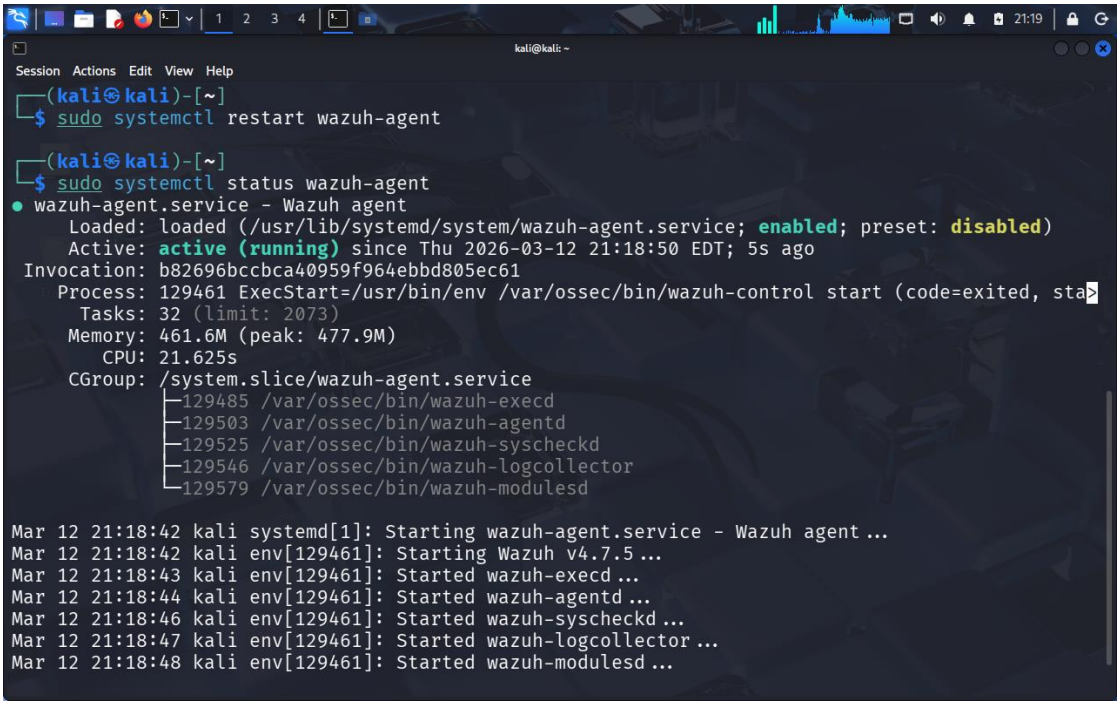

Fig: 12.4 Kali Linux — SSH failures to localhost generating authentication failure events for log analysis

Step 4: Monitor SSH Failures in Real Time on Ubuntu

The Ubuntu Manager's /var/log/auth.log was monitored in real time to verify SSH login failures from Kali were being recorded and would appear as Wazuh alerts.

```
sudo tail -f /var/log/auth.log
```

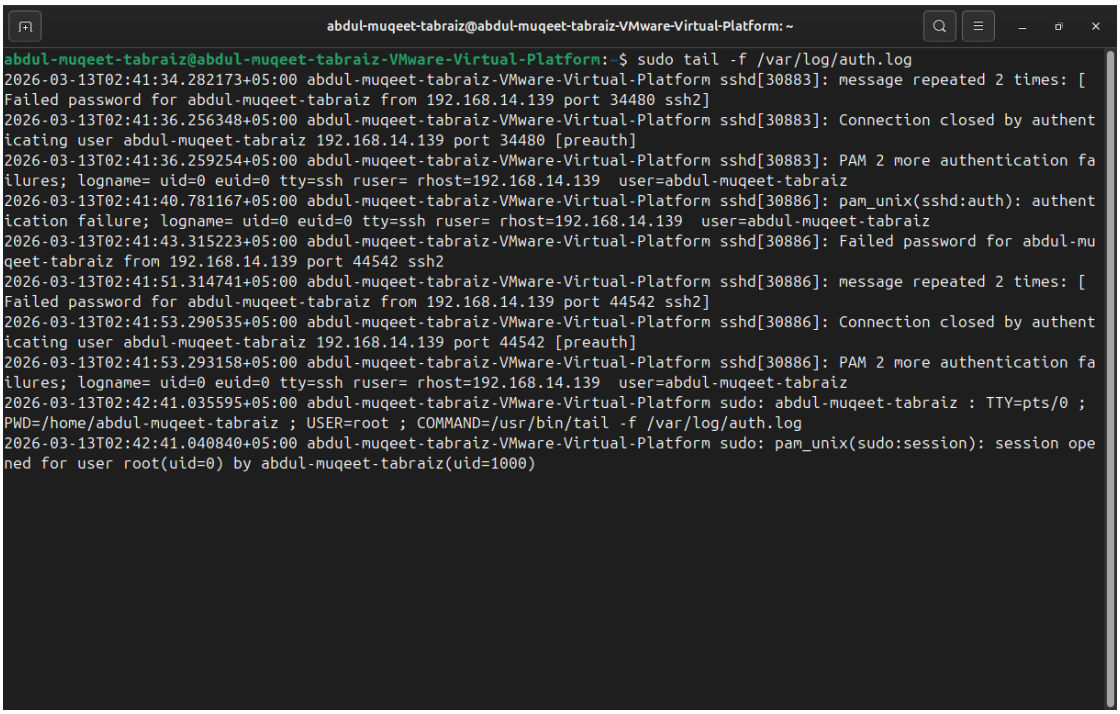

Fig: 12.5 Ubuntu auth.log — repeated SSH failed password entries from 192.168.14.139 (Kali) captured in real time

Step 5: Investigate Alert Details in Wazuh Dashboard

Each alert was expanded in the Wazuh Dashboard to review the complete event detail including rule ID, severity level, agent name, MITRE technique, rule groups, and the full raw log entry.

