WazuGuardix Training Program

Prepared By: Muhammad Safwan

Date: 9th November 2025

A Practical SOC Training Program Covering Wazuh

Purpose:

To empower students with hands-on experience in **SOC operations, threat detection, and automation** using Wazuh and integrated tools, preparing them for real-world cybersecurity challenges.

Scope:

This training includes **Wazuh server setup, static IP configuration, agent deployment, File Integrity Monitoring, Windows Defender log integration, SSH brute-force detection, dashboard navigation, email alert setup, and custom rule modification.** Each lab is designed to simulate real SOC scenarios and enhance operational skills.

Intended Audience:

Ideal for **students, SOC beginners, and cybersecurity enthusiasts** looking to build a strong technical foundation and gain practical skills in **security monitoring and automation.**

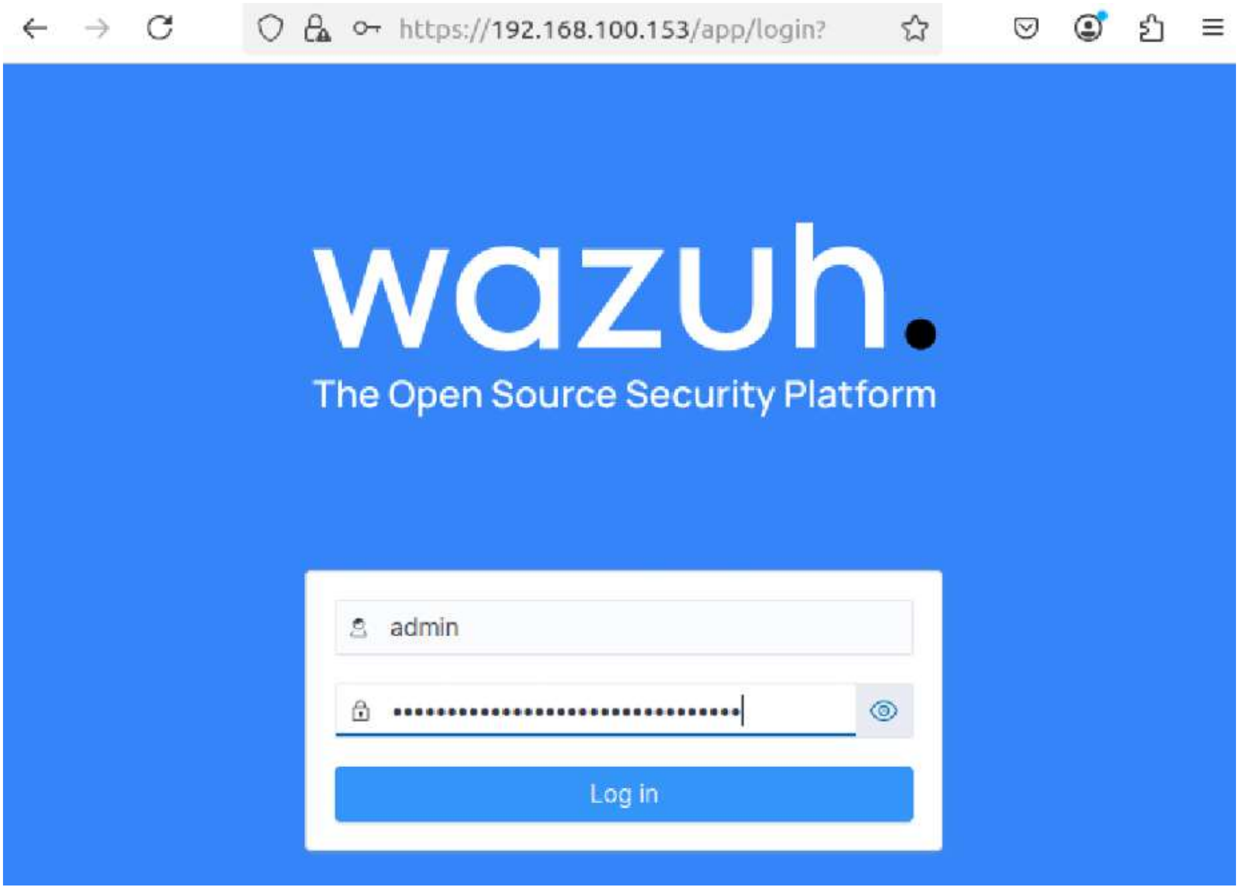
Document Version: 1.0

Lab 01: Wazuh Server Installation

- Install the Wazuh Server and ensure all services are running properly.
- Verify the Wazuh Manager, Indexer, and Dashboard installation.
- Submit screenshots of successful installation and running services.

The screenshot below shows the Wazuh web login page (large Wazuh branded splash) with admin username populated and a password entered. Browser address bar shows **https://192.168.100.153/** (the Wazuh host IP used in deployment).

This confirms the Dashboard is reachable and the web interface was installed and presented for authentication.

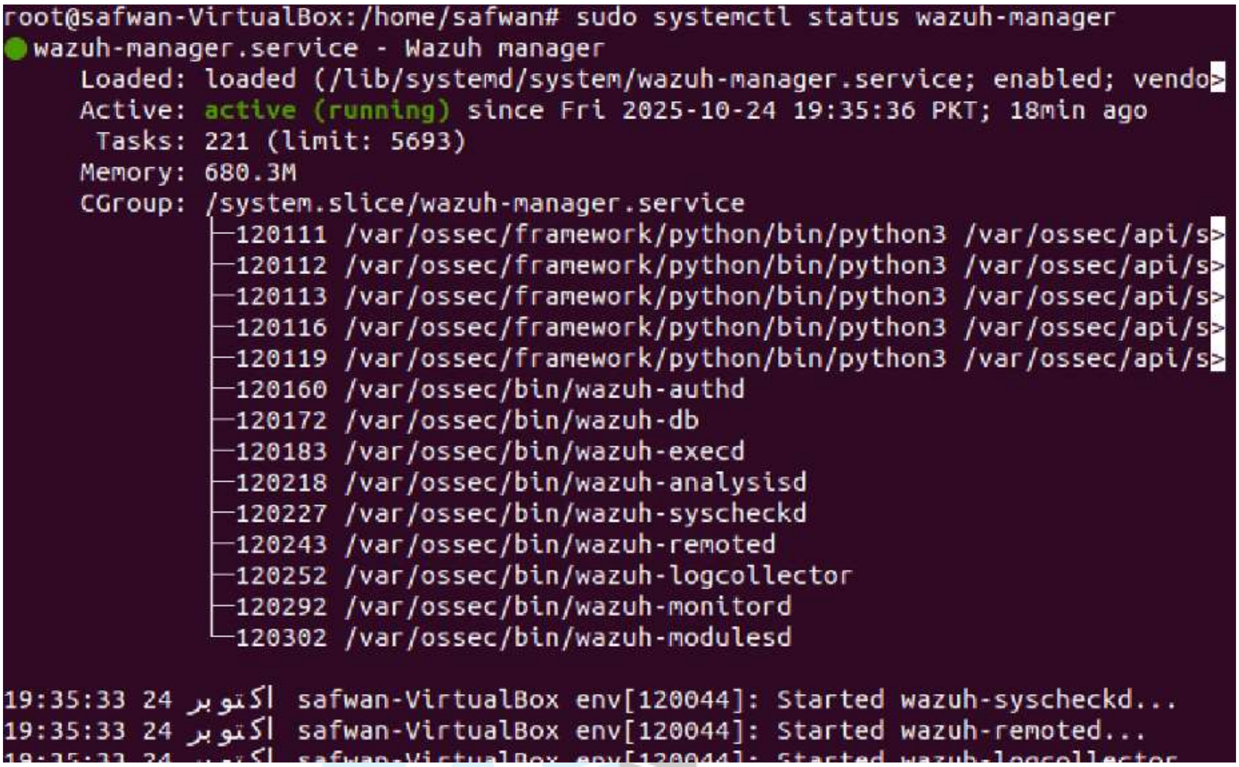


https://192.168.100.153/app/login?

admin

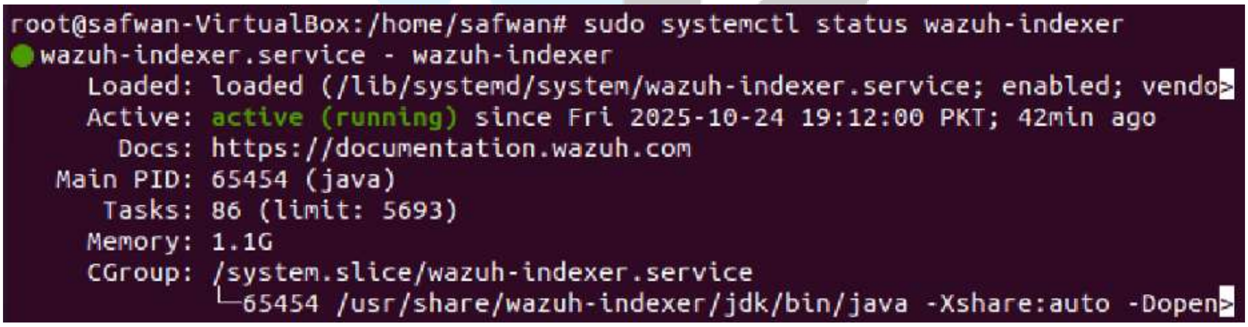
••••••••••••••••••••••••••••••••••

Two terminal screenshots stacked: the top shows sudo systemctl status wazuh-manager with Active: active (running) and multiple Wazuh processes listed (wazuh-manager components).



