[Local SOC Environment]

Selection of Network / System

System Overview:

The lab environment consists of a small, isolated network on the 192.168.80.0/24 subnet, designed to simulate a realistic SOC setup for penetration testing and detection analysis. At the center of the network is a Wazuh Manager (192.168.80.133) acting as the SIEM, connected to a switch that links all monitored endpoints. Three hosts—Windows 10 (192.168.80.130), CentOS (192.168.80.129), and Ubuntu (192.168.80.131)—are configured as target machines running Wazuh agents, each offering different services for attack and monitoring. A dedicated Kali 2024 attacker machine (192.168.80.132) is used to perform controlled offensive activities such as brute-force attempts, file tampering, and SQL injection tests. This setup allows full visibility of attacker actions and real-time detection across multiple operating systems within the same subnet.

Components and IP Configuration:

Component	OS	IP Address	Purpose
Wazuh Manager	Ubuntu 22.04	192.168.80.133	SOC / SIEM for detection
Windows 10	Windows 10 Pro	192.168.80.130	Endpoint monitored by Wazuh
CentOS	CentOS	192.168.80.129	Linux endpoint for monitoring/testing
Ubuntu	Ubuntu 22.04	192.168.80.131	Web/Service target for attacks
Kali Linux (Attacker)	Kali 2024	192.168.80.132	Attack source for penetration testing

Table 1: Shows the IP address and configurations of all VM's.

System Topology:

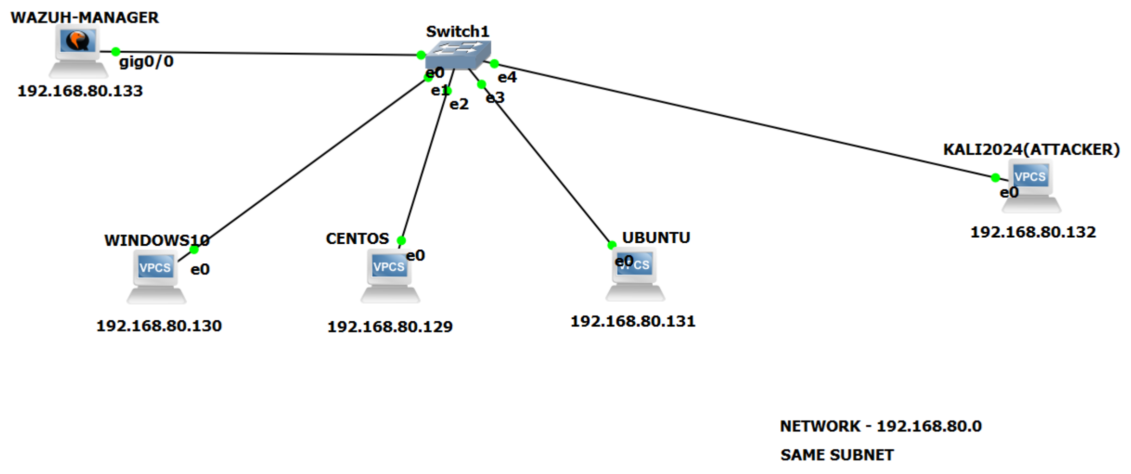


Figure 1: The above image shows topology made using GNS3 VM.

Environment Purpose: This lab was created to give me a realistic, hands-on SOC environment where I can see both the attack side and the detection side in one place. My Kali machine generates real attacks—like file changes, brute-force attempts, and SQL injection—while the Wazuh Manager collects logs from the Windows, Ubuntu, and CentOS systems and turns them into alerts. By using my own machines on the 192.168.80.0 network, I can clearly understand how different operating systems behave under attack and how a SIEM like Wazuh actually detects and reports them in real time.

Methodology

Testing Framework

For this lab, I followed the basic flow of the **Penetration Testing Execution Standard (PTES)** because it breaks a pentest into clear stages like recon, scanning, exploitation, and reviewing results [1]. This made the work easier to organise and helped me link every attack I performed with the alerts generated in Wazuh.

Approach Used in My Lab

All my testing was done from the **Kali machine (192.168.80.132)** against the Windows 10, CentOS, and Ubuntu hosts monitored by the **Wazuh Manager (192.168.80.133)**.
Here's a simple breakdown of what I actually did:

1. **Recon & Scanning:**
   I first checked which machines were active on the 192.168.80.0/24 network and scanned their open ports and services using Nmap [2].
2. **Preparing Wazuh:**
   Before attacking, I made sure Wazuh was watching the right log files and folders:
   a. Linux paths under <syscheck> for FIM
   b. Windows Desktop folder under <syscheck>
   c. /var/log/secure on CentOS for brute-force logs
   d. Apache access logs for SQLi detection
   (these were configured in each agent's ossec.conf)
3. **FIM Test:**
   I created, edited, and deleted a small test file inside monitored directories on both Linux and Windows. Wazuh picked up each action and showed it in the FIM alerts view.
4. **Brute-Force Test:**
   Using Hydra, I sent repeated failed login attempts to SSH on CentOS and RDP on Windows. These attempts appeared in /var/log/secure and Windows security logs, and Wazuh raised authentication-related alerts.
5. **SQL Injection Test:**
   On the Ubuntu web server, I used curl to send a SQLi-style request to the /users endpoint. Apache logged the request, and Wazuh flagged the suspicious SQL patterns from the access log.
6. **Reviewing Alerts:**
   After each PoC, I checked the Wazuh dashboard to confirm the alerts were triggered, looking at rule IDs, timestamps, the attacker IP, and severity.

Risk Rating

To keep things consistent, I rated each issue using **CVSS v3.1** [7]:

- **High:** 7.0–8.9
- **Medium:** 4.0–6.9
- **Low:** below 4

Tools Used in My Lab

This section explains the exact tools I used during my PoCs. I'm keeping it simple and focused on the real work I did.

Tools Overview

Tool / Component	Why I Used It	What It Helped Me Do
Nmap [2]	Starting point for every test.	Scanned the 192.168.80.0/24 network to discover hosts and open ports on Windows, Ubuntu, and CentOS.
Hydra [4]	To simulate brute-force behaviour.	Sent multiple failed login attempts to SSH/RDP so I could see how Wazuh reacts to authentication failures.
Wazuh Agents (Windows, Ubuntu, CentOS) [3]	To monitor each system.	Collected logs, file changes, and security events from every endpoint.
Wazuh Manager & Dashboard [3]	To verify detections.	Showed alerts for FIM changes, brute-force attempts, and SQLi patterns in real time.
Apache Web Server [5]	SQLi target on Ubuntu.	Logged all incoming requests, including SQLi payloads, which Wazuh then parsed.
curl	For SQL injection testing.	Sent the crafted SQLi request to the /users endpoint to generate detectable log entries.
Linux Logs & Windows Logs	Raw event data source.	/var/log/secure, Apache access.log, and Windows Desktop FIM events were used by Wazuh to trigger alerts.

Why These Tools Worked Well Together

I used this combination because it let me see the full picture:

- **Nmap** showed me what was exposed.
- **Hydra** created clear brute-force activity for Wazuh to pick up.
- **curl** helped me simulate SQL injection in a safe and simple way.
- **Apache**, Linux logs, and Windows logs served as the event sources.
- **Wazuh** tied everything together by collecting, correlating, and alerting on each activity.

This made the lab feel realistic — I could run attacks from Kali and then instantly see how a real SOC tool reacts.

<u>Findings Summary:</u> During testing, I generated three deliberate security events across my Windows, Ubuntu, and CentOS machines. Each one was designed to test how well Wazuh detects real attacker behaviour in a small SOC environment. The table below summarises the main findings, severity levels, and where the supporting evidence can be found in Appendix A.

Table 2: Summary of Findings from Local SOC Testing

Finding ID	Finding Name	Severity	Impact Summary	Evidence (Appendix A)	Recommendation
F1	File Integrity Monitoring (FIM) Detection	**Medium (CVSS 5.3)**	File tampering on monitored directories was detected immediately by Wazuh.	Figures **A2–A6**	Lock down file permissions, restrict access, and maintain strict FIM coverage.
F2	Brute-Force Authentication Attempt	**High (CVSS 8.0)**	Multiple failed SSH/RDP logins were flagged in real time, indicating possible credential-stuffing activity.	Figures **A7–A8**	Implement account lockouts, MFA, and IP throttling to slow brute-force attacks.
F3	SQL Injection Attempt Logged & Detected	**High (CVSS 7.5)**	Malicious SQLi request appeared in Apache logs and was flagged as suspicious by Wazuh.	Figures **A9–A10**	Sanitize inputs, use prepared statements, and monitor Apache logs for anomalies.

Detailed Discussion of Findings:

Below are short, direct explanations of what actually happened in each PoC and how Wazuh detected it.

<u>F1 – File Integrity Monitoring Detection (FIM):</u> To test FIM, I created, modified, and deleted test files in the monitored directories on Ubuntu, CentOS, and Windows. Wazuh immediately raised alerts showing the file path, action type (added/modified/deleted), and the corresponding agent.
These alerts matched every action I performed, proving the FIM configuration in ossec.conf was correct.
Evidence: Figures **A2–A6** in Appendix A.
Severity: Medium (5.3).
Medium: The attacker didn't gain privileges, but the ability to change system files is still a risk.
Recommendation: Restrict access to monitored directories, enforce strong permissions, and maintain FIM in real-time mode.

<u>F2 – Brute-Force Authentication Detection:</u> Using Hydra from the Kali machine, I simulated repeated failed login attempts to SSH on CentOS and RDP on Windows. These failures appeared in /var/log/secure (Linux) and Windows Security Logs, just as expected.
Wazuh collected these logs and raised brute-force alerts showing the attacker IP (192.168.80.132), number of failed attempts, and target host.
Evidence: Figures **A7–A8** in Appendix A.
Severity: High (8.0).
High: Brute-force attempts are a common path to credential compromise.
Recommendation: Use account lockouts, MFA, and rate-limiting. Consider firewall rules to block repeated failed attempts.