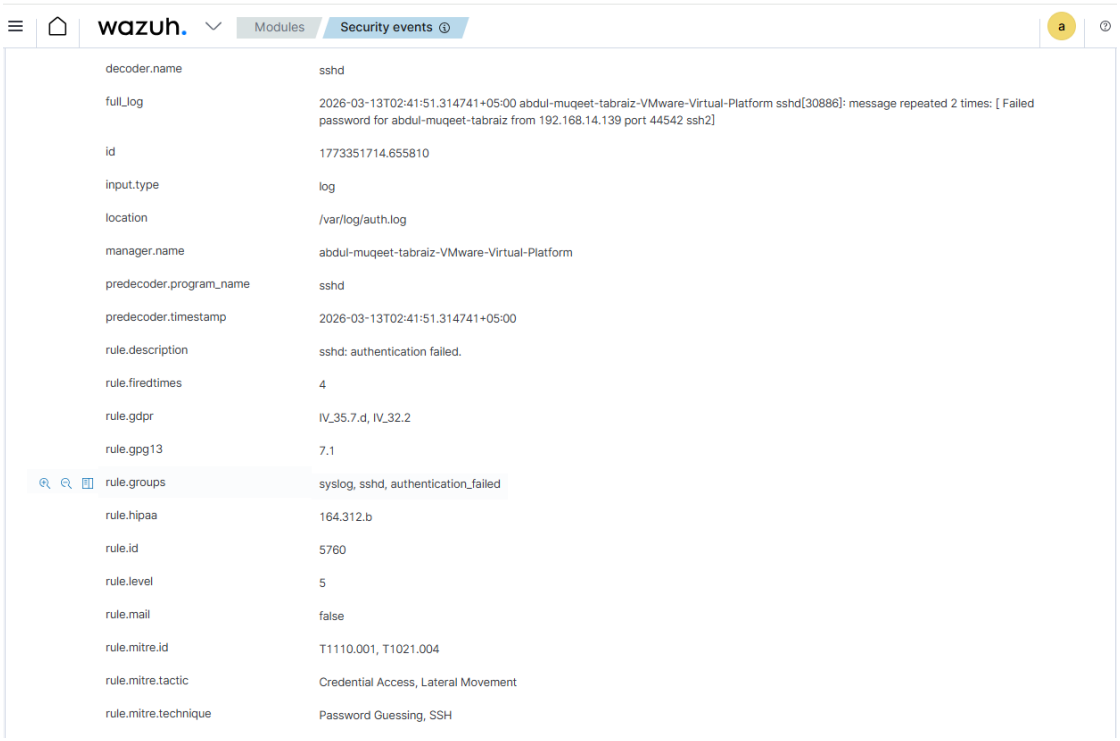

Fig: 12.6 Wazuh Dashboard event detail — rule.id 5760, groups: syslog/sshd/authentication_failed, full_log from /var/log/auth.log

Step 6: Security Alerts Dashboard Queries

The following dashboard queries were used during the threat hunting session to investigate specific event categories across all agents:

- rule.groups: authentication_failed — all failed authentication events
- rule.id: 100200 — custom SSH brute force correlation alerts
- rule.id: 100100 — custom new Linux user creation alerts
- rule.groups: syscheck — all FIM file integrity events
- win.system.eventID: 4625 — Windows failed logon events
- win.system.eventID: 4720 — Windows new user account created

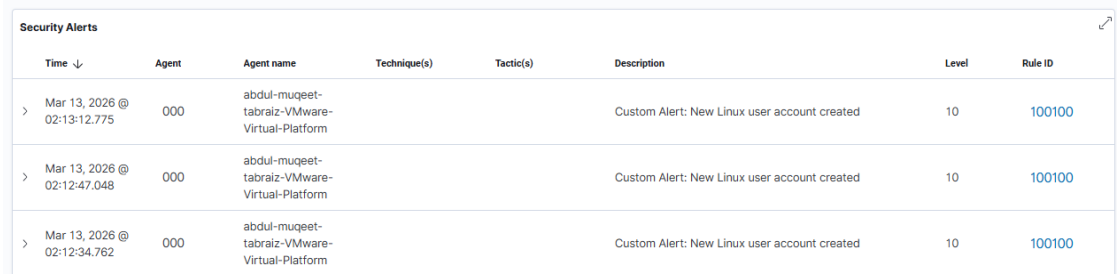

Fig: 12.7 Security Alerts dashboard — Rule 100100 multiple alerts across timestamp timeline for user creation events

Step 7: Review SSH Brute Force Alert Chain

The full chain of SSH brute force events was reviewed — from individual rule 5710 failures through to the Rule 100200 correlation alert firing after the frequency threshold was exceeded.

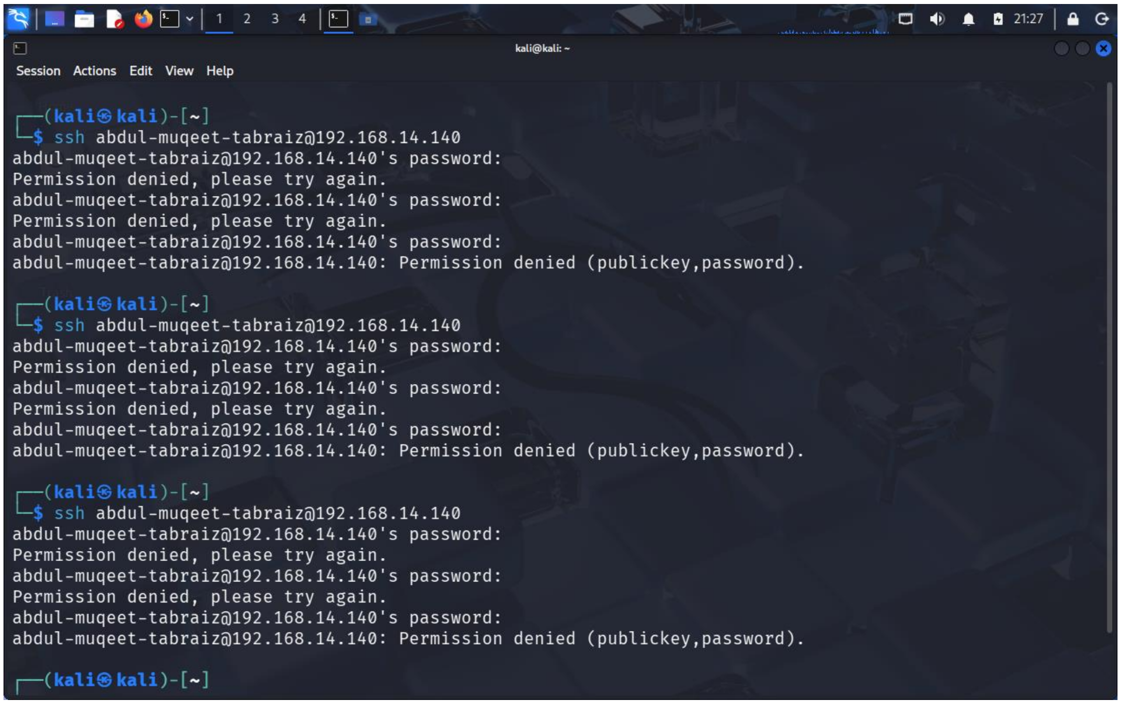

Fig: 12.8 Kali Linux — continued SSH brute force sequence generating repeated authentication failure events