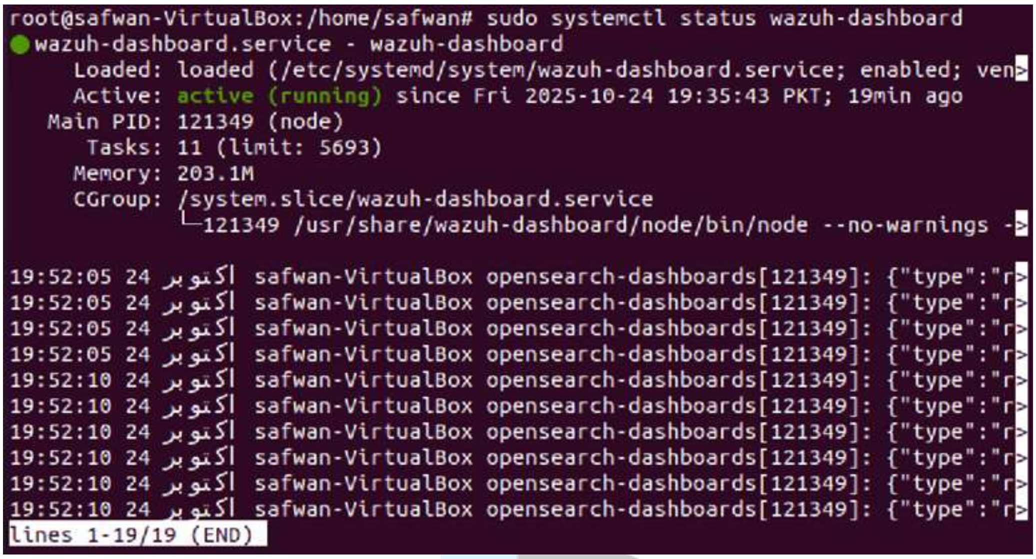
```
root@safwan-VirtualBox:/home/safwan# sudo systemctl status wazuh-manager
● wazuh-manager.service - Wazuh manager
     Loaded: loaded (/lib/systemd/system/wazuh-manager.service; enabled; vendo>
     Active: active (running) since Fri 2025-10-24 19:35:36 PKT; 18min ago
      Tasks: 221 (limit: 5693)
     Memory: 680.3M
     CGroup: /system.slice/wazuh-manager.service
             ├─120111 /var/ossec/framework/python/bin/python3 /var/ossec/api/s>
             ├─120112 /var/ossec/framework/python/bin/python3 /var/ossec/api/s>
             ├─120113 /var/ossec/framework/python/bin/python3 /var/ossec/api/s>
             ├─120116 /var/ossec/framework/python/bin/python3 /var/ossec/api/s>
             ├─120119 /var/ossec/framework/python/bin/python3 /var/ossec/api/s>
             ├─120160 /var/ossec/bin/wazuh-authd
             ├─120172 /var/ossec/bin/wazuh-db
             ├─120183 /var/ossec/bin/wazuh-execd
             ├─120218 /var/ossec/bin/wazuh-analysisd
             ├─120227 /var/ossec/bin/wazuh-syscheckd
             ├─120243 /var/ossec/bin/wazuh-remoted
             ├─120252 /var/ossec/bin/wazuh-logcollector
             ├─120292 /var/ossec/bin/wazuh-monitord
             └─120302 /var/ossec/bin/wazuh-modulesd

19:35:33 24 اکتوبر safwan-VirtualBox env[120044]: Started wazuh-syscheckd...
19:35:33 24 اکتوبر safwan-VirtualBox env[120044]: Started wazuh-remoted...
19:35:33 24 اکتوبر safwan-VirtualBox env[120044]: Started wazuh-logcollector
```

The lower part shows sudo systemctl status wazuh-indexer with Active: active (running) and a Java PID for the indexer. This verifies the Wazuh Manager and Indexer services are running on the host.



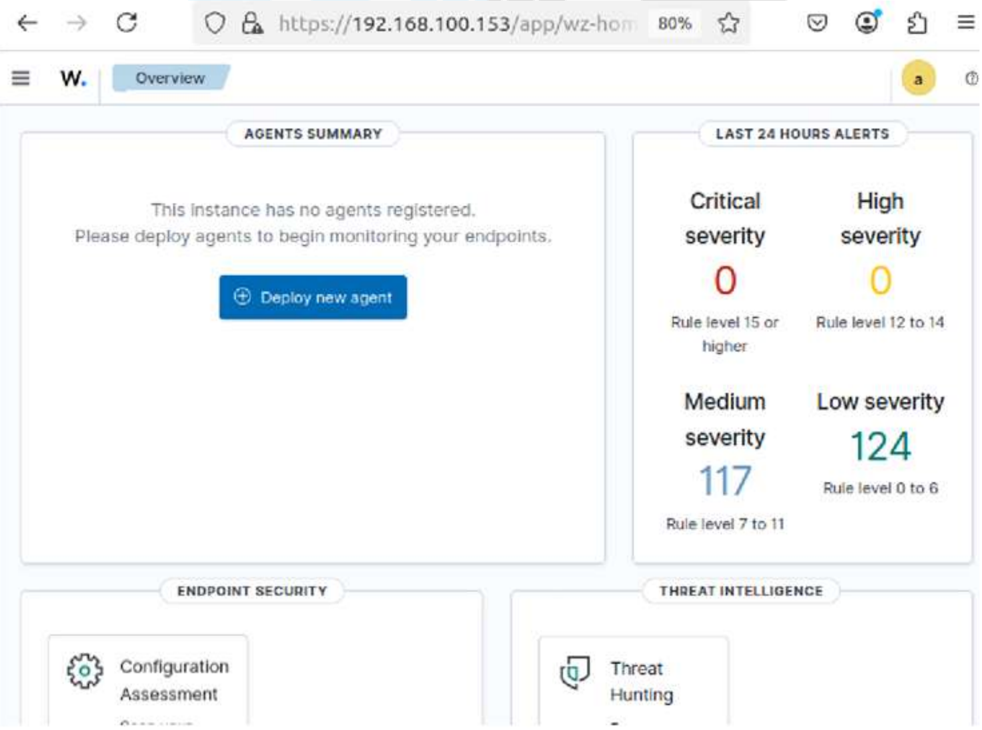
```
root@safwan-VirtualBox:/home/safwan# sudo systemctl status wazuh-indexer
● wazuh-indexer.service - wazuh-indexer
     Loaded: loaded (/lib/systemd/system/wazuh-indexer.service; enabled; vendo>
     Active: active (running) since Fri 2025-10-24 19:12:00 PKT; 42min ago
       Docs: https://documentation.wazuh.com
   Main PID: 65454 (java)
      Tasks: 86 (limit: 5693)
     Memory: 1.1G
     CGroup: /system.slice/wazuh-indexer.service
             └─65454 /usr/share/wazuh-indexer/jdk/bin/java -Xshare:auto -Dopen>
```

This screenshot of the terminal shows sudo systemctl status wazuh-dashboard with Active: active (running) and the dashboard/node process details.



```
root@safwan-VirtualBox:/home/safwan# sudo systemctl status wazuh-dashboard
● wazuh-dashboard.service - wazuh-dashboard
     Loaded: loaded (/etc/systemd/system/wazuh-dashboard.service; enabled; ven>
     Active: active (running) since Fri 2025-10-24 19:35:43 PKT; 19min ago
   Main PID: 121349 (node)
      Tasks: 11 (limit: 5693)
     Memory: 203.1M
     CGroup: /system.slice/wazuh-dashboard.service
             └─121349 /usr/share/wazuh-dashboard/node/bin/node --no-warnings ->

19:52:05 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:05 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:05 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:05 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:10 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:10 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:10 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:10 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:10 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
19:52:10 24 أكتوبر safwan-VirtualBox opensearch-dashboards[121349]: {"type":"r>
lines 1-19/19 (END)
```

Below that is a screenshot of the Wazuh Dashboard overview page showing an “AGENTS SUMMARY” panel that reads **“This instance has no agents registered”** and a prominent Deploy new agent button; the right shows alert counters (Critical 0, High 0, Medium 117, Low 124). Confirms dashboard service is running and accessible, and that no agents were yet registered at the moment of the screenshot.



https://192.168.100.153/app/wz-hom 80%

AGENTS SUMMARY

This instance has no agents registered.
Please deploy agents to begin monitoring your endpoints.

LAST 24 HOURS ALERTS

Critical severity 0 Rule level 15 or higher

High severity 0 Rule level 12 to 14

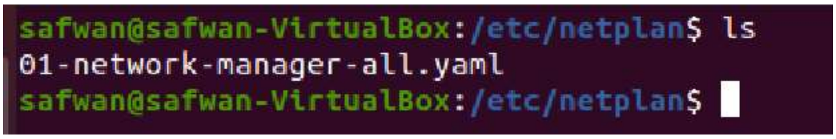
Medium severity 117 Rule level 7 to 11

Low severity 124 Rule level 0 to 6

Lab 02: Assigning Static IP Address to Wazuh Server

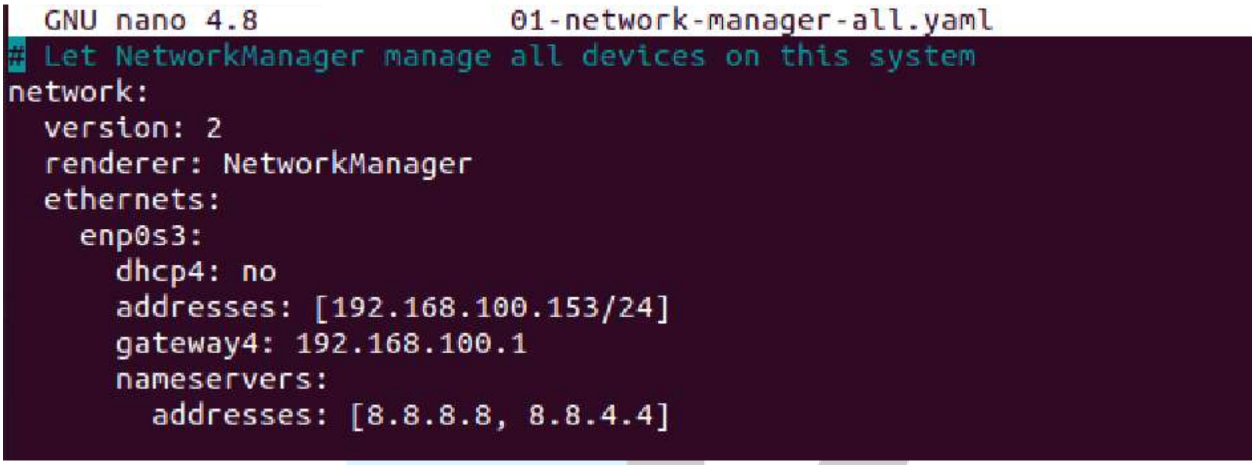
- Assign a static IP to your Wazuh server.
- Confirm connectivity via ping or SSH.
- Ensure IP persistence after system reboot.

This screenshot shows me editing the network configuration file to assign a static IP to my Wazuh server.



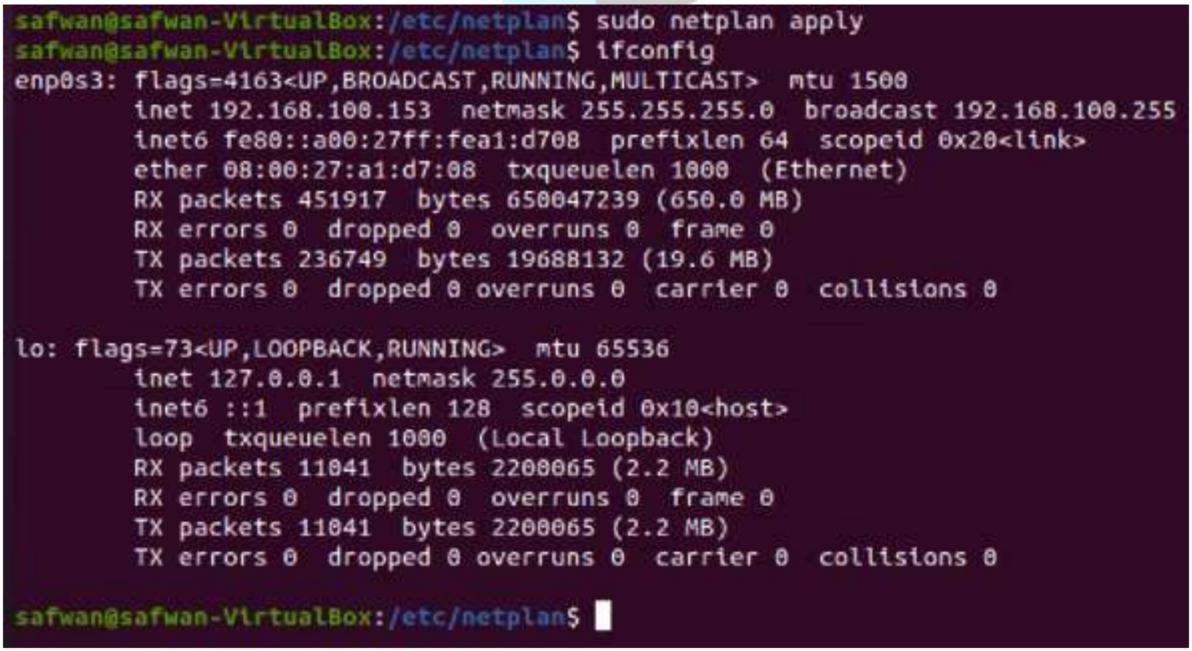
```
safwan@safwan-VirtualBox:/etc/netplan$ ls
01-network-manager-all.yaml
safwan@safwan-VirtualBox:/etc/netplan$ 
```

The Netplan configuration file (.yaml) was edited to assign a static IP address of 192.168.100.153/24 to the enp0s3 interface, setting dhcp4: no and defining the gateway4 and nameservers. This ensures persistence across reboots.