<u>F3 – SQL Injection Attempt Logged & Detected:</u> For SQLi testing, I used curl to send an intentionally malicious SQL payload to the /users' endpoint on the Ubuntu web server. Apache logged the request with the injected parameter, and Wazuh flagged it due to suspicious SQL syntax.
The detection confirmed that log collection from Apache was working correctly.
Evidence: Figures **A9–A10** in Appendix A.
Severity: High (7.5).
High: SQLi can expose or modify data if unpatched, even though my PoC was safe and controlled.
Recommendation: Sanitize all inputs, move to parameterized queries, enforce least-privilege DB accounts, and enable WAF-style filtering if possible.

Remediation, Conclusion, Reflection

Remediation: Based on the findings in my lab, these are the key improvements I would apply to strengthen the security of the environment:

- **Tighten file permissions** on Ubuntu, CentOS, and Windows to reduce the chance of unauthorized changes to monitored directories.
- **Enable account lockout policies and MFA** so brute-force attempts cannot continue endlessly.
- **Harden SSH and RDP exposure** by limiting access to specific IP ranges and enforcing stronger password policies.
- **Secure the web application** by using prepared statements, validating inputs, and restricting database privileges to the minimum required.
- **Tune Wazuh rules** to reduce noise and make critical alerts stand out more clearly.
- **Regularly review logs** (Apache, secure logs, Windows Security logs) to catch abnormal behaviour early.

Conclusion: Building this SOC lab helped me understand how attacker activities look from both sides — what I see on Kali, and what the SOC sees through Wazuh. All three PoCs (FIM, brute-force, SQLi) were detected correctly, which proved that my agent configurations and log sources were set up properly. Overall, the environment behaved like a small real-world SOC, and the detection flow from "attack → log → alert" was clear and easy to follow.

Reflection & Individual Contribution: This lab taught me how important proper log monitoring and configuration are in detection-based security. Setting up Wazuh agents on Windows, Ubuntu, and CentOS helped me practice real SIEM configuration, while the hands-on tests improved my understanding of how attacks appear inside logs. For this project, I personally:

- Built the entire Local SOC Lab with all five VMs.
- Configured Wazuh Manager and all Wazuh agents.
- Performed all PoCs: FIM, brute-force (Hydra), and SQL injection.
- Verified all alerts in the Wazuh dashboard.
- Wrote Pages 1–6 of my section and prepared Appendix A with screenshots.

This work gave me practical confidence in both offensive and defensive skills, especially in interpreting alerts the way real SOC analysts do.

Configuration Summary (Short, Humanised, Non-Generic): Before running any tests in my SOC lab, I had to make sure every machine was actually being watched by Wazuh. I started by updating the Wazuh agent configuration on each endpoint so the manager would receive the right logs and file-change events. On Ubuntu and CentOS, I enabled real-time file monitoring under the <syscheck> block and added both the Apache access logs and /var/log/secure into <localfile> so brute-force attempts and SQLi activity would show up immediately. On Windows, I monitored the Desktop folder because it gave me a simple, controlled place to generate FIM alerts without touching system files. After updating each ossec.conf, I restarted the Wazuh agents to apply the changes and confirmed that all three endpoints were actively sending logs to the manager. Once I saw each agent appear in the Wazuh dashboard, I knew the environment was correctly wired and ready for the attacks I later performed.
(Detailed commands, configurations, and screenshots for this setup are provided in **Appendix A**.)

References:

[1] PTES Technical Guidelines, “Penetration Testing Execution Standard,” 2024. [Online].

Available: https://www.pentest-standard.org/

[2] Nmap Project, “Nmap: Network Scanning Tool,” 2025. [Online].

Available: https://nmap.org/

[3] Wazuh, “File Integrity Monitoring (FIM),” 2025. [Online].

Available: https://documentation.wazuh.com/

[4] THC Hydra Project, “THC-Hydra – Network Logon Cracker,” 2025. [Online].

Available: https://github.com/vanhauser-thc/thc-hydra

[5] Apache Software Foundation, “Apache HTTP Server Project,” 2025. [Online].

Available: https://httpd.apache.org/

[6] FIRST, “CVSS v3.1: Specification Document,” 2019. [Online].

Available: https://www.first.org/cvss/specification-document

Appendix A — Ritesh Indore (Local SOC Lab Evidence): This appendix contains the complete step-by-step process, commands, configurations, and screenshots from my FIM, brute-force, and SQL injection PoCs. All evidence directly supports the findings discussed in the main report.

A.1 Wazuh Agent Installation Steps:

1. curl -sO https://packages.wazuh.com/4.12/wazuh-install.sh
2. bash wazuh-install.sh -a
3. bash wazuh-passwords-tool.sh -u admin -p <new_password>

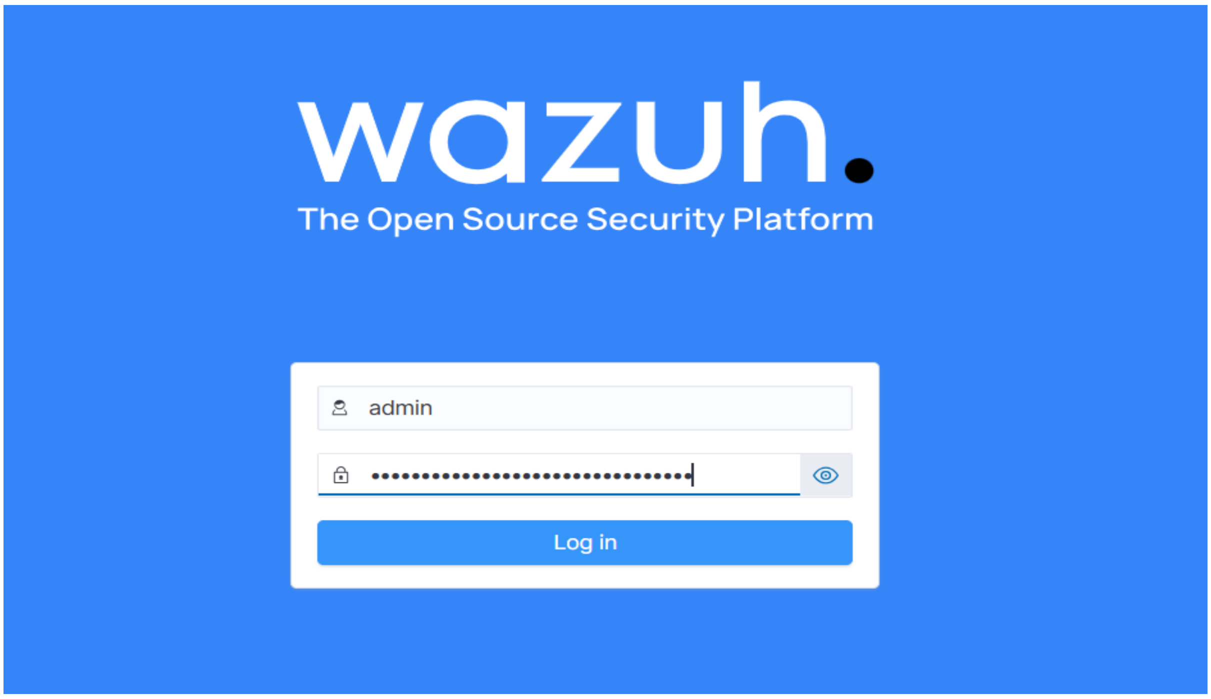


Figure A0 — Wazuh dashboard after installation

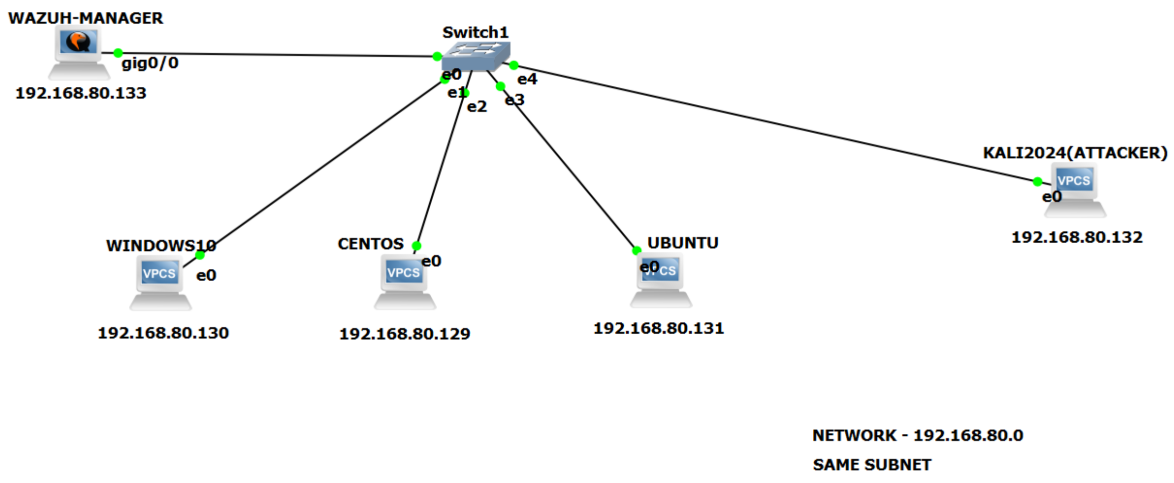


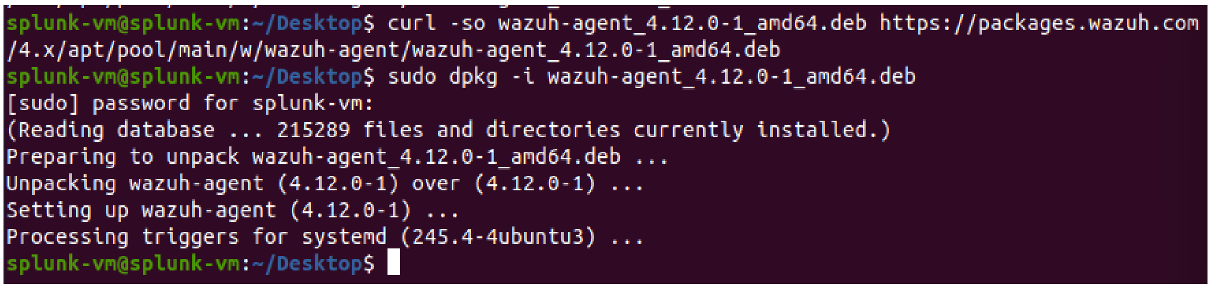
Figure A1 —Local SOC Lab Topology created in GNS3

Caption: This diagram shows my five-VM environment on the 192.168.80.0/24 network: Wazuh Manager, Windows 10, Ubuntu, CentOS, and the Kali attacker machine.