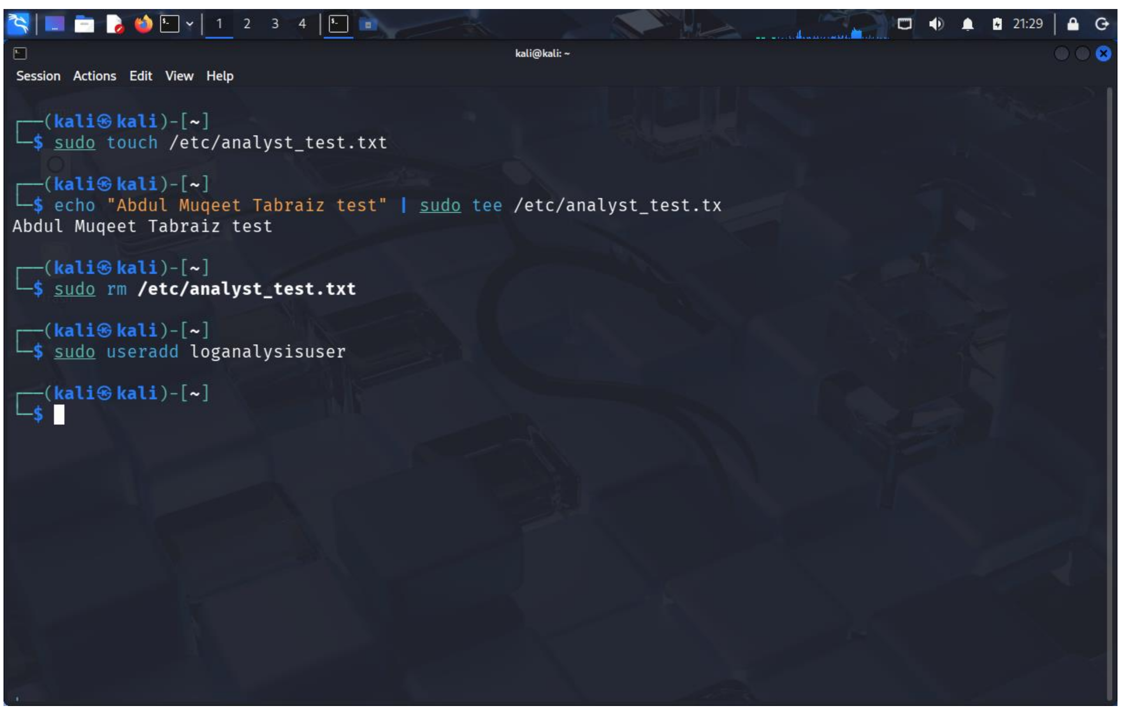

Fig: 12.9 Kali Linux — SSH brute force with Permission denied responses; events captured for dashboard review

Step 8: Additional Log Analysis Events Generated

Further events were generated across the session to build a comprehensive event timeline for analysis, including additional user creation events, FIM file changes, and Windows failed login attempts.

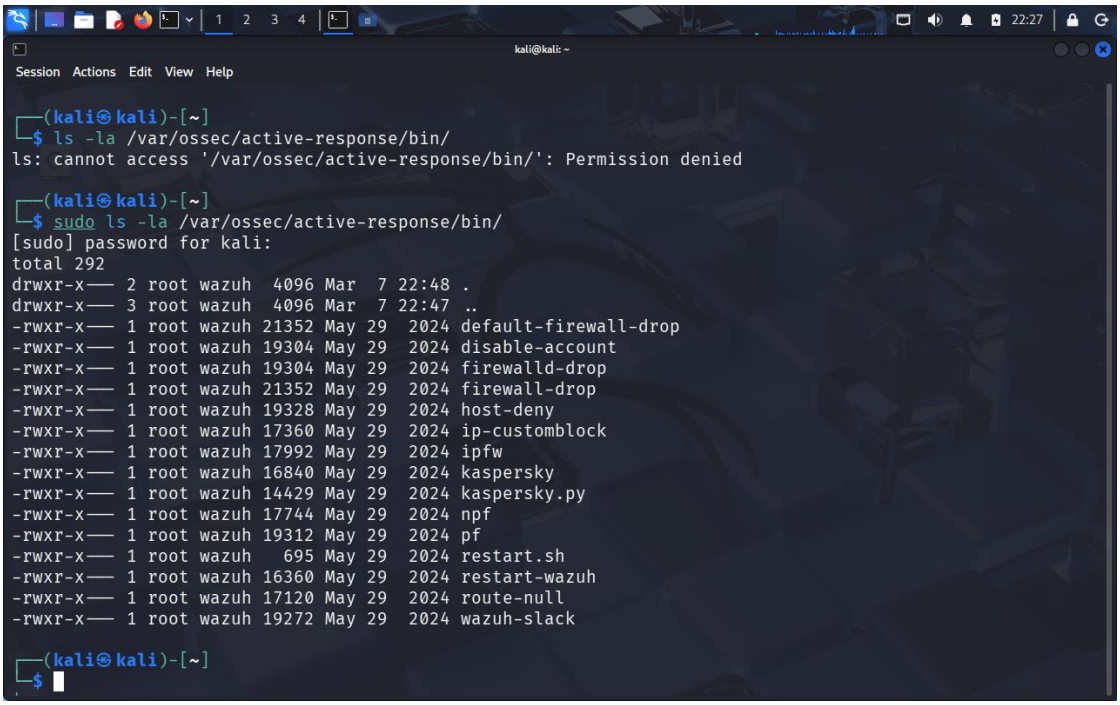

Fig: 12.10 Kali Linux — additional SSH brute force events for comprehensive log analysis investigation