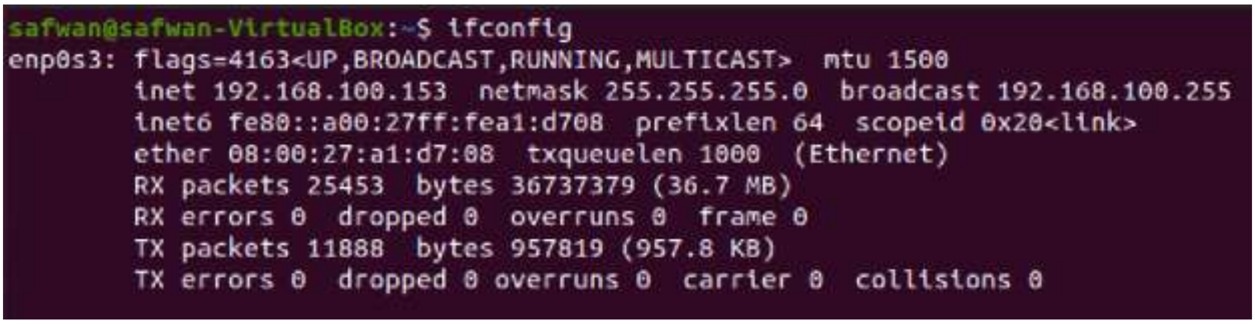
```
  GNU nano 4.8              01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.100.153/24]
      gateway4: 192.168.100.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

The changes were applied using sudo netplan apply. The ifconfig output confirmed the enp0s3 interface successfully received the static IP address (192.168.100.153), completing the assignment and allowing for subsequent connectivity checks (ping/SSH).



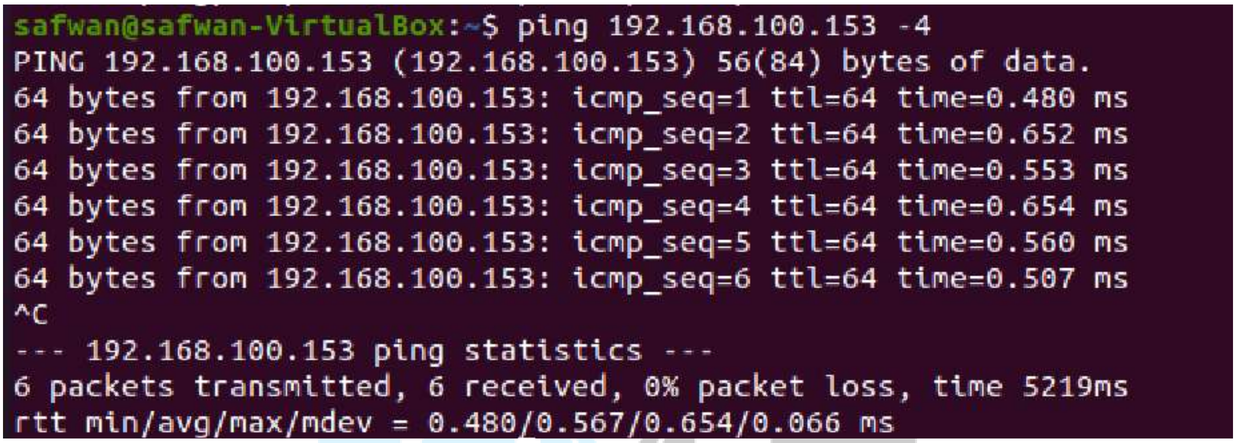
```
safwan@safwan-VirtualBox:/etc/netplan$ sudo netplan apply
safwan@safwan-VirtualBox:/etc/netplan$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.153  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:fea1:d708  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a1:d7:08  txqueuelen 1000  (Ethernet)
        RX packets 451917  bytes 650047239 (650.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 236749  bytes 19688132 (19.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 11041  bytes 2200065 (2.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11041  bytes 2200065 (2.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

safwan@safwan-VirtualBox:/etc/netplan$ 
```

After the above configuration I then reboot the wazuh-manager to check the static IP configuration. This below screenshot demonstrates the activeness of static IP configuration of manager.



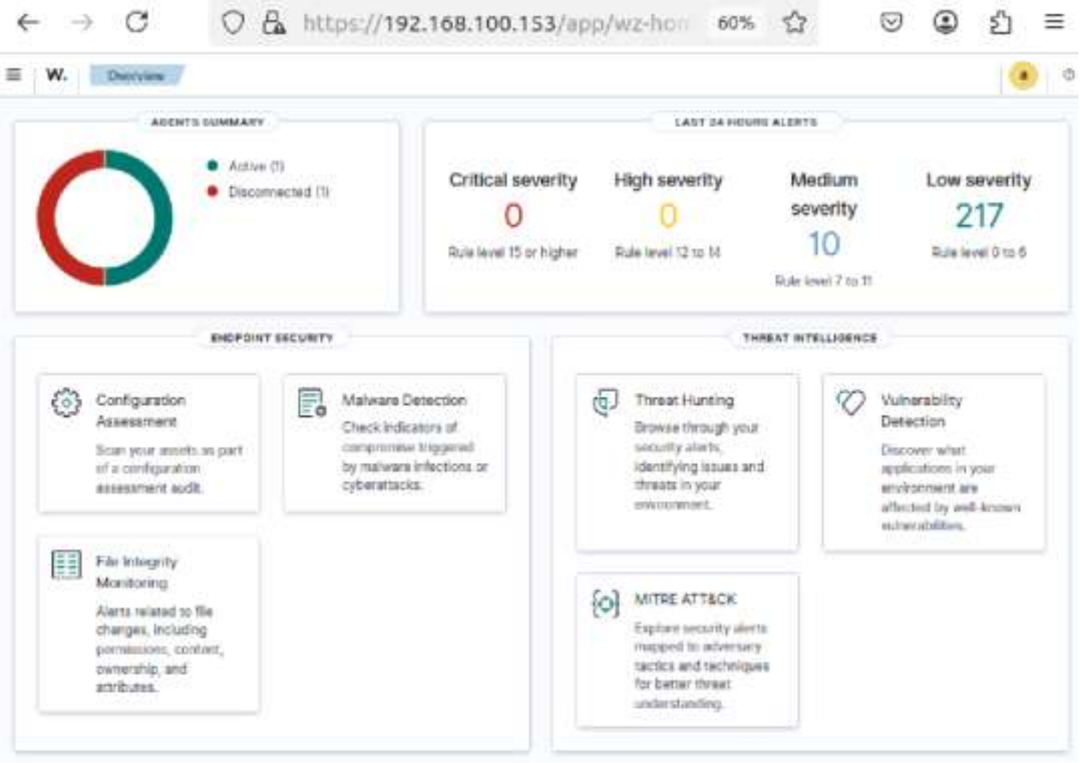
```
safwan@safwan-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.153  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a00:27ff:fea1:d708  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a1:d7:08  txqueuelen 1000  (Ethernet)
        RX packets 25453  bytes 36737379 (36.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11888  bytes 957819 (957.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Here, I tested the connectivity using the ping command to confirm the server was reachable on the assigned IP.



```
safwan@safwan-VirtualBox:~$ ping 192.168.100.153 -4
PING 192.168.100.153 (192.168.100.153) 56(84) bytes of data.
64 bytes from 192.168.100.153: icmp_seq=1 ttl=64 time=0.480 ms
64 bytes from 192.168.100.153: icmp_seq=2 ttl=64 time=0.652 ms
64 bytes from 192.168.100.153: icmp_seq=3 ttl=64 time=0.553 ms
64 bytes from 192.168.100.153: icmp_seq=4 ttl=64 time=0.654 ms
64 bytes from 192.168.100.153: icmp_seq=5 ttl=64 time=0.560 ms
64 bytes from 192.168.100.153: icmp_seq=6 ttl=64 time=0.507 ms
^C
--- 192.168.100.153 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5219ms
rtt min/avg/max/mdev = 0.480/0.567/0.654/0.066 ms
```

After applying the static IP configuration, the final step was to confirm system connectivity and service accessibility. By navigating a web browser to the newly assigned static IP address, 192.168.100.153, the Wazuh Dashboard login page successfully loaded.



AGENTS SUMMARY

- Active (1)
- Disconnected (1)

LAST 24 HOURS ALERTS

Critical severity	High severity	Medium severity	Low severity
0	0	10	217
Rule level 15 or higher	Rule level 12 to 14	Rule level 7 to 11	Rule level 0 to 6

ENDPOINT SECURITY

Configuration Assessment
Scan your assets as part of a configuration assessment audit.

Malware Detection
Check indicators of compromise triggered by malware infections or cyberattacks.

File Integrity Monitoring
Alerts related to file changes, including permissions, content, ownership, and attributes.

THREAT INTELLIGENCE

Threat Hunting
Browse through your security alerts, identifying issues and threats in your environment.

Vulnerability Detection
Discover what applications in your environment are affected by well-known vulnerabilities.

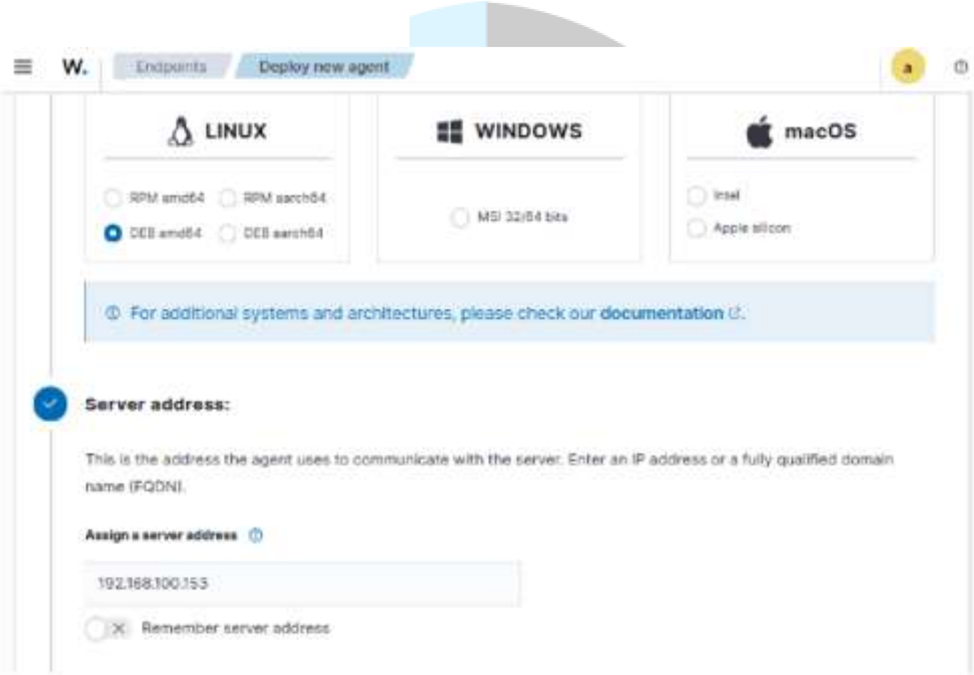
MITRE ATT&CK
Explore security alerts mapped to adversary tactics and techniques for better threat understanding.