A.2 Ubuntu Agent Installation (Wazuh Agent 4.12)

Command used:

1. curl -so wazuh-agent_4.12.0-1_amd64.deb https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.12.0-1_amd64.deb
2. sudo dpkg -i wazuh-agent_4.12.0-1_amd64.deb



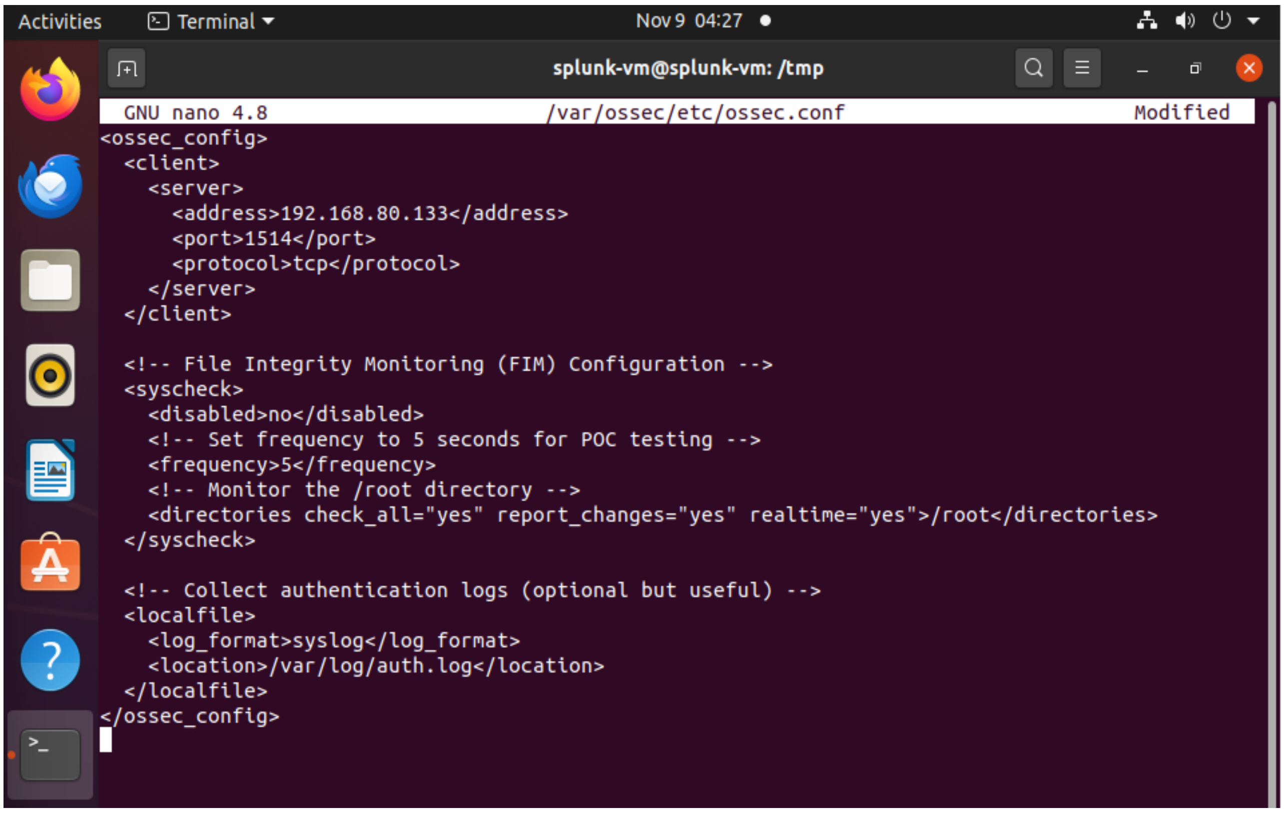
```
splunk-vm@splunk-vm:~/Desktop$ curl -so wazuh-agent_4.12.0-1_amd64.deb https://packages.wazuh.com
/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.12.0-1_amd64.deb
splunk-vm@splunk-vm:~/Desktop$ sudo dpkg -i wazuh-agent_4.12.0-1_amd64.deb
[sudo] password for splunk-vm:
(Reading database ... 215289 files and directories currently installed.)
Preparing to unpack wazuh-agent_4.12.0-1_amd64.deb ...
Unpacking wazuh-agent (4.12.0-1) over (4.12.0-1) ...
Setting up wazuh-agent (4.12.0-1) ...
Processing triggers for systemd (245.4-4ubuntu3) ...
splunk-vm@splunk-vm:~/Desktop$
```

Figure A2 — Installing the Wazuh Agent using dpkg

A.2.1 Configuring the Agent to Connect to the Wazuh Manager/File Integrity Monitoring:

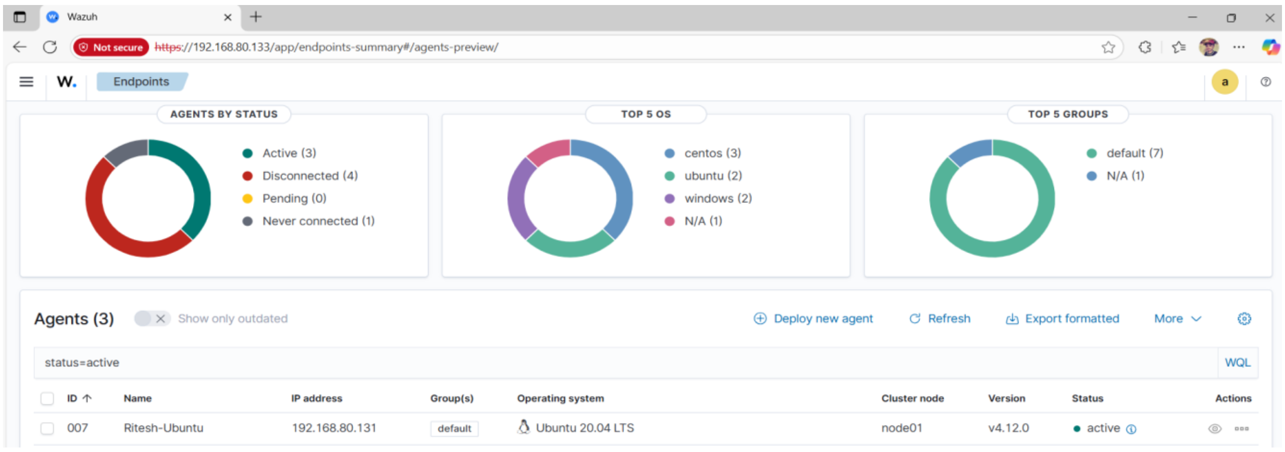
Command used:

1. sudo nano /var/ossec/etc/ossec.conf
2. sudo systemctl enable wazuh-agent
3. sudo systemctl start wazuh-agent
4. sudo systemctl status wazuh-agent



```
  GNU nano 4.8                    /var/ossec/etc/ossec.conf                    Modified
<ossec_config>
  <client>
    <server>
      <address>192.168.80.133</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
  </client>

  <!-- File Integrity Monitoring (FIM) Configuration -->
  <syscheck>
    <disabled>no</disabled>
    <!-- Set frequency to 5 seconds for POC testing -->
    <frequency>5</frequency>
    <!-- Monitor the /root directory -->
    <directories check_all="yes" report_changes="yes" realtime="yes">/root</directories>
  </syscheck>

  <!-- Collect authentication logs (optional but useful) -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>
</ossec_config>
```

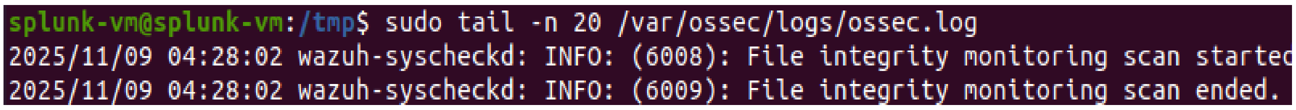
Figure A3: ossec.conf showing Wazuh Manager IP configured & FIM configuration



ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
007	Ritesh-Ubuntu	192.168.80.131	default	Ubuntu 20.04 LTS	node01	v4.12.0	active	