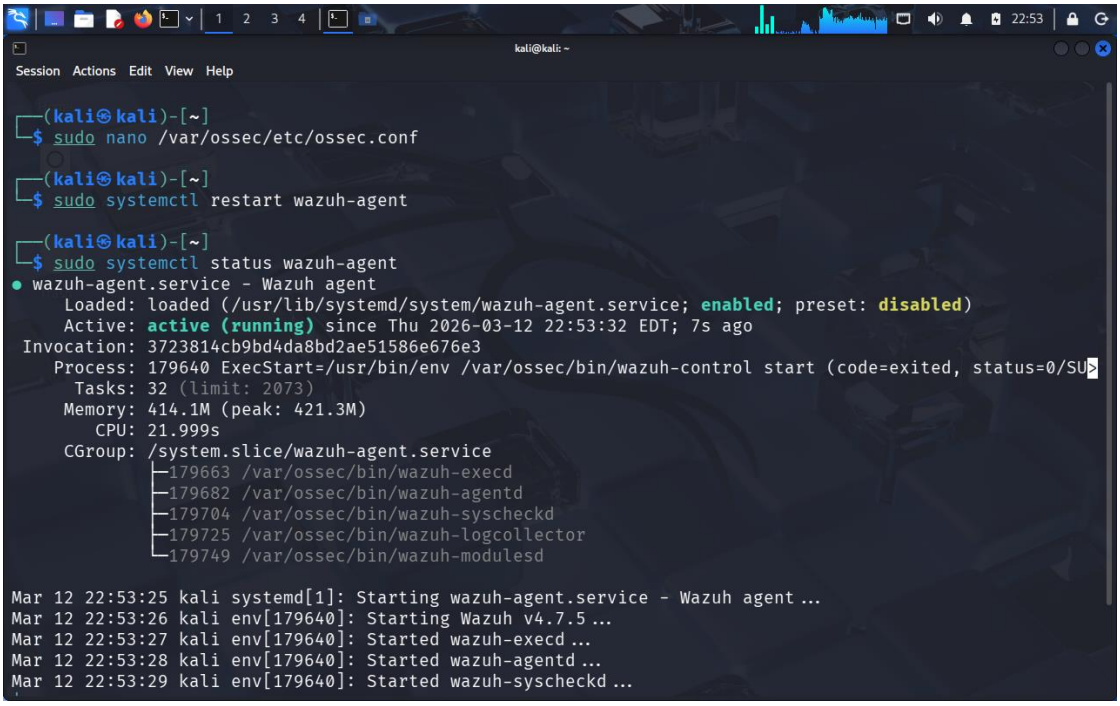

Fig: 12.11 Kali Linux — ongoing authentication failure events building the threat hunting event timeline

Result

A full log analysis and threat hunting session was successfully completed. Events spanning SSH authentication failures, user account creation, FIM changes, and Windows security events were

generated, captured, and investigated using the Wazuh Dashboard. All events were correctly attributed to their source agents with appropriate rule IDs, MITRE ATT&CK mappings, and severity levels.

Summary of Days 12–13

Log analysis is the core daily activity of every SOC analyst. These two days reinforced a critical mindset shift: a raw log is just data. An alert is a hypothesis. Only by examining the full event chain sequence, timing, agent attribution, and rule correlation can you determine whether a real threat exists.

Using the dashboard's query filters made it clear that knowing exactly which fields Wazuh indexes is essential for effective threat hunting. Queries like rule.groups: authentication_failed are far more efficient than scrolling through thousands of raw events. This is the difference between a reactive analyst and a proactive threat hunter.

The most valuable outcome was seeing the full kill chain from the first failed SSH attempt in auth.log through the Wazuh rule firing, the correlation rule triggering, and the structured JSON alert in the dashboard. Understanding this complete pipeline from raw log to actionable alert is fundamental to detection engineering.

Cyberster Blue Team Internship — Week 2 Day 14

Active Response

Objective

The objective of Day 14 was to configure and test Wazuh's Active Response capability. Active Response allows the SIEM to automatically execute a defensive action whenever a specific rule fires. The goal was to automatically block the source IP of an SSH brute force attacker using the firewall-drop command for 300 seconds (5 minutes) as soon as Rule 100200 (SSH Brute Force Detected) triggered on the Kali Linux agent.

Tools Used

- Wazuh Manager (Ubuntu) — ossec.conf active-response block configuration
- Kali Linux Agent — target of the brute force attack and host of the firewall-drop script
- firewall-drop — Wazuh built-in active response script using iptables/ip6tables
- /var/ossec/logs/active-responses.log — active response execution log on the Kali agent
- ip6tables — verified the automatic firewall block rule was applied to the source IP

Procedure

Step 1: Install OpenSSH Server on Ubuntu Manager

For the brute force test, SSH needed to be running on the Ubuntu Manager (the target of the attack). OpenSSH server was installed and the service started.

```
sudo apt install openssh-server
sudo systemctl start ssh
sudo systemctl enable ssh
```

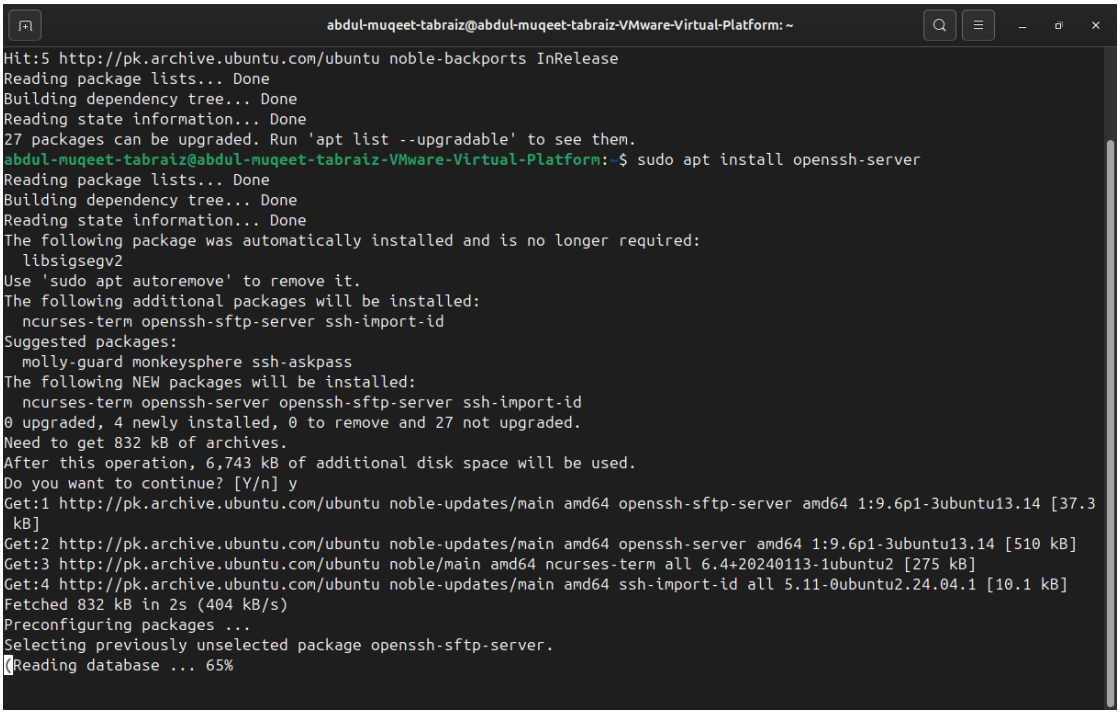

Fig: 14.1 Ubuntu Manager — sudo apt install openssh-server, package downloaded and installed