Lab 03: Installing Wazuh Agents (Windows & Linux) <u>[Lab 3 and 4 combine]</u>

- Install Wazuh agents on both Windows and Linux endpoints.
- Configure the agents to communicate with the Wazuh server.
- Verify agent status in the Wazuh dashboard.

Shows the Wazuh “Deploy new agent” UI with platform options: Linux, Windows, macOS and radio buttons for package types (DEB/RPM, MSI, etc.).

The UI is clearly set to choose Linux DEB (DEB amd64 selected) as an installation package. This screen is the Dashboard-based registration/installation step and directly corresponds to the Agent Deployment. Displays the Deploy-agent form area titled Server address: with an assigned server address filled in as 192.168.100.153 (Wazuh manager IP) and an option to remember it.



○ RPM amd64 ○ RPM aarch64
● DEB amd64 ○ DEB aarch64

○ MSI 32/64 bits

○ Intel
○ Apple silicon

ⓘ For additional systems and architectures, please check our **documentation**.

Server address:

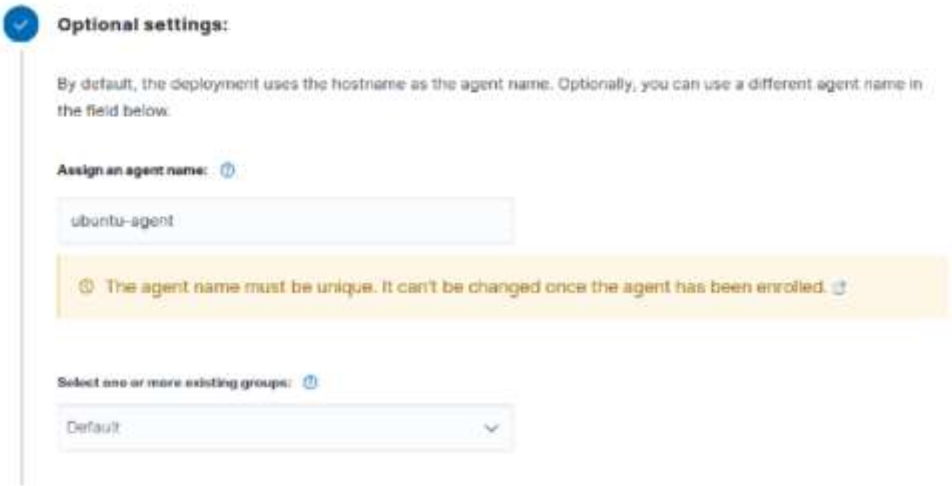
This is the address the agent uses to communicate with the server. Enter an IP address or a fully qualified domain name (FQDN).

Assign a server address ⓘ

192.168.100.153

☒ Remember server address

Below the name provided to the agent.



Optional settings:

By default, the deployment uses the hostname as the agent name. Optionally, you can use a different agent name in the field below.

Assign an agent name:

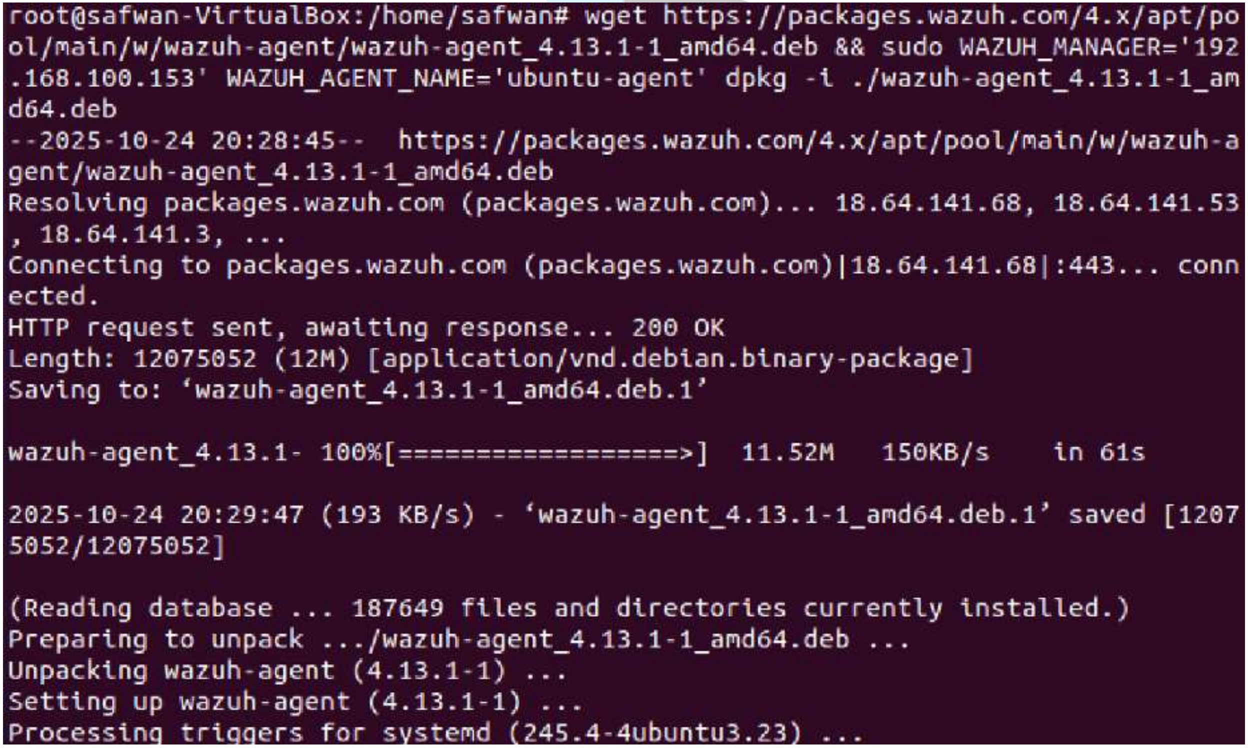
ubuntu-agent

The agent name must be unique. It can't be changed once the agent has been enrolled.

Select one or more existing groups:

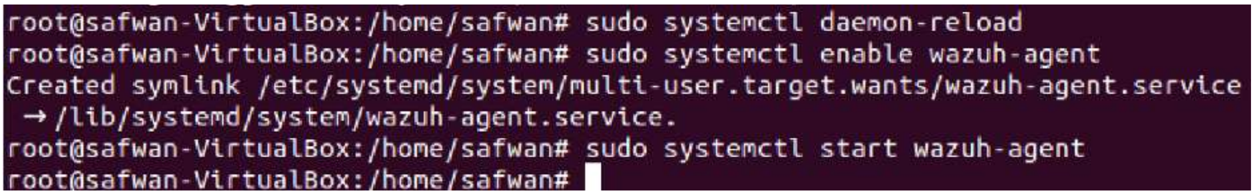
Default

This screenshot displays the installation packages to configure ubuntu-agent successfully. This is the important step in the agent installation process.



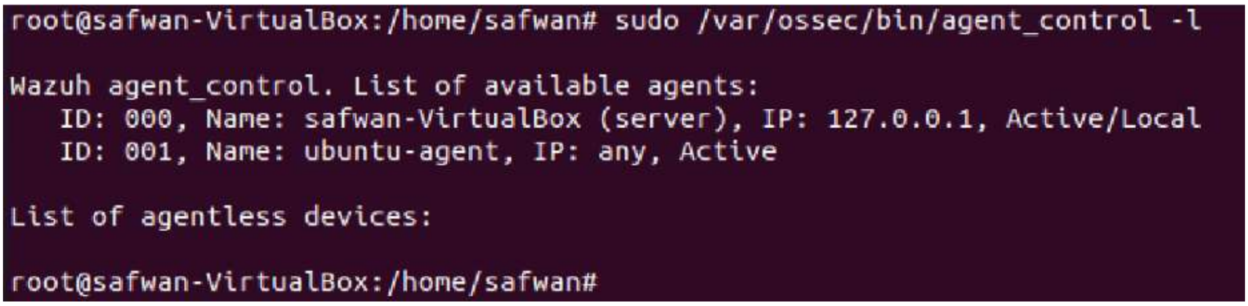
```
root@safwan-VirtualBox:/home/safwan# wget https://packages.wazuh.com/4.x/apt/po
ol/main/w/wazuh-agent/wazuh-agent_4.13.1-1_amd64.deb && sudo WAZUH_MANAGER='192
.168.100.153' WAZUH_AGENT_NAME='ubuntu-agent' dpkg -i ./wazuh-agent_4.13.1-1_am
d64.deb
--2025-10-24 20:28:45--  https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-a
gent/wazuh-agent_4.13.1-1_amd64.deb
Resolving packages.wazuh.com (packages.wazuh.com)... 18.64.141.68, 18.64.141.53
, 18.64.141.3, ...
Connecting to packages.wazuh.com (packages.wazuh.com)|18.64.141.68|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 12075052 (12M) [application/vnd.debian.binary-package]
Saving to: ‘wazuh-agent_4.13.1-1_amd64.deb.1’

wazuh-agent_4.13.1- 100%[===================>]  11.52M   150KB/s    in 61s

2025-10-24 20:29:47 (193 KB/s) - ‘wazuh-agent_4.13.1-1_amd64.deb.1’ saved [1207
5052/12075052]

(Reading database ... 187649 files and directories currently installed.)
Preparing to unpack .../wazuh-agent_4.13.1-1_amd64.deb ...
Unpacking wazuh-agent (4.13.1-1) ...
Setting up wazuh-agent (4.13.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.23) ...
```

That image shows the final steps of installing and configuring the Wazuh Agent on a Linux (Ubuntu) system. These commands ensure the agent starts, runs correctly, and remains active after a reboot.



```
root@safwan-VirtualBox:/home/safwan# sudo systemctl daemon-reload
root@safwan-VirtualBox:/home/safwan# sudo systemctl enable wazuh-agent
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-agent.service
 → /lib/systemd/system/wazuh-agent.service.
root@safwan-VirtualBox:/home/safwan# sudo systemctl start wazuh-agent
root@safwan-VirtualBox:/home/safwan# 
```