Figure A4: Verifying Ubuntu Agent Appears in Wazuh Dashboard

A.2.2 FIM Initialization in Log

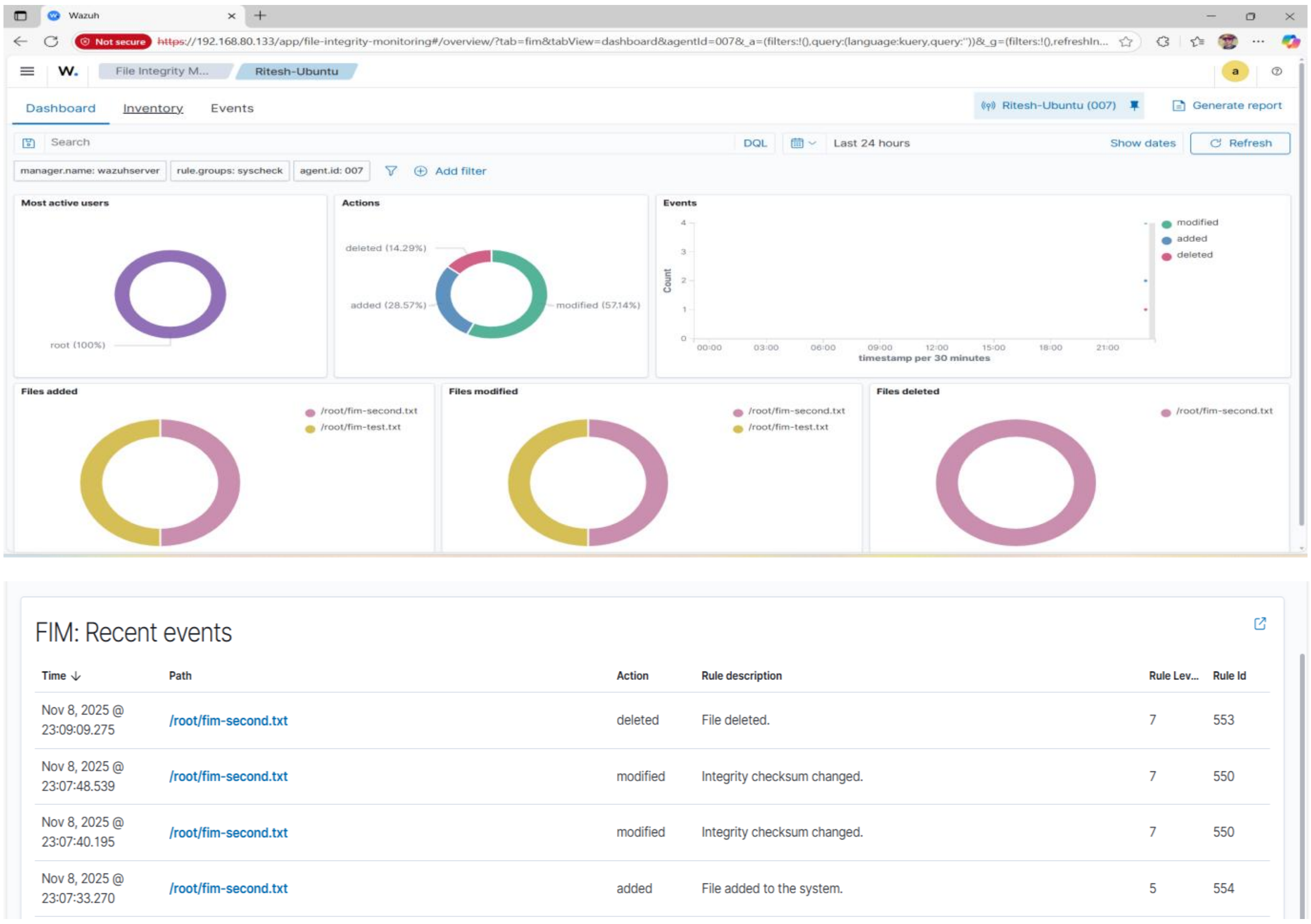


```
splunk-vm@splunk-vm:/tmp$ sudo tail -n 20 /var/ossec/logs/ossec.log
2025/11/09 04:28:02 wazuh-syscheckd: INFO: (6008): File integrity monitoring scan startec
2025/11/09 04:28:02 wazuh-syscheckd: INFO: (6009): File integrity monitoring scan ended.
```

Figure A5: The ossec.log confirms that File Integrity Monitoring (FIM) has started successfully

A.2.3 Ubuntu File Integrity Monitoring

Commands used: **1]** touch Second.txt **2]** echo "modified" >> /root/Second.txt **3]**rm /root/Second.txt



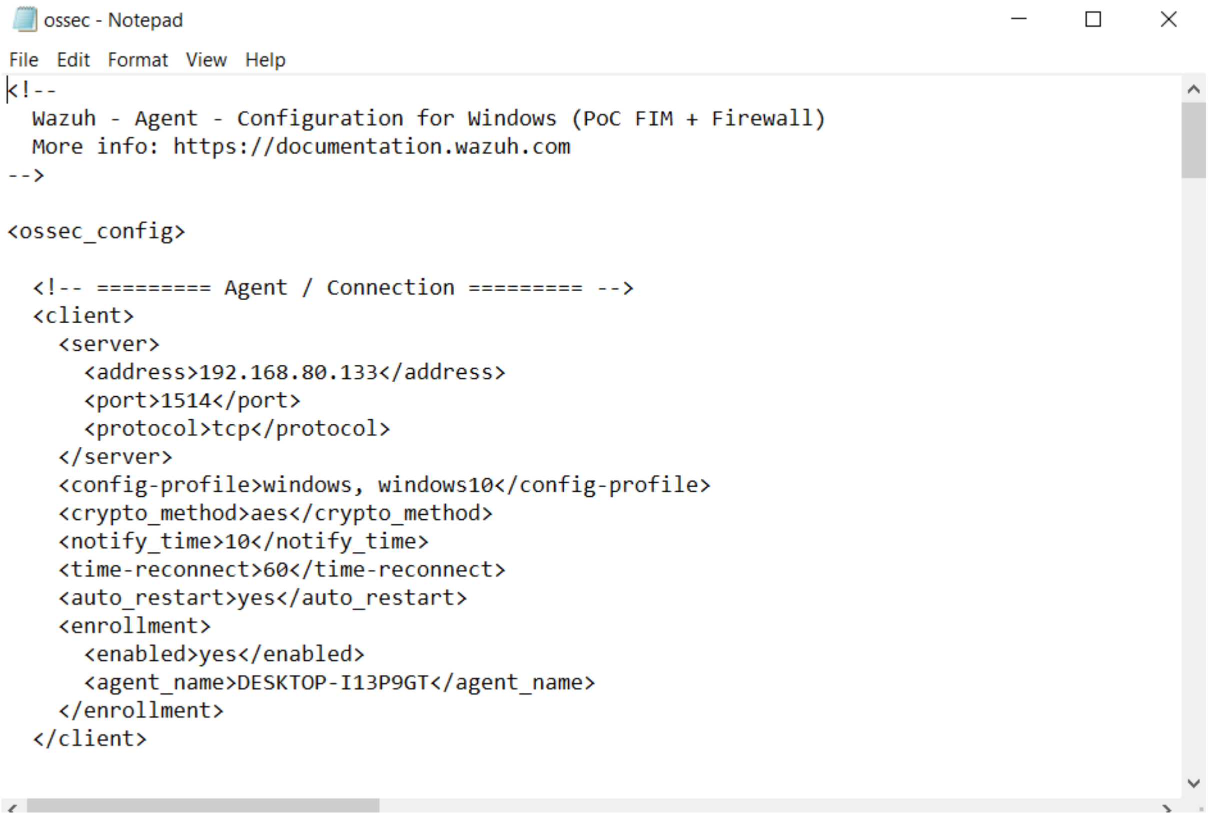
Time	Path	Action	Rule description	Rule Lev...	Rule Id
Nov 8, 2025 @ 23:09:09.275	/root/fim-second.txt	deleted	File deleted.	7	553
Nov 8, 2025 @ 23:07:48.539	/root/fim-second.txt	modified	Integrity checksum changed.	7	550
Nov 8, 2025 @ 23:07:40.195	/root/fim-second.txt	modified	Integrity checksum changed.	7	550
Nov 8, 2025 @ 23:07:33.270	/root/fim-second.txt	added	File added to the system.	5	554

Figure A6 — Wazuh FIM alert for file creation/modification on Ubuntu

A.3 Windows 10 Agent Installation/FIM POC

Commands used:

1. https://packages.wazuh.com/4.x/windows/wazuh-agent-4.12.0-1.msi
2. notepad "C:\Program Files (x86)\ossec-agent\ossec.conf"
3. Restart-Service -Name wazuh
4. Add the directories for monitoring within the <syscheck> block. For this use case, you configure Wazuh to monitor the C:\Users\<User_Name>\Desktop directory. <directories check_all="yes" report_changes="yes" realtime="yes">C:\Users\<USER_NAME>\Desktop</*directories*>
5. Restart-Service -Name Wazuh



```
<!--
  Wazuh - Agent - Configuration for Windows (PoC FIM + Firewall)
  More info: https://documentation.wazuh.com
-->

<ossec_config>

  <!-- ========= Agent / Connection ========= -->
  <client>
    <server>
      <address>192.168.80.133</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>windows, windows10</config-profile>
    <crypto_method>aes</crypto_method>
    <notify_time>10</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <enrollment>
      <enabled>yes</enabled>
      <agent_name>DESKTOP-I13P9GT</agent_name>
    </enrollment>
  </client>
```

Figure A7: Edit the C:\Program Files (x86)\ossec-agent\ossec.conf configuration file on the monitored Windows endpoint.