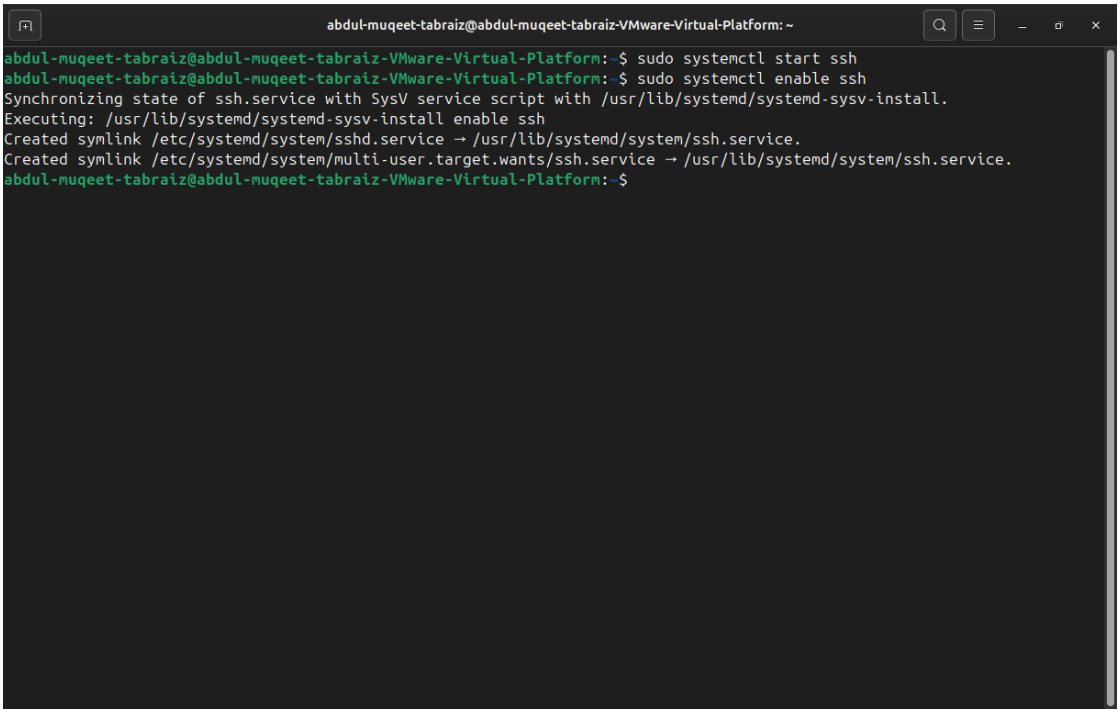

Fig: 14.2 Ubuntu Manager — SSH service started and enabled, systemd symlinks created for persistence

Step 2: Verify SSH Service Running on Ubuntu

The SSH service status was confirmed and the Ubuntu Manager's IP address verified to ensure the brute force target was reachable from Kali Linux.

```
sudo systemctl status ssh
```

```
ip a
```

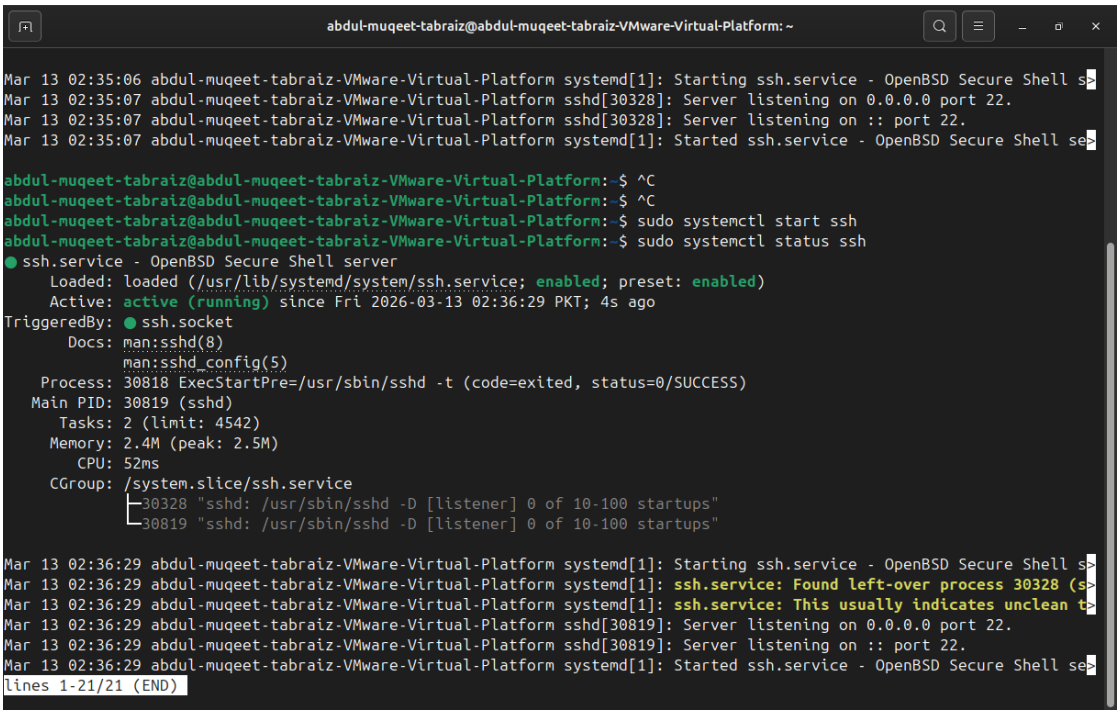

Fig: 14.3 Ubuntu SSH service status — active (running), listening on 0.0.0.0:22 and :::22