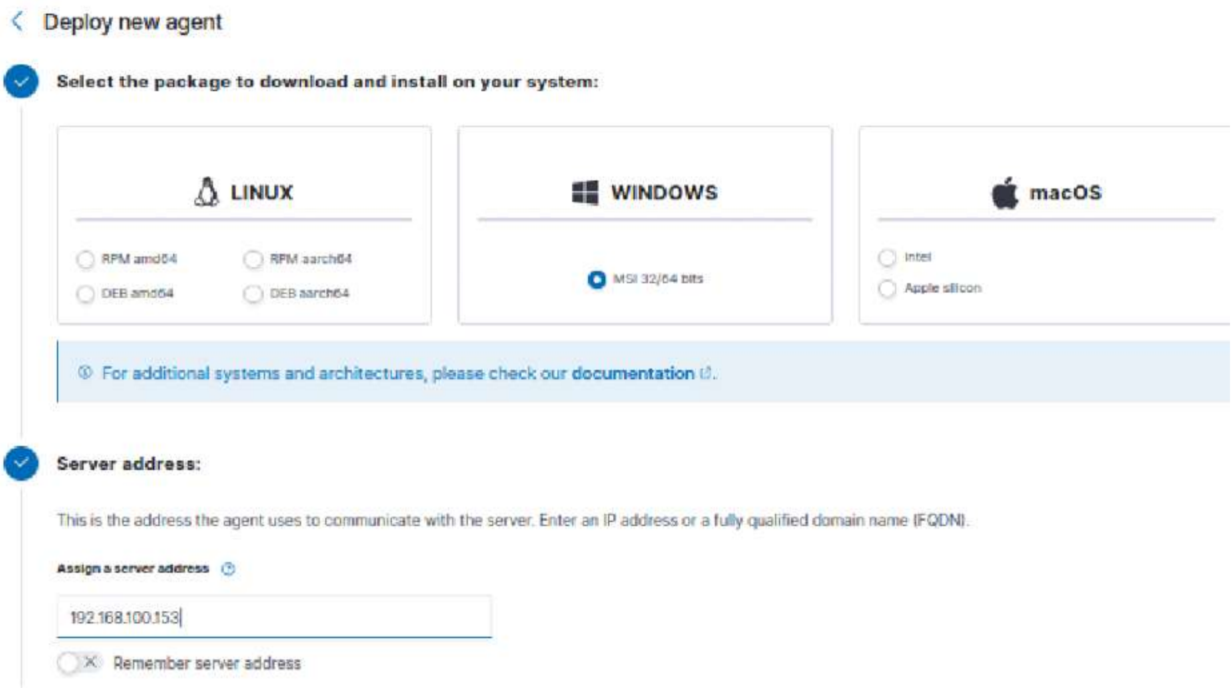
This screenshot shows the successful deployment of wazuh agent name ubuntu-agent, and its not showing IP address because it deployed through the dashboard.



```
root@safwan-VirtualBox:/home/safwan# sudo /var/ossec/bin/agent_control -l

Wazuh agent_control. List of available agents:
   ID: 000, Name: safwan-VirtualBox (server), IP: 127.0.0.1, Active/Local
   ID: 001, Name: ubuntu-agent, IP: any, Active

List of agentless devices:

root@safwan-VirtualBox:/home/safwan#
```

Windows Agent Deployment

The agent package and the Wazuh Server's static IP (192.168.100.153) were selected via the Wazuh Dashboard interface.



Deploy new agent

Select the package to download and install on your system:

LINUX
○ RPM amd64 ○ RPM aarch64
○ DEB amd64 ○ DEB aarch64

WINDOWS
● MSI 32/64 bits

macOS
○ Intel
○ Apple silicon

For additional systems and architectures, please check our documentation.

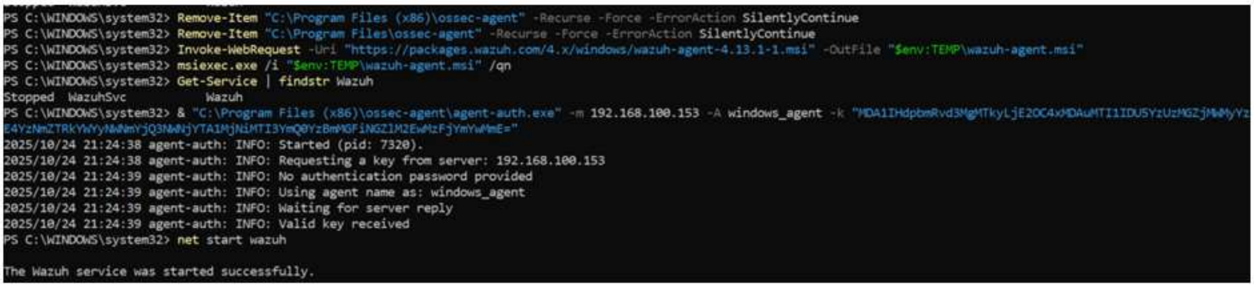
Server address:

This is the address the agent uses to communicate with the server. Enter an IP address or a fully qualified domain name (FQDN).

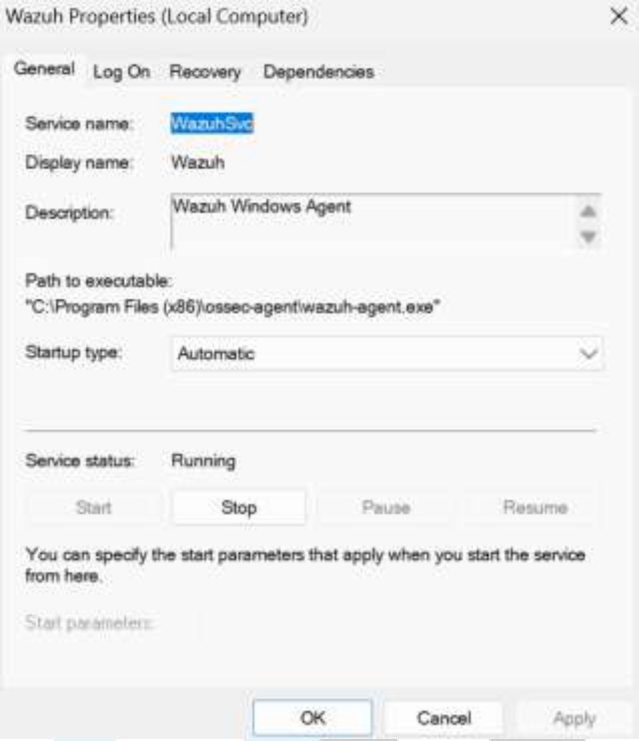
Assign a server address: 192.168.100.153

Remember server address

PowerShell commands were used for a silent installation and secure registration. The critical step was the key exchange, where the agent connected to the server IP and confirmed a "Valid key received," proving successful communication and registration.



```
PS C:\WINDOWS\system32> Remove-Item "C:\Program Files (x86)\ossec-agent" -Recurse -Force -ErrorAction SilentlyContinue
PS C:\WINDOWS\system32> Remove-Item "C:\Program Files\ossec-agent" -Recurse -Force -ErrorAction SilentlyContinue
PS C:\WINDOWS\system32> Invoke-WebRequest -Uri "https://packages.wazuh.com/4.x/windows/wazuh-agent-4.13.1-1.msi" -OutFile "$env:TEMP\wazuh-agent.msi"
PS C:\WINDOWS\system32> msiexec.exe /i "$env:TEMP\wazuh-agent.msi" /qn
PS C:\WINDOWS\system32> Get-Service | findstr Wazuh
Stopped  WazuhSvc           Wazuh
PS C:\WINDOWS\system32> & "C:\Program Files (x86)\ossec-agent\agent-auth.exe" -m 192.168.100.153 -A windows_agent -k "MDA1IHdpbmRvd3MgMTkyLjE2OC4xMDAuMTI1IDU5YzUzMGZjMWMyYz
E4YzNmZTRkYWYyNWNmYjQ3NWNjYTA1MjNiMTI3YmQ0YzBmMGFiNGZlM2EwMzFjYmYwMmE="
2025/10/24 21:24:38 agent-auth: INFO: Started (pid: 7320).
2025/10/24 21:24:38 agent-auth: INFO: Requesting a key from server: 192.168.100.153
2025/10/24 21:24:39 agent-auth: INFO: No authentication password provided
2025/10/24 21:24:39 agent-auth: INFO: Using agent name as: windows_agent
2025/10/24 21:24:39 agent-auth: INFO: Waiting for server reply
2025/10/24 21:24:39 agent-auth: INFO: Valid key received
PS C:\WINDOWS\system32> net start wazuh

The Wazuh service was started successfully.
```



Wazuh Properties (Local Computer)

Service name: WazuhSvc

Display name: Wazuh

Description: Wazuh Windows Agent

Path to executable:
"C:\Program Files (x86)\ossec-agent\wazuh-agent.exe"

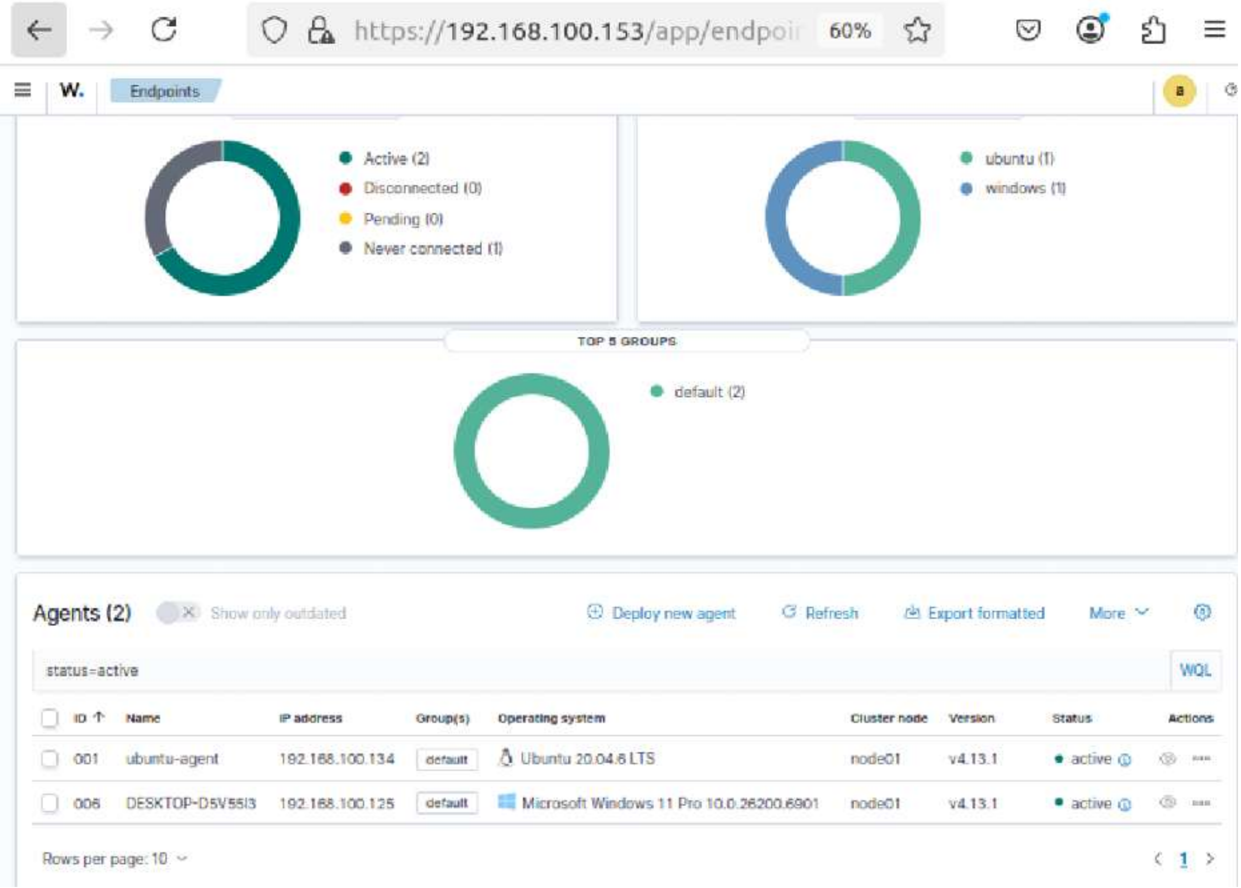
Startup type: Automatic

Service status: Running

You can specify the start parameters that apply when you start the service from here.