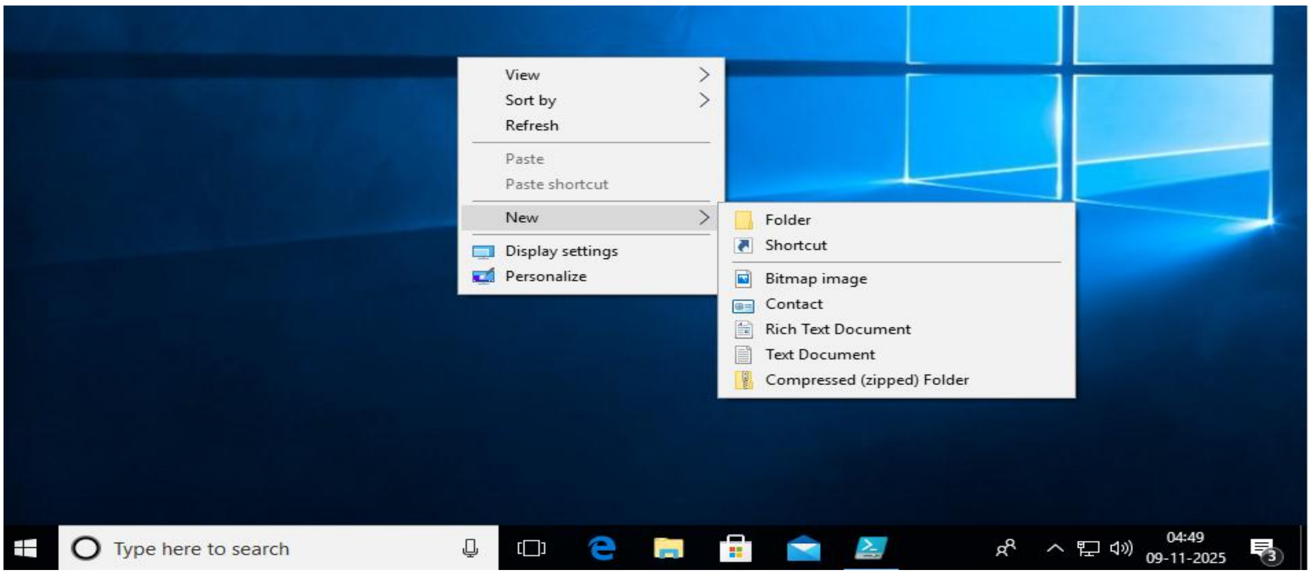
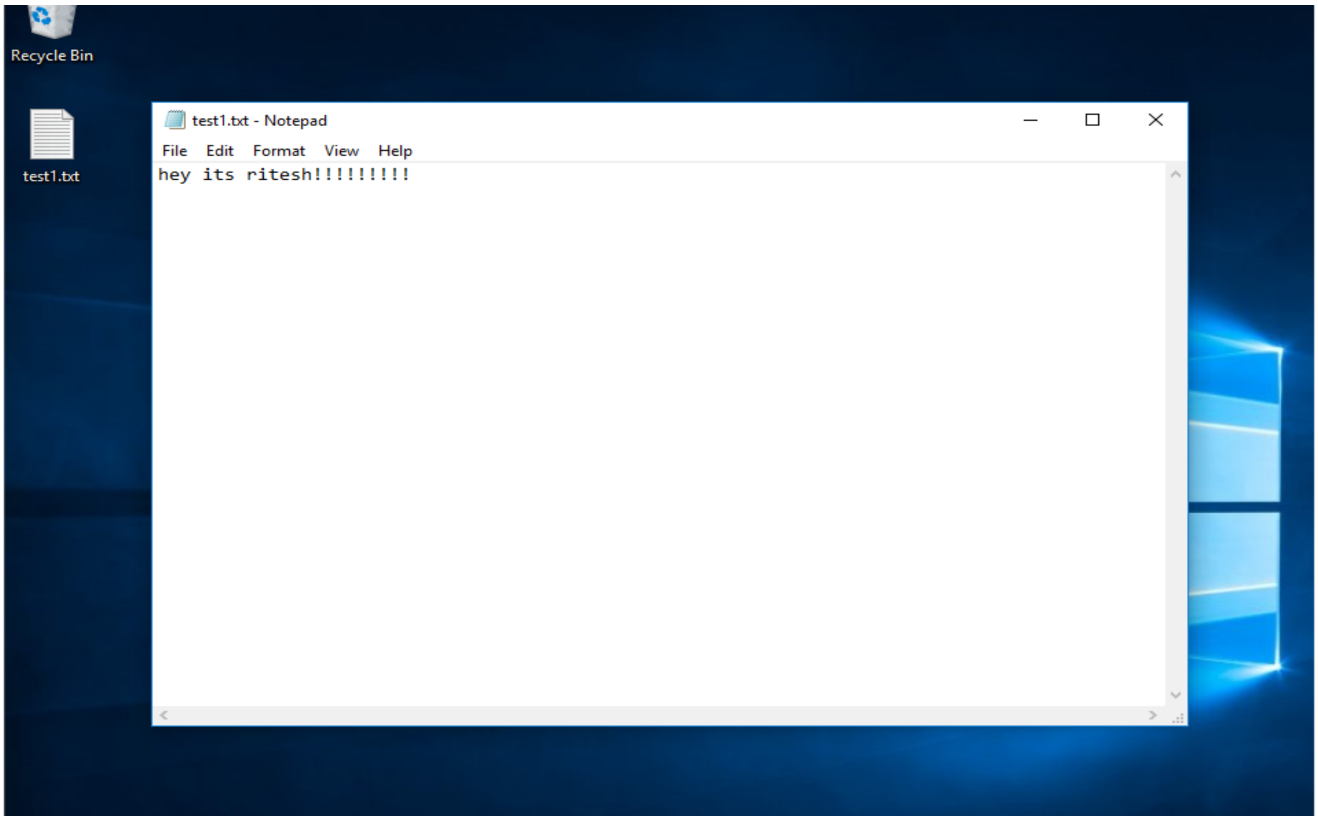
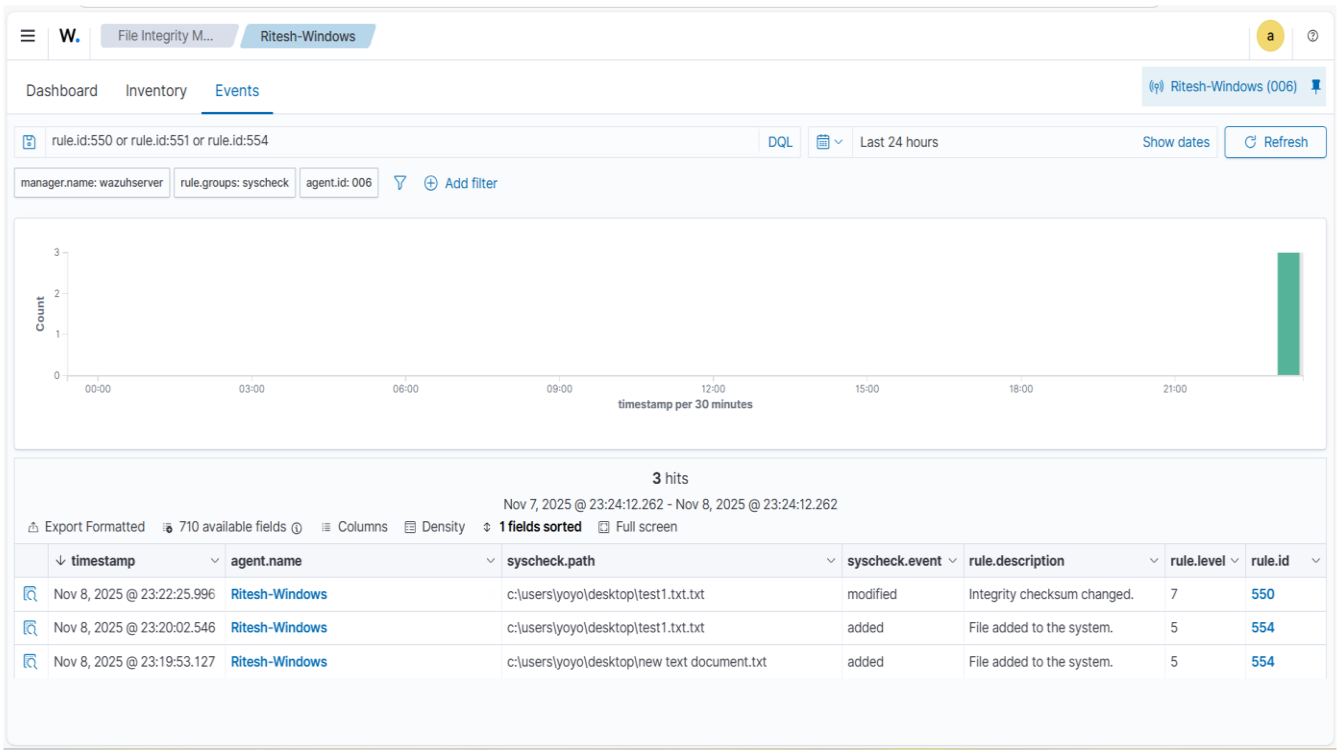


Figure A8: Shows file creation on monitored windows desktop.



```
hey its ritesh!!!!!!!!!
```

Figure A9: Shows File Modification on Monitored Desktop.



rule.id:550 or rule.id:551 or rule.id:554

manager.name: wazuhserver | rule.groups: syscheck | agent.id: 006

3 hits

Nov 7, 2025 @ 23:24:12.262 - Nov 8, 2025 @ 23:24:12.262

timestamp	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Nov 8, 2025 @ 23:22:25.996	Ritesh-Windows	c:\users\yoyo\desktop\test1.txt.txt	modified	Integrity checksum changed.	7	550
Nov 8, 2025 @ 23:20:02.546	Ritesh-Windows	c:\users\yoyo\desktop\test1.txt.txt	added	File added to the system.	5	554
Nov 8, 2025 @ 23:19:53.127	Ritesh-Windows	c:\users\yoyo\desktop\new text document.txt	added	File added to the system.	5	554

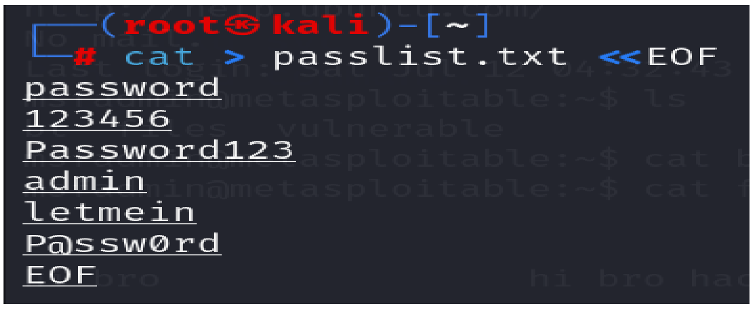
Figure A10: Shows Wazuh dashboard with File Creation/Modification alert