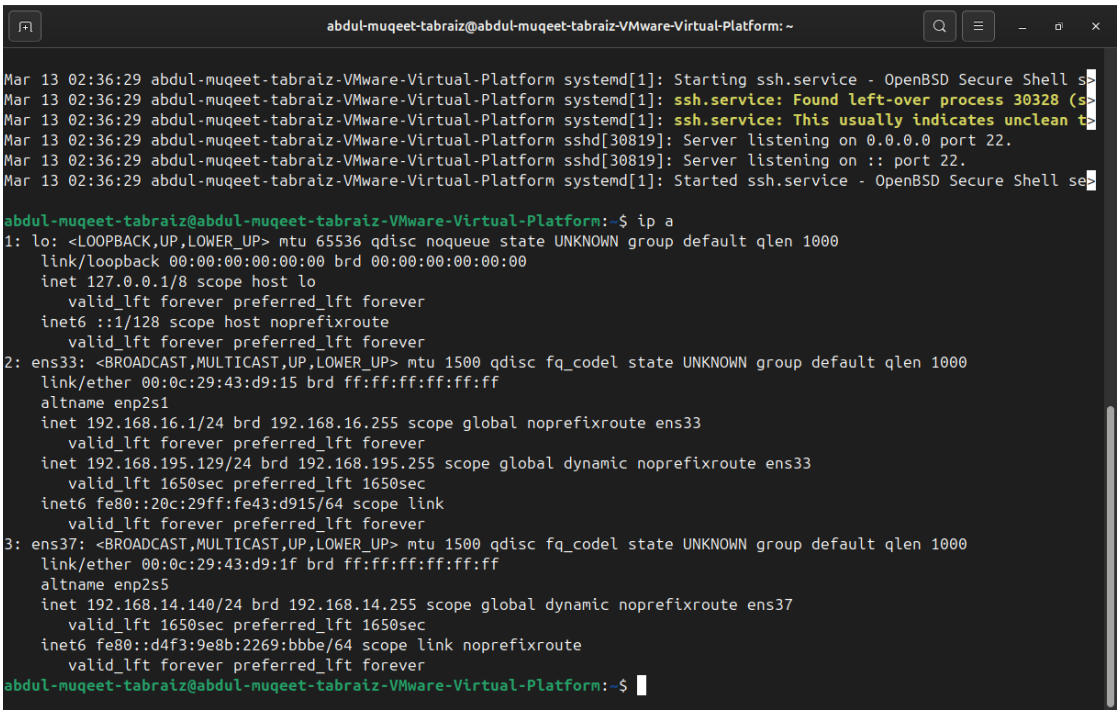

Fig: 14.4 Ubuntu ip a output — IP 192.168.14.140 on ens37 confirmed as brute force target address

Step 3: Add Active Response Block to ossec.conf on Ubuntu Manager

The active-response configuration block was added to the Wazuh Manager's ossec.conf. This block links the firewall-drop command to Rule 100200 with a 300-second (5 minute) automatic timeout.

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

```
<!-- Active Response Configuration -->
<!-- Fires firewall-drop when SSH brute force rule 100200 triggers -->
<!-- Blocks the source IP for 300 seconds (5 minutes) -->

<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>100200</rules_id>
  <timeout>300</timeout>
</active-response>
```

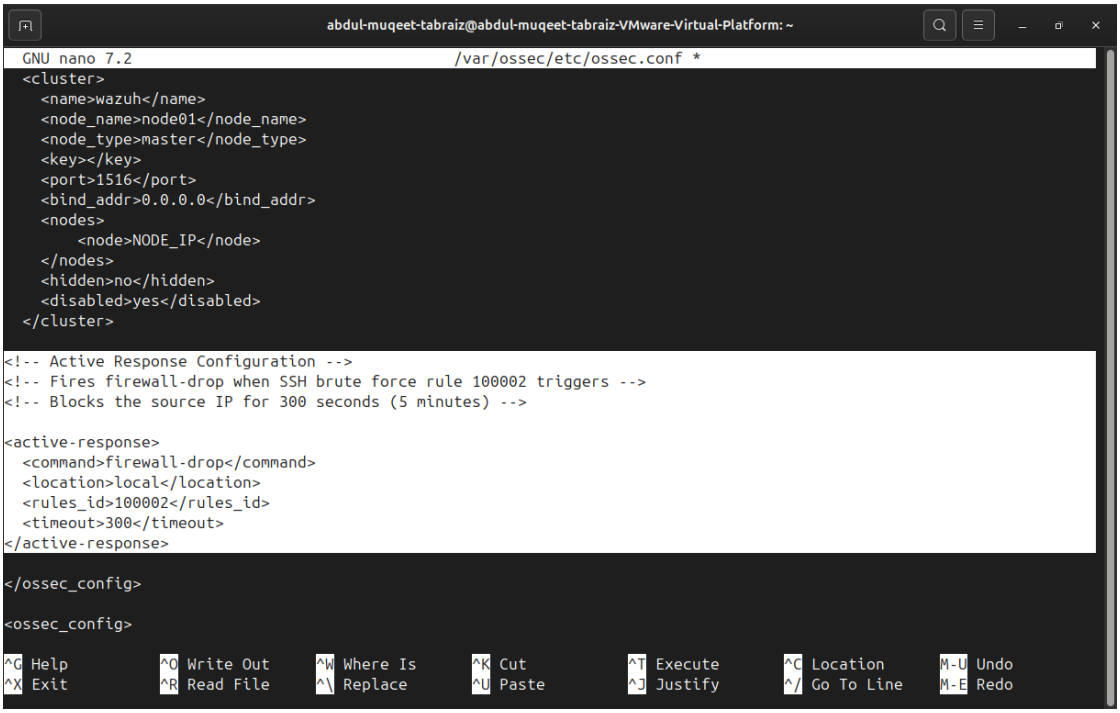

Fig: 14.5 Ubuntu ossec.conf — active-response block configured: firewall-drop on Rule 100200, 300-second timeout

Step 4: Restart Wazuh Manager to Apply Active Response Config

The Wazuh Manager was restarted to load the active-response configuration. The service status was verified after restart.

```
sudo systemctl restart wazuh-manager
sudo systemctl status wazuh-manager
```

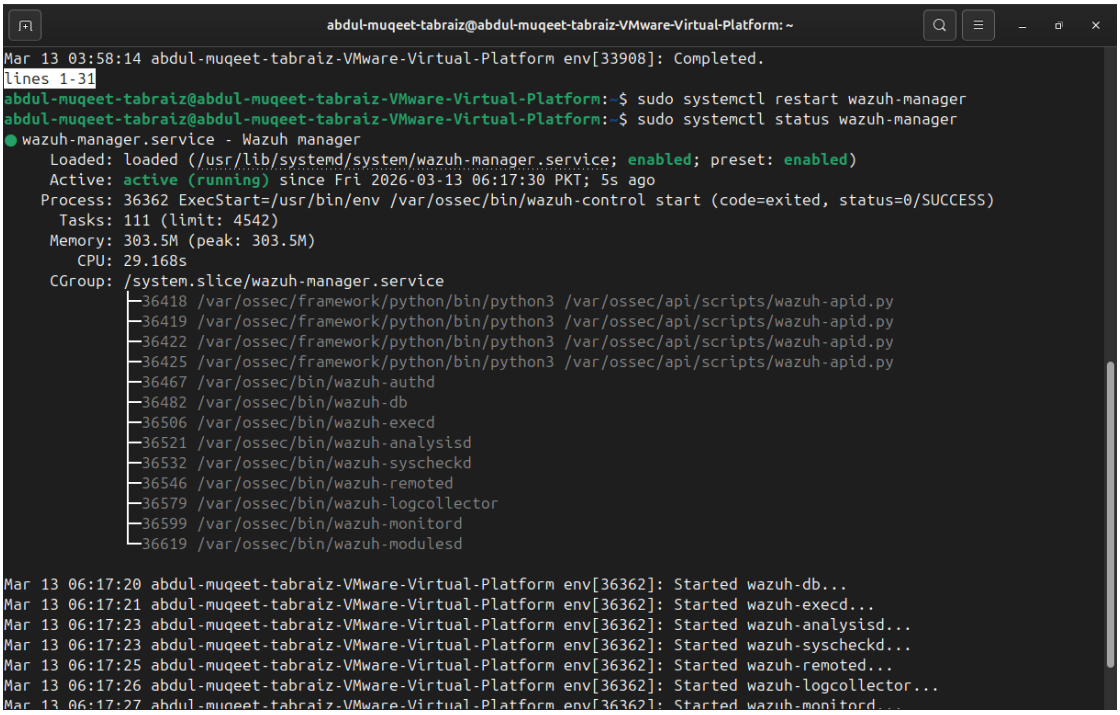

Fig: 14.6 Wazuh Manager restarted at 06:17 — active-response configuration loaded, all 111 tasks running