Start parameters:

The Windows Service Manager confirmed the WazuhSvc is set to Automatic startup and is currently Running, validating that the agent is persistent and active.



https://192.168.100.153/app/endpoi 60%

- ● Active (2)
- ● Disconnected (0)
- ● Pending (0)
- ● Never connected (1)

- ● ubuntu (1)
- ● windows (1)

TOP 5 GROUPS

- ● default (2)

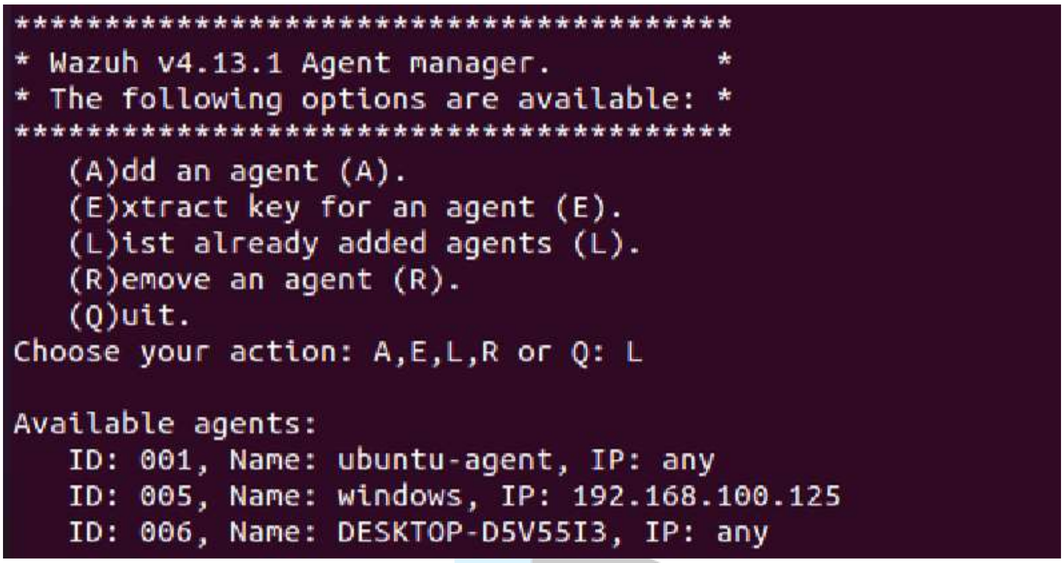
Agents (2)

status=active

☐	ID ↑	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
☐	001	ubuntu-agent	192.168.100.134	default	Ubuntu 20.04.6 LTS	node01	v4.13.1	● active	
☐	006	DESKTOP-D5V55I3	192.168.100.125	default	Microsoft Windows 11 Pro 10.0.26200.6901	node01	v4.13.1	● active	

Rows per page: 10

This below screenshot shows the successful deployment of windows-agent and it shows IP because of the configuration through CLI.

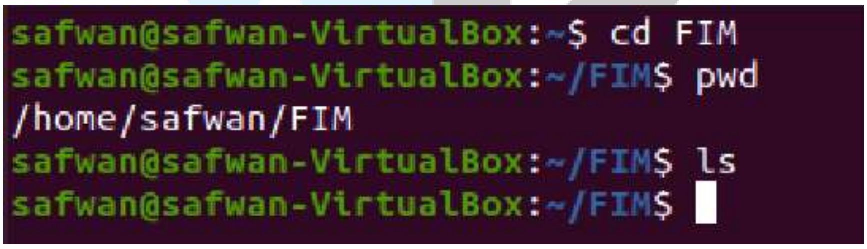


```
****************************************
* Wazuh v4.13.1 Agent manager.         *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: L

Available agents:
   ID: 001, Name: ubuntu-agent, IP: any
   ID: 005, Name: windows, IP: 192.168.100.125
   ID: 006, Name: DESKTOP-D5V55I3, IP: any
```

Lab 05: File Integrity Monitoring

- Enable and configure File Integrity Monitoring (FIM).
- Select directories/files to monitor and make test changes.
- Verify alert generation in Wazuh dashboard.

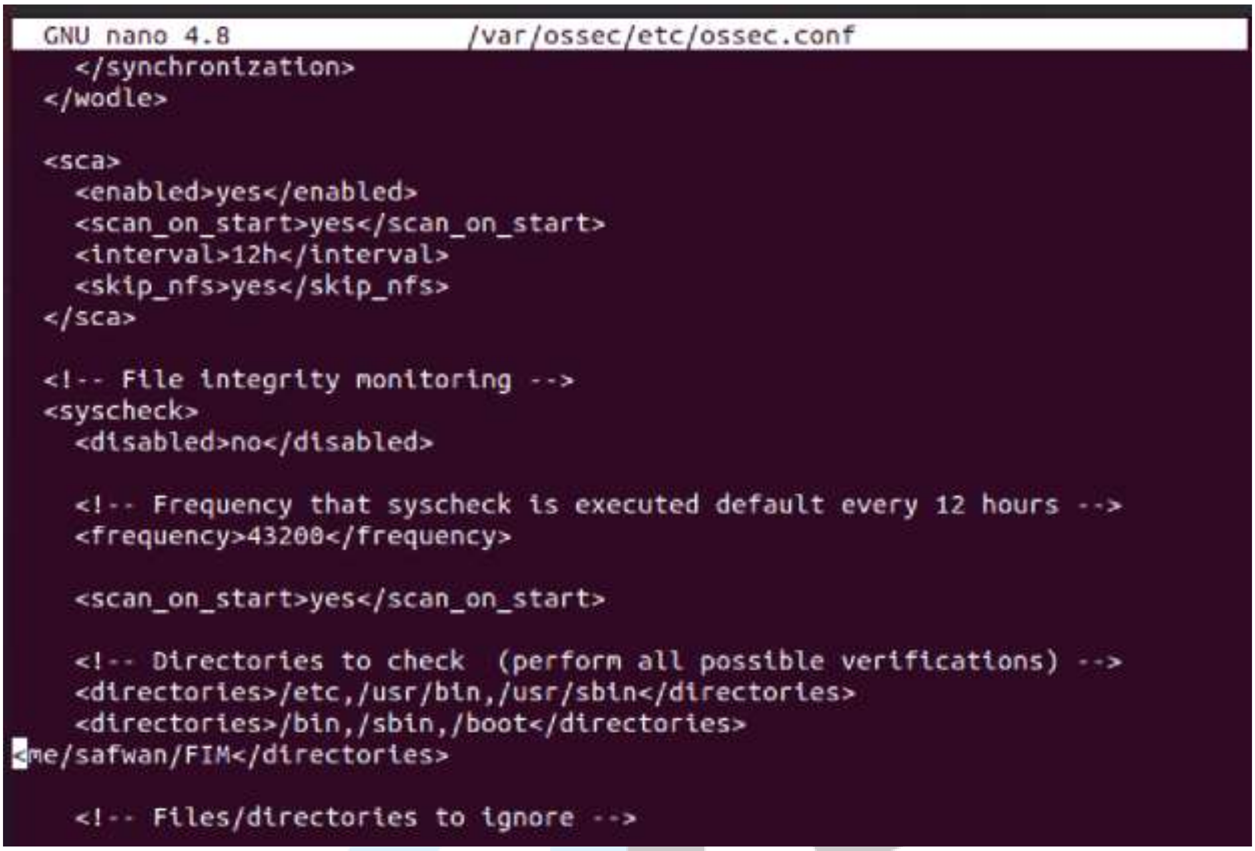
This screenshot shows the user setting up the testing environment by navigating to the designated monitoring directory, /home/safwan/FIM, using the cd and pwd commands. This confirms the exact location where the file changes will be made to test the Wazuh Agent's File Integrity Monitoring (FIM) capabilities.



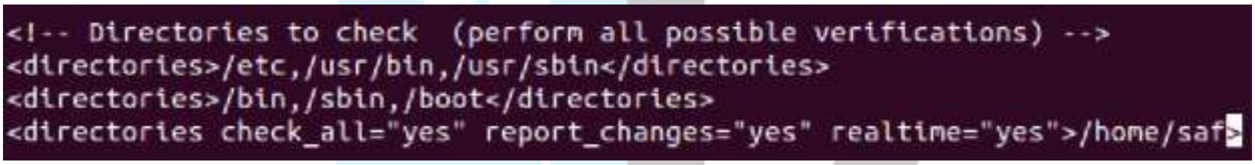
```
safwan@safwan-VirtualBox:~$ cd FIM
safwan@safwan-VirtualBox:~/FIM$ pwd
/home/safwan/FIM
safwan@safwan-VirtualBox:~/FIM$ ls
safwan@safwan-VirtualBox:~/FIM$ 
```

Then by using this command
sudo nano /var/ossec/etc/ossec.conf
I navigated into the ossec.conf to deploy FIM rules.

This image displays the Wazuh Agent configuration snippet where FIM is enabled and customized. It shows the inclusion of the user's home path (/home/Safwan/FIM) with check_all="yes" and realtime="yes". This configuration ensures the agent performs a comprehensive, immediate check for any creation, modification, or deletion within that directory.