A.3.1 Windows Brute-Force Attack (Kali → Windows 10) – POC

Command used:

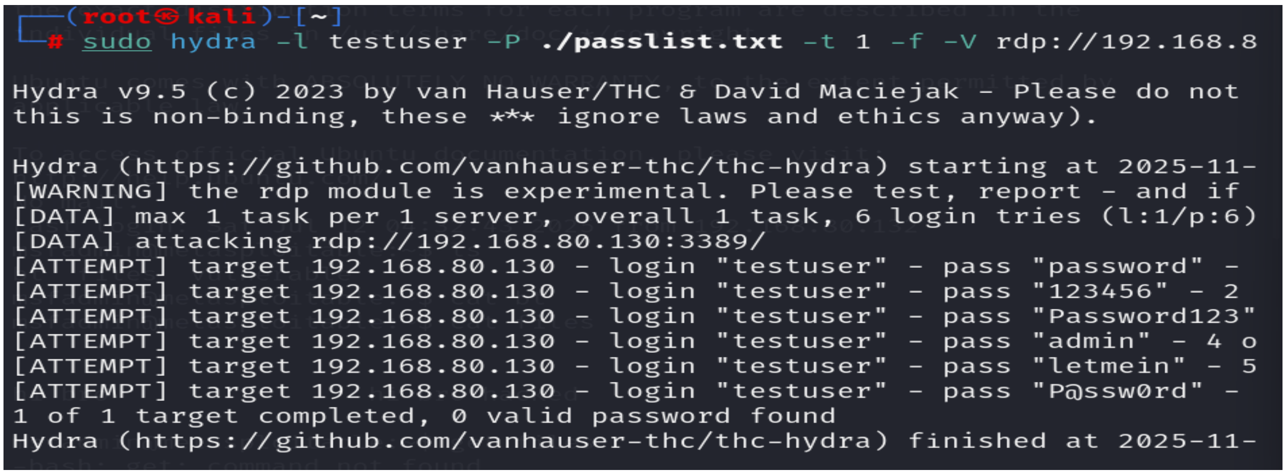
1. cat > passlist.txt <<EOF

   hydra -l Ritesh -P passlist.txt rdp://192.168.80.130



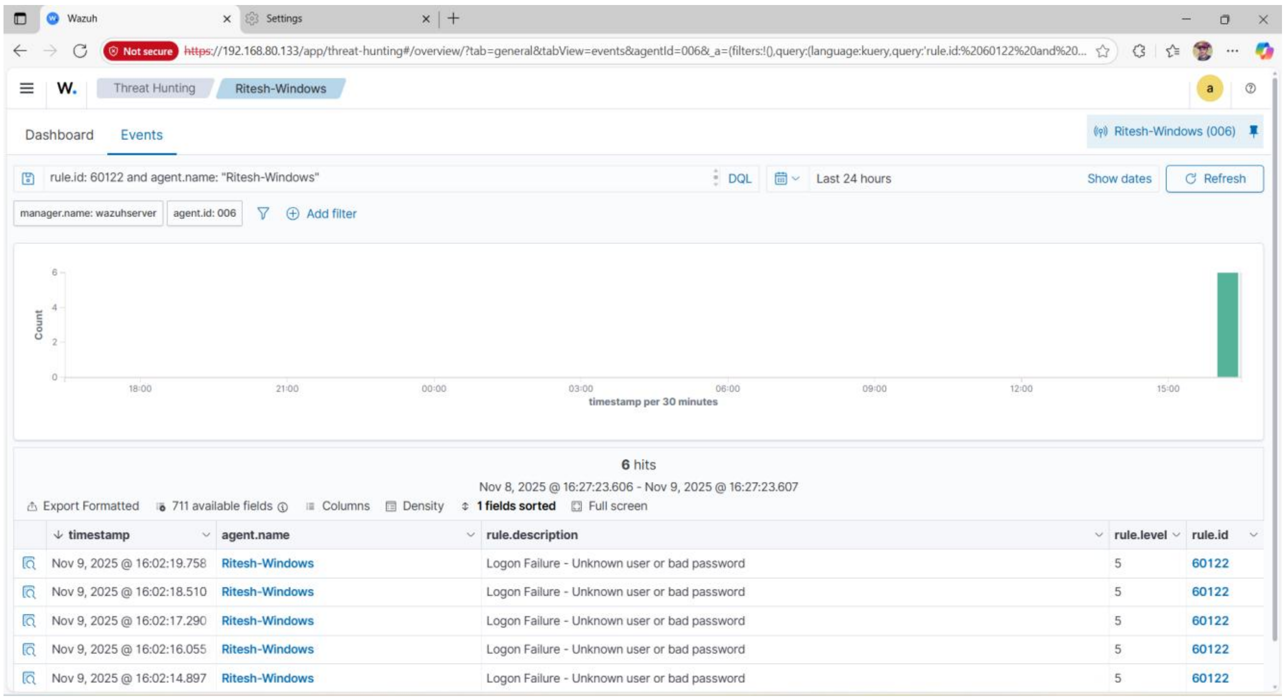
```
┌──(root㉿kali)-[~]
└─# cat > passlist.txt <<EOF
password
123456
Password123
admin
letmein
P@ssw0rd
EOF
```

Figure A11: Shows kali commands for creating file with the password list used for brute force



```
┌──(root㉿kali)-[~]
└─# sudo hydra -l testuser -P ./passlist.txt -t 1 -f -V rdp://192.168.8

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not
this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-
[WARNING] the rdp module is experimental. Please test, report - and if
[DATA] max 1 task per 1 server, overall 1 task, 6 login tries (l:1/p:6)
[DATA] attacking rdp://192.168.80.130:3389/
[ATTEMPT] target 192.168.80.130 - login "testuser" - pass "password" -
[ATTEMPT] target 192.168.80.130 - login "testuser" - pass "123456" - 2
[ATTEMPT] target 192.168.80.130 - login "testuser" - pass "Password123"
[ATTEMPT] target 192.168.80.130 - login "testuser" - pass "admin" - 4 o
[ATTEMPT] target 192.168.80.130 - login "testuser" - pass "letmein" - 5
[ATTEMPT] target 192.168.80.130 - login "testuser" - pass "P@ssw0rd" -
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-
```

Figure A12 — Hydra brute-force attempts running from Kali



rule.id: 60122 and agent.name: "Ritesh-Windows"

manager.name: wazuhserver | agent.id: 006

6 hits

Nov 8, 2025 @ 16:27:23.606 - Nov 9, 2025 @ 16:27:23.607

timestamp	agent.name	rule.description	rule.level	rule.id
Nov 9, 2025 @ 16:02:19.758	Ritesh-Windows	Logon Failure - Unknown user or bad password	5	60122
Nov 9, 2025 @ 16:02:18.510	Ritesh-Windows	Logon Failure - Unknown user or bad password	5	60122
Nov 9, 2025 @ 16:02:17.290	Ritesh-Windows	Logon Failure - Unknown user or bad password	5	60122
Nov 9, 2025 @ 16:02:16.055	Ritesh-Windows	Logon Failure - Unknown user or bad password	5	60122
Nov 9, 2025 @ 16:02:14.897	Ritesh-Windows	Logon Failure - Unknown user or bad password	5	60122

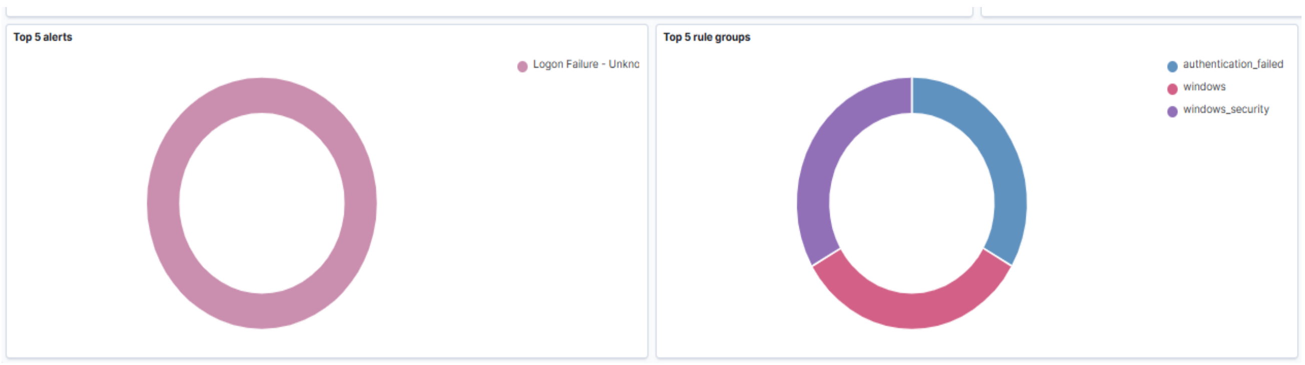


Figure A13: Shows Login Failure alerts on windows agent

A.3.2 Windows Nmap Scan Detection POC (Kali → Windows 10)

Commands used:

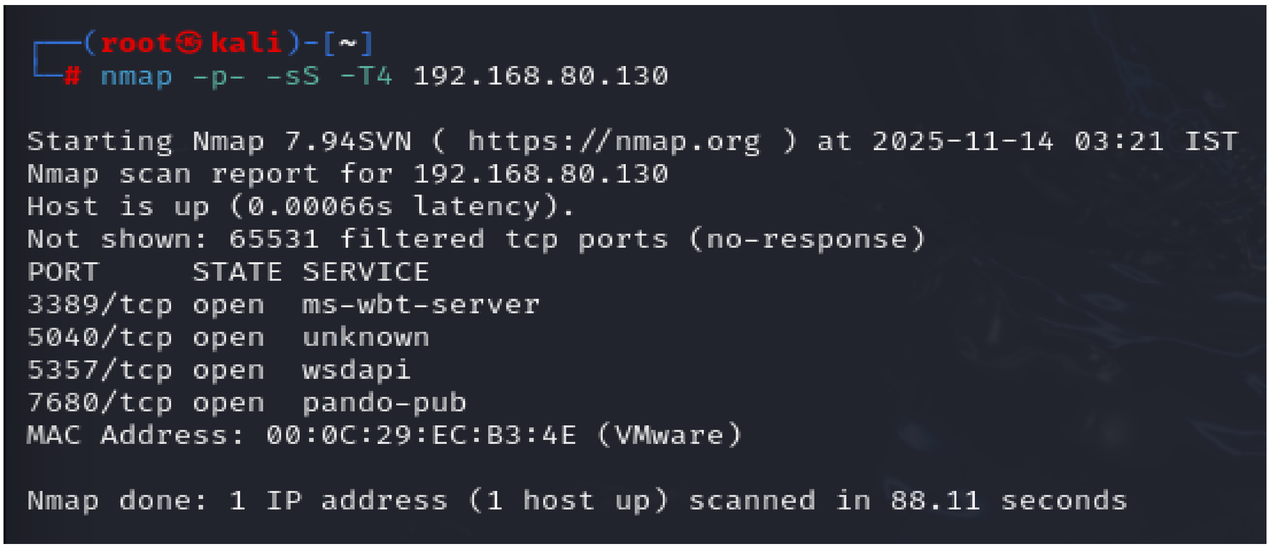
1] Start-Process notepad "C:\Program Files (x86)\ossec-agent\ossec.conf" -Verb RunAs

2] Inside <localfile> blocks, add the following:

```
<localfile>
  <location>Microsoft-Windows-Windows Firewall With Advanced Security/Firewall</location>
  <log_format>eventchannel</log_format>
</localfile>
<localfile>
  <location>System</location>  <log_format>eventchannel</log_format>
</localfile>
```

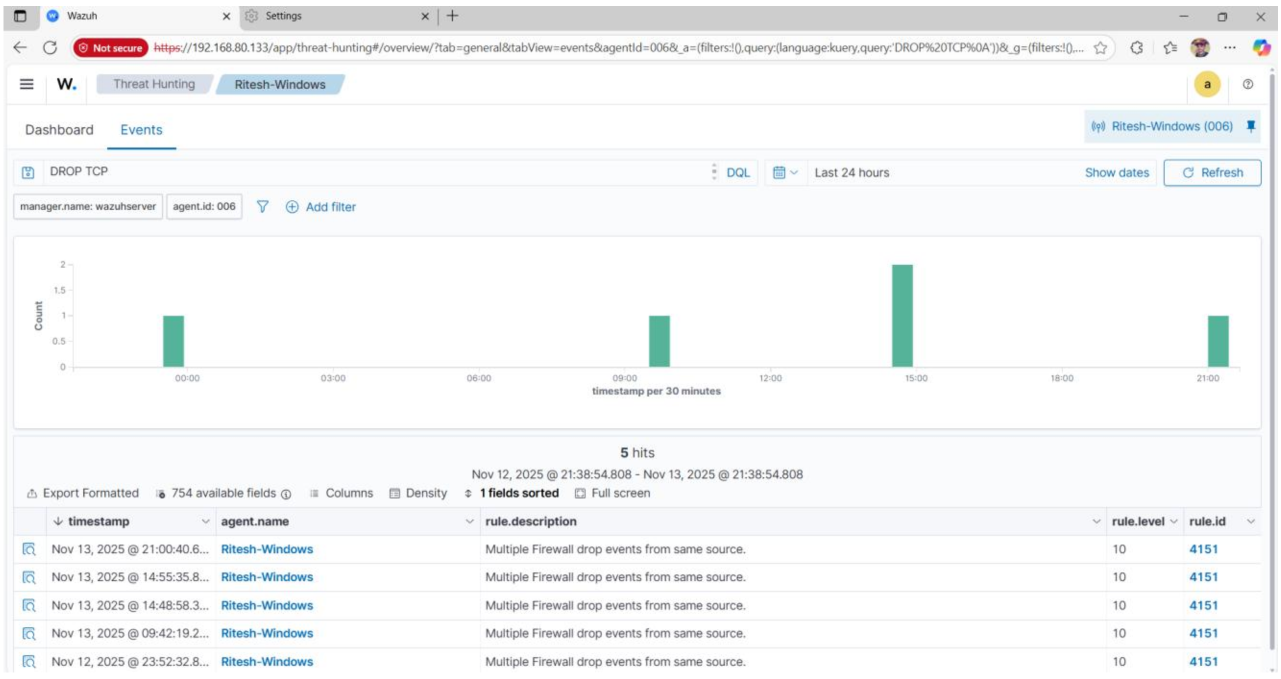
3] Restart-Service -Name wazuh

4] nmap -p- -sS -T4 192.168.80.130



```
┌──(root㉿kali)-[~]
└─# nmap -p- -sS -T4 192.168.80.130

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-11-14 03:21 IST
Nmap scan report for 192.168.80.130
Host is up (0.00066s latency).
Not shown: 65531 filtered tcp ports (no-response)
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server
5040/tcp open  unknown
5357/tcp open  wsdapi
7680/tcp open  pando-pub
MAC Address: 00:0C:29:EC:B3:4E (VMware)

Nmap done: 1 IP address (1 host up) scanned in 88.11 seconds
```

Figure A14: Shows Nmap scan done by kali on windows IP.



DROP TCP

manager.name: wazuhserver | agent.id: 006

5 hits

Nov 12, 2025 @ 21:38:54.808 - Nov 13, 2025 @ 21:38:54.808

timestamp	agent.name	rule.description	rule.level	rule.id
Nov 13, 2025 @ 21:00:40.6...	Ritesh-Windows	Multiple Firewall drop events from same source.	10	4151
Nov 13, 2025 @ 14:55:35.8...	Ritesh-Windows	Multiple Firewall drop events from same source.	10	4151
Nov 13, 2025 @ 14:48:58.3...	Ritesh-Windows	Multiple Firewall drop events from same source.	10	4151
Nov 13, 2025 @ 09:42:19.2...	Ritesh-Windows	Multiple Firewall drop events from same source.	10	4151
Nov 12, 2025 @ 23:52:32.8...	Ritesh-Windows	Multiple Firewall drop events from same source.	10	4151

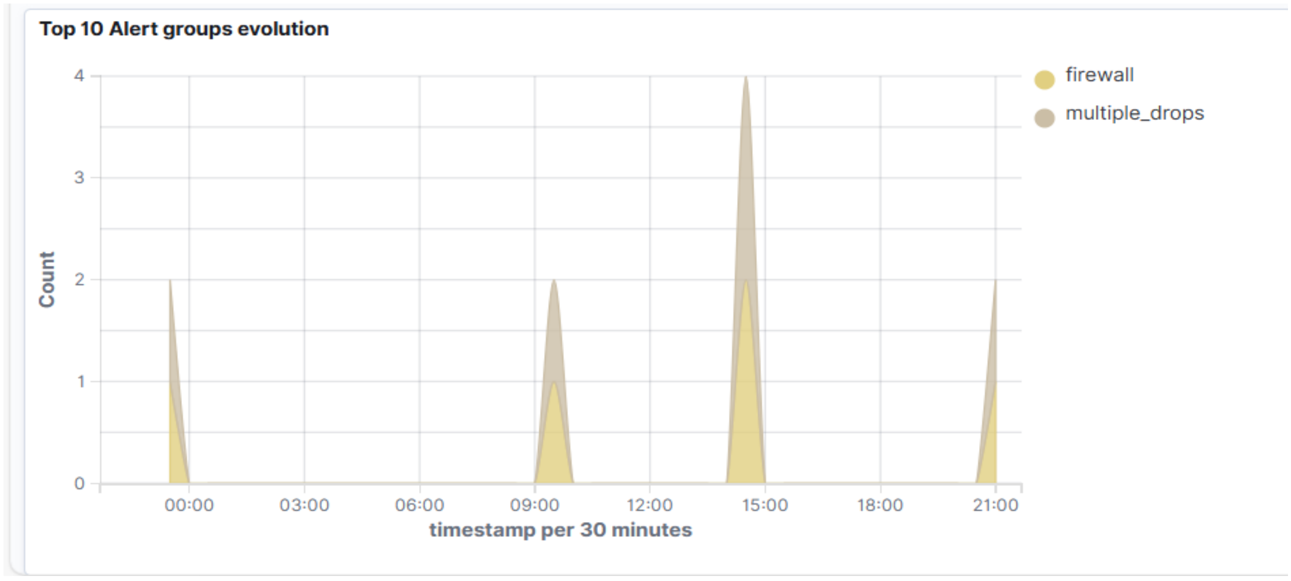
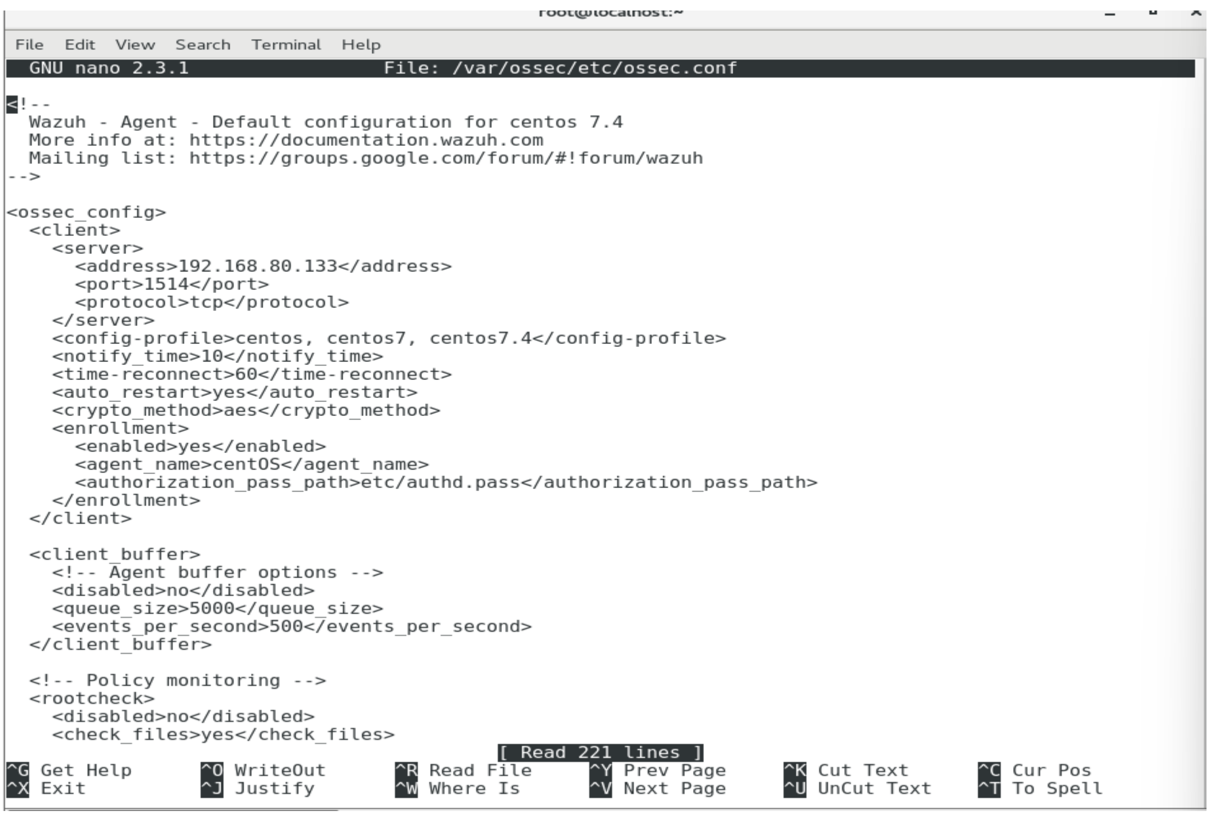