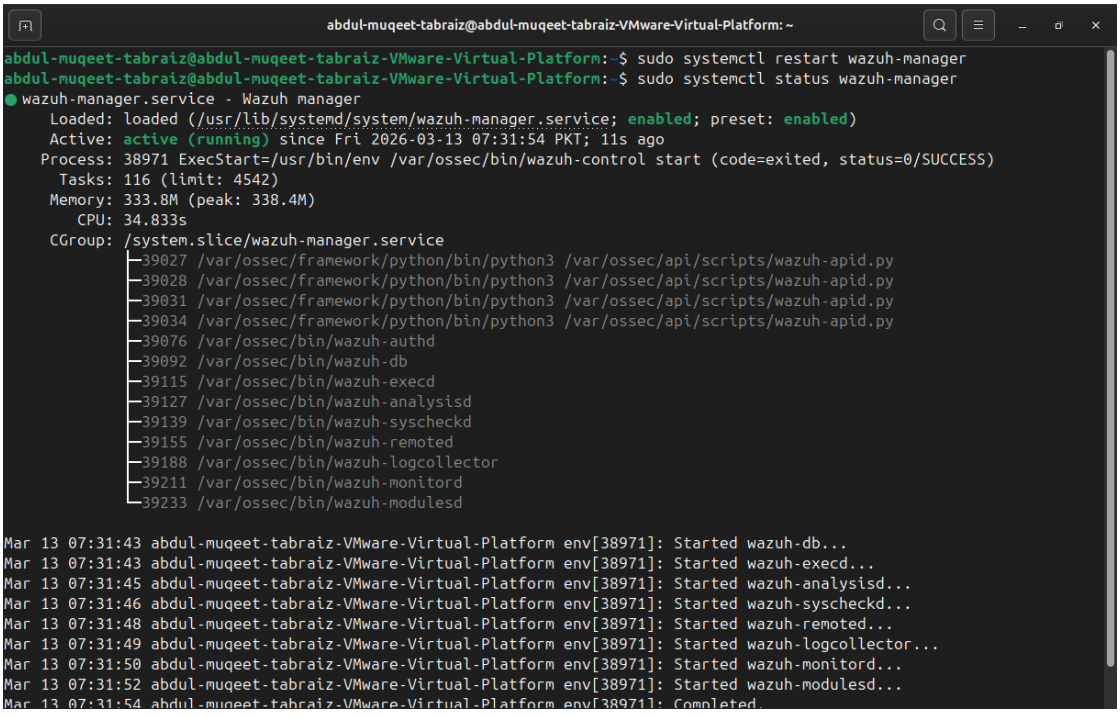

Fig: 14.7 Wazuh Manager final restart at 07:31 — fully operational, all services started including active response module

Step 5: Execute SSH Brute Force Loop from Kali Linux

A brute force loop was run from Kali Linux targeting localhost SSH to exceed the Rule 100200 frequency threshold and trigger the active response. The loop ran 10 SSH attempts with an invalid username.

```
for i in {1..10}; do ssh wronguser@localhost; done
```

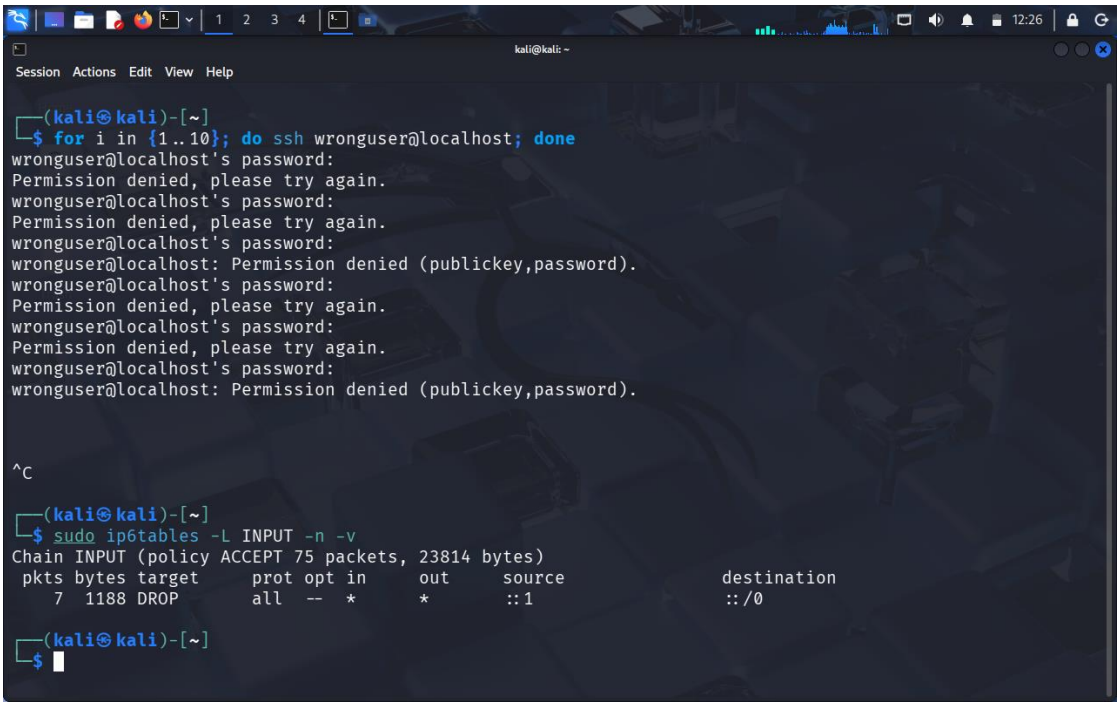

Fig: 14.8 Kali Linux — brute force loop executed; ip6tables INPUT showing DROP rule applied to ::1 (IPv6 localhost)

Step 6: Confirm Active Response Fired — Check active-responses.log

The active-responses.log on the Kali agent confirmed that the firewall-drop command was executed automatically. The log shows the complete JSON payload including the matched rule description, agent details, and source IP that was blocked.

```
sudo tail -f /var/ossec/logs/active-responses.log
```