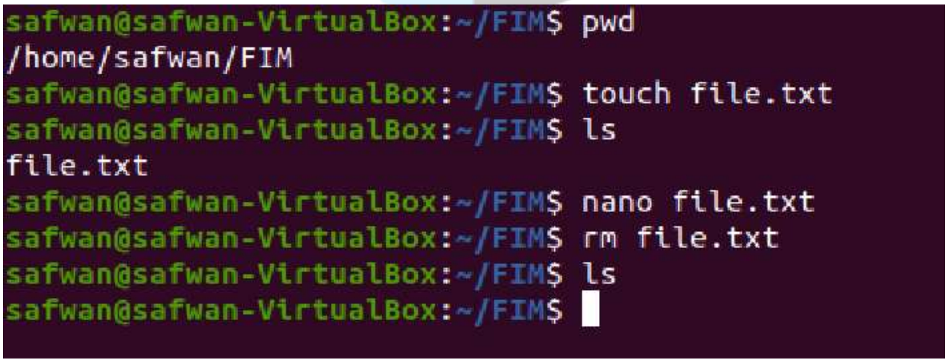


```
  GNU nano 4.8                    /var/ossec/etc/ossec.conf
    </synchronization>
  </wodle>

  <sca>
    <enabled>yes</enabled>
    <scan_on_start>yes</scan_on_start>
    <interval>12h</interval>
    <skip_nfs>yes</skip_nfs>
  </sca>

  <!-- File integrity monitoring -->
  <syscheck>
    <disabled>no</disabled>

    <!-- Frequency that syscheck is executed default every 12 hours -->
    <frequency>43200</frequency>

    <scan_on_start>yes</scan_on_start>

    <!-- Directories to check  (perform all possible verifications) -->
    <directories>/etc,/usr/bin,/usr/sbin</directories>
    <directories>/bin,/sbin,/boot</directories>
<me/safwan/FIM</directories>

    <!-- Files/directories to ignore -->
```



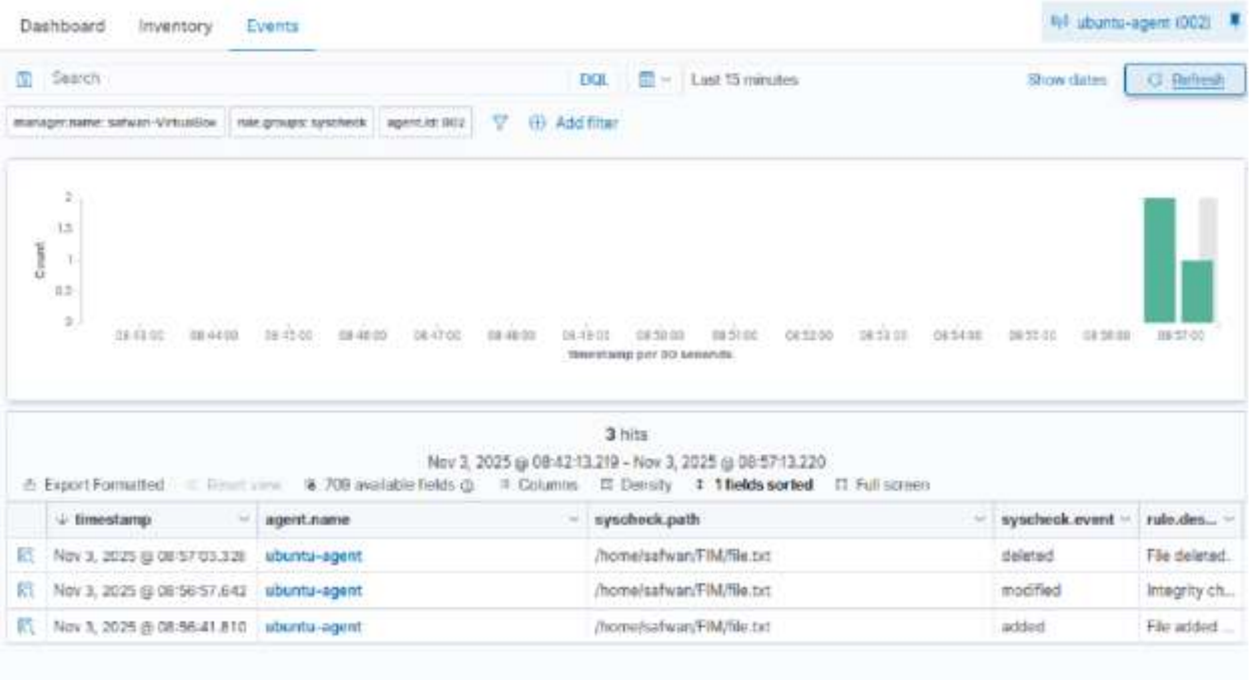
```
<!-- Directories to check  (perform all possible verifications) -->
<directories>/etc,/usr/bin,/usr/sbin</directories>
<directories>/bin,/sbin,/boot</directories>
<directories check_all="yes" report_changes="yes" realtime="yes">/home/saf>
```

This screenshot documents the sequence of events used to validate FIM: a file was created (touch file.txt), then modified (nano file.txt), and finally deleted (rm file.txt) within the monitored directory. Each of these actions serves as a distinct test case that should trigger an immediate security alert on the Wazuh Manager, confirming the FIM setup is working.



```
safwan@safwan-VirtualBox:~/FIM$ pwd
/home/safwan/FIM
safwan@safwan-VirtualBox:~/FIM$ touch file.txt
safwan@safwan-VirtualBox:~/FIM$ ls
file.txt
safwan@safwan-VirtualBox:~/FIM$ nano file.txt
safwan@safwan-VirtualBox:~/FIM$ rm file.txt
safwan@safwan-VirtualBox:~/FIM$ ls
safwan@safwan-VirtualBox:~/FIM$ 
```

The test sequence (create, modify, delete) within the monitored directory confirmed File Integrity Monitoring (FIM) is fully functional. The Wazuh Dashboard immediately generated distinct alerts for the file added, modified, and the file deleted. This validates the customized FIM settings and confirms the agent is correctly reporting all changes in real-time.

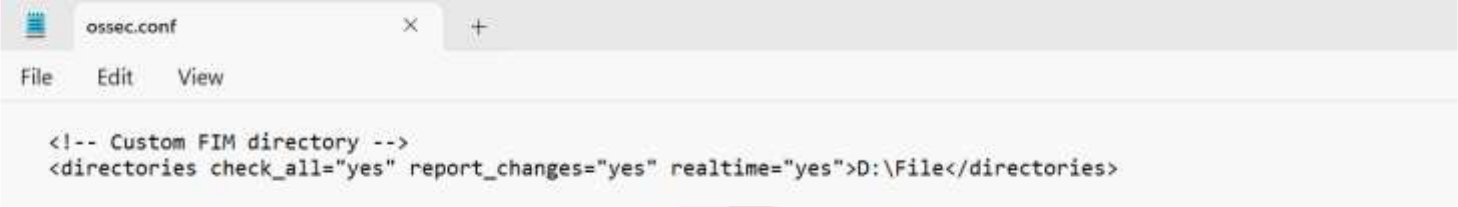


3 hits

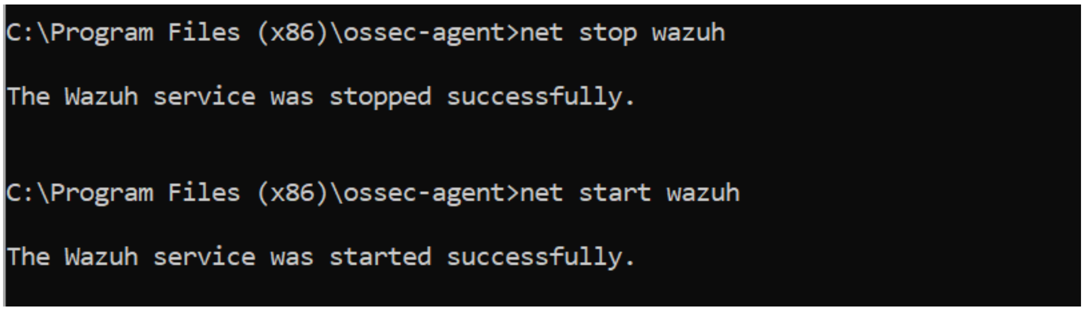
Nov 3, 2025 @ 08:42:13.219 - Nov 3, 2025 @ 08:57:13.220

timestamp	agent.name	syscheck.path	syscheck.event	rule.des...
Nov 3, 2025 @ 08:57:03.328	ubuntu-agent	/home/safwan/FIM/file.txt	deleted	File deleted...
Nov 3, 2025 @ 08:56:57.642	ubuntu-agent	/home/safwan/FIM/file.txt	modified	Integrity ch...
Nov 3, 2025 @ 08:56:41.810	ubuntu-agent	/home/safwan/FIM/file.txt	added	File added ...

This screenshot shows the key modification made to the Windows Agent's configuration file. A custom <directories> entry was added specifically targeting the path D:\File. Crucially, the attributes check_all="yes" and realtime="yes" were set, instructing the agent to perform comprehensive, immediate monitoring of this directory for File Integrity Monitoring (FIM).

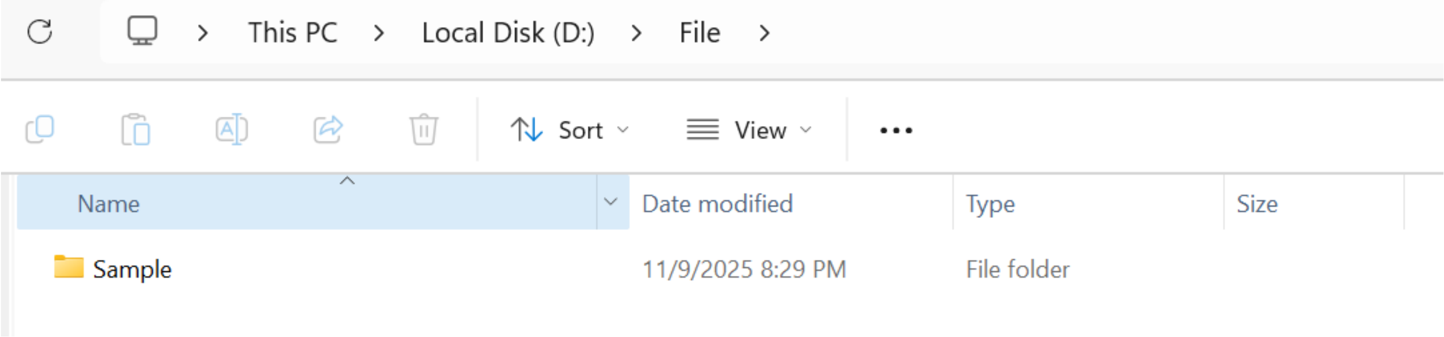


```
<!-- Custom FIM directory -->
<directories check_all="yes" report_changes="yes" realtime="yes">D:\File</directories>
```



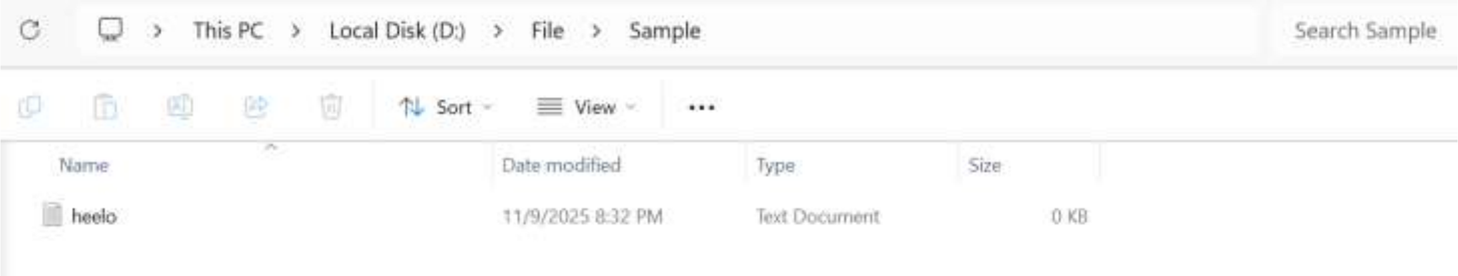
```
C:\Program Files (x86)\ossec-agent>net stop wazuh

The Wazuh service was stopped successfully.


C:\Program Files (x86)\ossec-agent>net start wazuh

The Wazuh service was started successfully.
```

This screenshot confirms the setup of the testing location on the Windows machine. The Sample directory was created within the custom monitoring path, D:\File.