Figure A15: Shows Wazuh alert for Port scan.

A.4 CentOS Wazuh Agent Installation & FIM POC

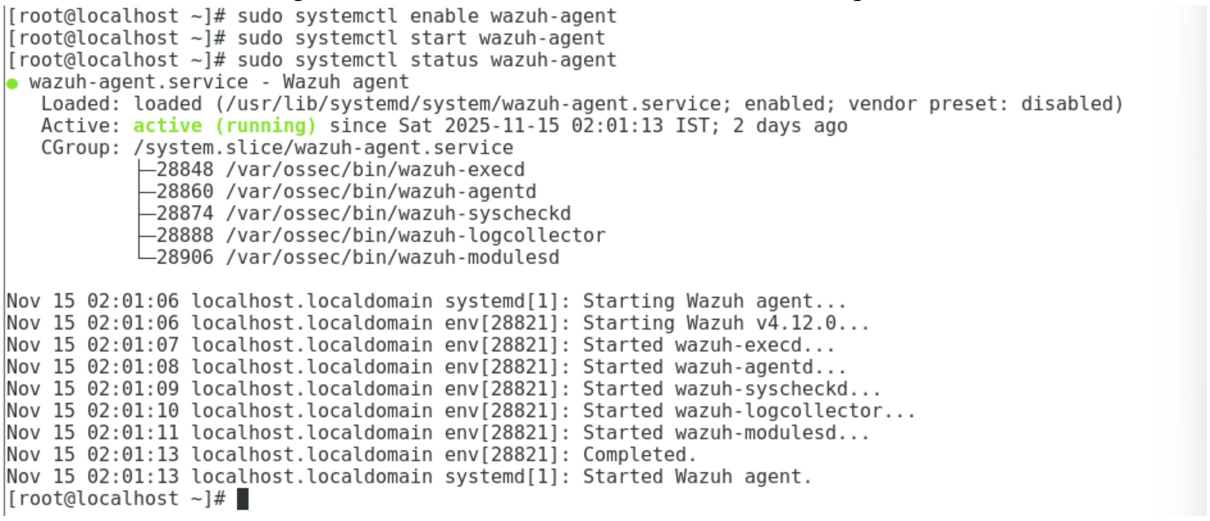
Commands used:

- curl -so wazuh-agent-4.12.0-1.x86_64.rpm https://packages.wazuh.com/4.x/yum/wazuh-agent-4.12.0-1.x86_64.rpm (Download agent directly)
- sudo rpm -ivh wazuh-agent-4.12.0-1.x86_64.rpm (Install the agent)
- sudo nano /var/ossec/etc/ossec.conf
- sudo systemctl start wazuh-agent
- sudo systemctl status wazuh-agent
- touch fim-centos.txt
- echo "Hello" > /etc/fim-centos.txt
- rm /etc/fim-centos.txt



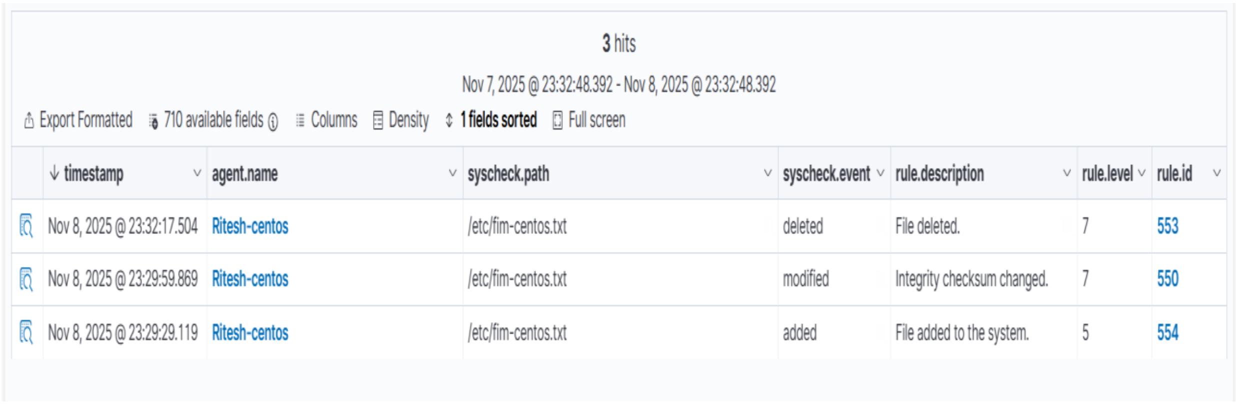
```
root@localhost:~
File  Edit  View  Search  Terminal  Help
  GNU nano 2.3.1                 File: /var/ossec/etc/ossec.conf

<!--
  Wazuh - Agent - Default configuration for centos 7.4
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <client>
    <server>
      <address>192.168.80.133</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>centos, centos7, centos7.4</config-profile>
    <notify_time>10</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
    <enrollment>
      <enabled>yes</enabled>
      <agent_name>centOS</agent_name>
      <authorization_pass_path>etc/authd.pass</authorization_pass_path>
    </enrollment>
  </client>

  <client_buffer>
    <!-- Agent buffer options -->
    <disabled>no</disabled>
    <queue_size>5000</queue_size>
    <events_per_second>500</events_per_second>
  </client_buffer>

  <!-- Policy monitoring -->
  <rootcheck>
    <disabled>no</disabled>
    <check_files>yes</check_files>
                                   [ Read 221 lines ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

Figure A16: Shows the ossec.conf edited to add manager and FIM rules.



```
[root@localhost ~]# sudo systemctl enable wazuh-agent
[root@localhost ~]# sudo systemctl start wazuh-agent
[root@localhost ~]# sudo systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2025-11-15 02:01:13 IST; 2 days ago
   CGroup: /system.slice/wazuh-agent.service
           ├─28848 /var/ossec/bin/wazuh-execd
           ├─28860 /var/ossec/bin/wazuh-agentd
           ├─28874 /var/ossec/bin/wazuh-syscheckd
           ├─28888 /var/ossec/bin/wazuh-logcollector
           └─28906 /var/ossec/bin/wazuh-modulesd

Nov 15 02:01:06 localhost.localdomain systemd[1]: Starting Wazuh agent...
Nov 15 02:01:06 localhost.localdomain env[28821]: Starting Wazuh v4.12.0...
Nov 15 02:01:07 localhost.localdomain env[28821]: Started wazuh-execd...
Nov 15 02:01:08 localhost.localdomain env[28821]: Started wazuh-agentd...
Nov 15 02:01:09 localhost.localdomain env[28821]: Started wazuh-syscheckd...
Nov 15 02:01:10 localhost.localdomain env[28821]: Started wazuh-logcollector...
Nov 15 02:01:11 localhost.localdomain env[28821]: Started wazuh-modulesd...
Nov 15 02:01:13 localhost.localdomain env[28821]: Completed.
Nov 15 02:01:13 localhost.localdomain systemd[1]: Started Wazuh agent.
[root@localhost ~]#
```

Figure A17: Shows agent successfully started after configuration.



3 hits

Nov 7, 2025 @ 23:32:48.392 - Nov 8, 2025 @ 23:32:48.392

timestamp	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Nov 8, 2025 @ 23:32:17.504	Ritesh-centos	/etc/fim-centos.txt	deleted	File deleted.	7	553
Nov 8, 2025 @ 23:29:59.869	Ritesh-centos	/etc/fim-centos.txt	modified	Integrity checksum changed.	7	550
Nov 8, 2025 @ 23:29:29.119	Ritesh-centos	/etc/fim-centos.txt	added	File added to the system.	5	554

Figure A18: Shows Wazuh Dashboard showing alert for file creation deletion.