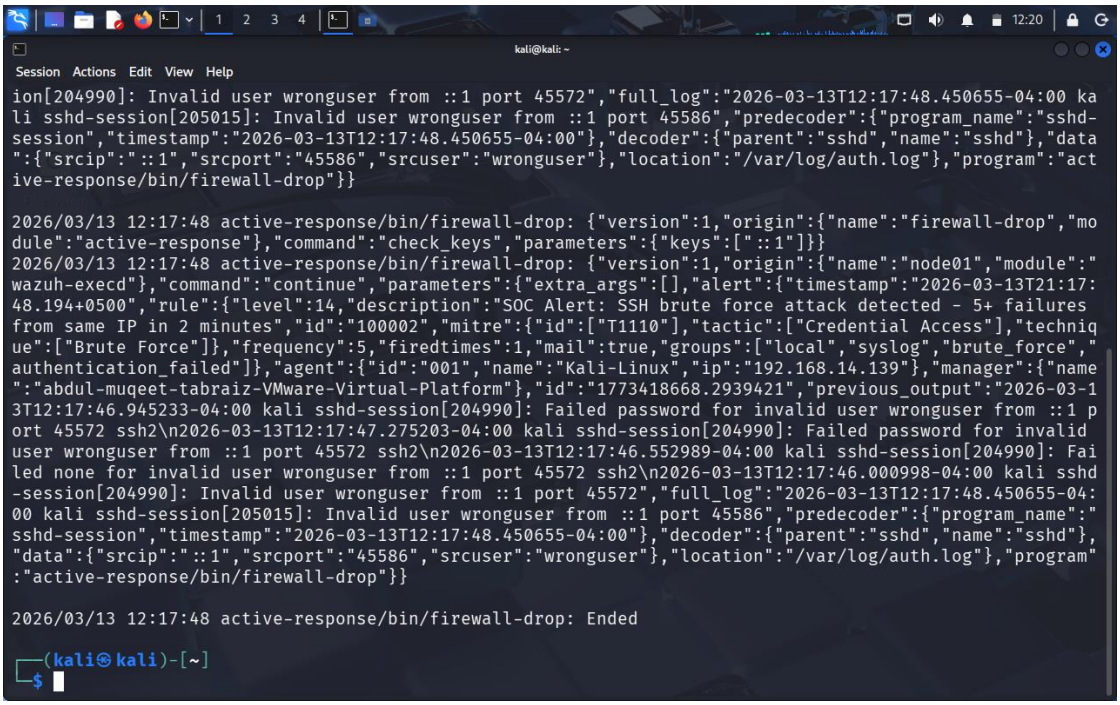

Fig: 14.9 Kali active-responses.log — firewall-drop executed, 'SOC Alert: SSH brute force attack detected' with source ::1 blocked and 'Ended' confirmation

Step 7: Verify ip6tables Block Applied

The firewall block was applied to ip6tables (not iptables) because the loopback address ::1 is an IPv6 address. The ip6tables INPUT chain showed a DROP rule for all traffic from ::1, confirming the active response was successful.

```
sudo ip6tables -L INPUT -n -v
```

Result confirmed: 7 packets, 1188 bytes — DROP — all — source ::1 — destination ::/0. The automatic block was applied within seconds of Rule 100200 firing (as shown in Fig 14.8 above). The block auto-lifted after the 300-second timeout.

Result

The complete Active Response chain was successfully demonstrated end to end:

- SSH brute force loop executed on Kali Linux exceeding the frequency threshold
- Rule 5710 fired on each individual SSH failure, Rule 100200 fired after the 6th failure within 60 seconds
- Wazuh Manager sent the firewall-drop command to the Kali Linux agent via wazuh-execd
- The firewall-drop script applied a DROP rule in ip6tables for source IP ::1
- The block was confirmed active in ip6tables with 7 packets already dropped
- The full execution was logged in /var/ossec/logs/active-responses.log on Kali
- The block automatically lifted after 300 seconds as configured

Summary of Day 14

Day 14 was the most impactful day of the week because it closed the loop from passive detection to automated response. Until this point, Wazuh had only been generating alerts a human analyst still needed to act. Active Response makes the SIEM autonomous for well-defined, high-confidence threat signatures.

The most important real-world lesson was that the block was applied to ip6tables instead of iptables because the loopback interface uses IPv6 (::1). In a production environment, if an active response appears not to be working, both iptables and ip6tables must be checked before concluding there is a failure. This is a common diagnostic mistake even among experienced administrators.

The 300-second auto-expiry is also a deliberate security design principle. Permanent blocks triggered by automated rules risk locking out legitimate users based on false positives. Timed blocks provide immediate protection while preserving the ability to restore access, with the SOC team able to escalate to permanent blocks only after manual confirmation of malicious intent.

Week 2 Conclusion

Week 2 of the Cyberster Blue Team Internship was a deep-dive into the advanced operational capabilities of Wazuh as a production SIEM. Across seven days and four major tasks, the internship progressed from basic agent log forwarding into real detection engineering, threat hunting, and automated incident response.

Key Accomplishments

- **— Deployed realtime FIM on Kali Linux (/etc, /var/log) and Windows (C:\Users\Public), verified with create/modify/delete test events and confirmed alerts in the dashboard.**File Integrity Monitoring
- **— Authored and deployed three custom rules: user account creation detection (Rule 100100), SSH brute force correlation (Rule 100200), and Windows USB insertion (Rule 100003), each with MITRE ATT&CK mapping.**Custom Rule Development
- **— Conducted a full threat hunting session using dashboard queries across authentication, FIM, and Windows security event categories, investigating the complete event chain from raw log to correlated alert.**Log Analysis & Threat Hunting
- **— Configured and verified automatic IP blocking via firewall-drop triggered by the SSH brute force rule, with 300-second timeout and ip6tables verification confirming the block was applied.**Active Response

Lab Environment

- Ubuntu (Wazuh Manager): IP 192.168.14.140 — Wazuh 4.7.5 Manager, Indexer, Dashboard
- Kali Linux (Agent 001): IP 192.168.14.139 — Linux endpoint, FIM, active response
- Windows (Agent 002): IP 192.168.14.1 — Windows 10 endpoint, FIM, USB monitoring

Key Lessons Learned

The most valuable outcome of Week 2 was developing the SOC analyst mindset, understanding not just how to use a tool but why each configuration decision matters for real threat detection. Every rule, every monitored directory, every active response timeout is a deliberate security decision with real operational consequences.

Detection engineering is not about writing as many rules as possible; it is about writing the right rules, for the right events, with the right thresholds. Wazuh's power comes not from its default ruleset but from the ability to extend it with environment-specific detections that match the actual threat landscape of the organization being protected.