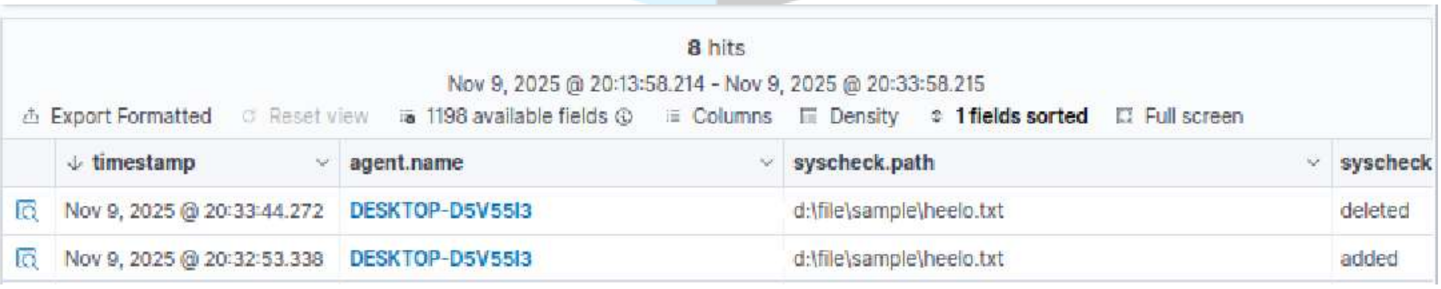
Name	Date modified	Type	Size
Sample	11/9/2025 8:29 PM	File folder	

This image shows the initial FIM test being executed. A new file named heelo.txt was created inside the monitored D:\File\Sample path. This action is designed to trigger the first alert—the "file added" event—confirming the agent's real-time detection capabilities are fully active on the Windows endpoint.



Name	Date modified	Type	Size
heelo	11/9/2025 8:32 PM	Text Document	0 KB

This screenshot provides the final, essential proof of FIM functionality. The Wazuh Dashboard successfully displays syscheck events for the Windows agent. The logs clearly show the sequence of detection: an event for the file being added and a subsequent event for the file being deleted. This confirms that the customized Windows FIM is correctly operating and reporting all changes in real-time


8 hits

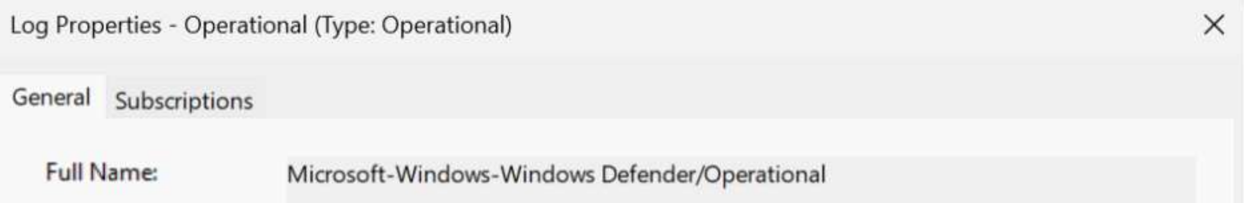
Nov 9, 2025 @ 20:13:58.214 - Nov 9, 2025 @ 20:33:58.215

timestamp	agent.name	syscheck.path	syscheck
Nov 9, 2025 @ 20:33:44.272	DESKTOP-D5V55I3	d:\file\sample\heelo.txt	deleted
Nov 9, 2025 @ 20:32:53.338	DESKTOP-D5V55I3	d:\file\sample\heelo.txt	added

Lab 06: Windows Defender Logs Integration

- Enable Defender log collection via Wazuh agents.
- Simulate a malware detection (EICAR test file).
- Confirm logs are visible in the Wazuh dashboard.

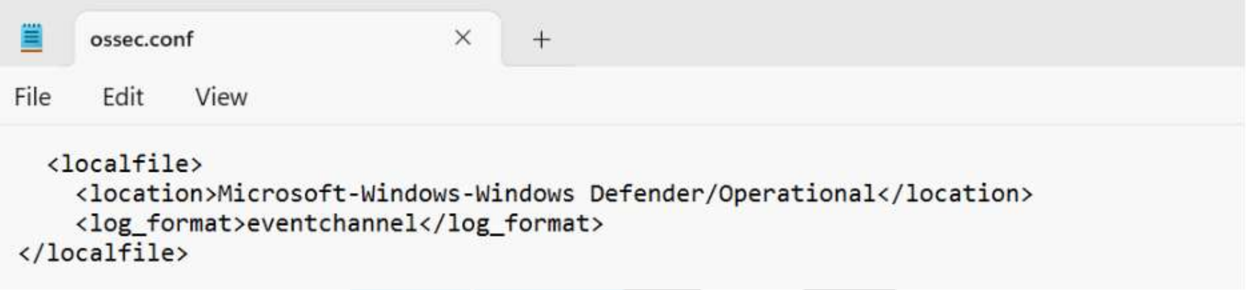
This image confirms the full system name of the event channel being targeted for log collection. It validates that the correct log source is Microsoft-Windows-Windows Defender/Operational.



Log Properties - Operational (Type: Operational)

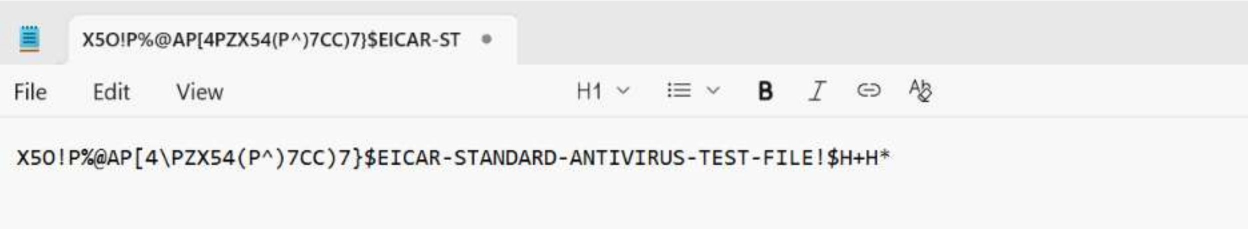
Full Name: Microsoft-Windows-Windows Defender/Operational

This screenshot shows the necessary addition to the Windows Agent's configuration file (ossec.conf). The <localfile> block is added to specify that the agent should monitor the Windows Event Channel located at Microsoft-Windows-Windows Defender/Operational. This crucial step enables the agent to read and forward all antivirus and threat detection logs generated by Windows Defender.



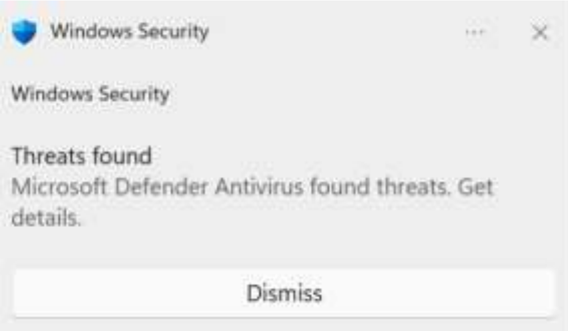
```
  <localfile>
    <location>Microsoft-Windows-Windows Defender/Operational</location>
    <log_format>eventchannel</log_format>
</localfile>
```

This screenshot shows the creation of the standard **EICAR-STANDARD-ANTIVIRUS-TEST-FILE**. This is not actual malware but a globally recognized, harmless string designed to simulate a virus detection event. The act of creating or saving this file immediately triggers Windows Defender, generating the log event we intend to capture. The EICAR file save as **file.com**.



```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

This image captures the immediate response from the endpoint's security software. The Windows Security notification confirms that "Threats found" have been detected by Microsoft Defender Antivirus. This visible system alert verifies that the simulation (the EICAR file) was successful and that a detection event has been recorded in the Windows Defender operational log.

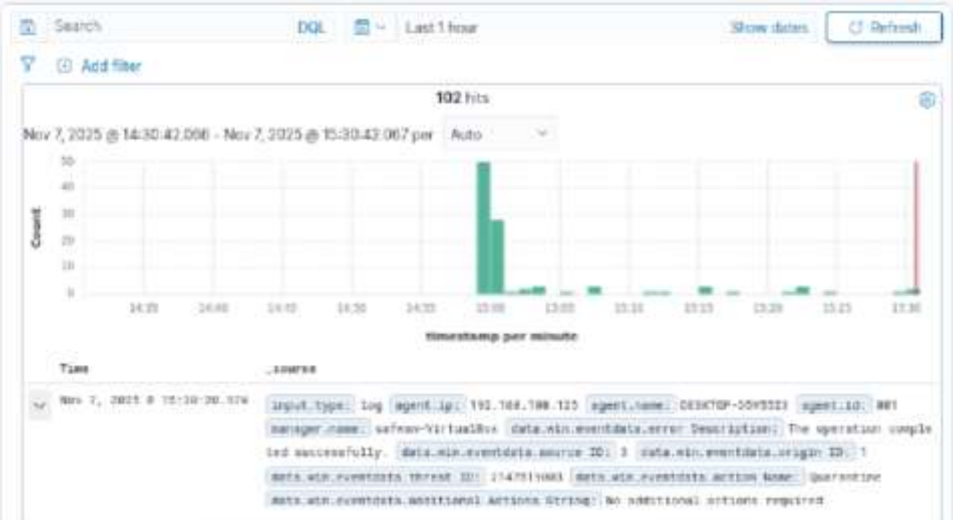


Windows Security

Windows Security

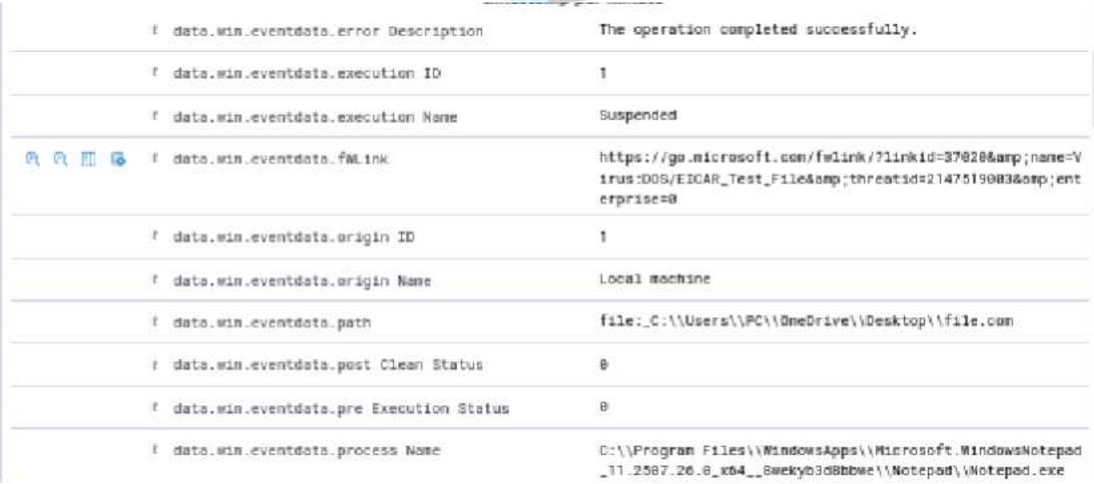
Threats found
Microsoft Defender Antivirus found threats. Get details.

This screenshot from the Wazuh Dashboard confirms that the detection was successfully forwarded to the Manager. The chart shows a clear spike in events around 15:00, correlating with the simulated attack. The log entry confirms the source is the Windows agent (DESKTOP-D5V55I3) and provides critical details, including the successful Quarantine action taken by Defender.



```
Nov 7, 2025 @ 15:30:20.376  input.type: log agent.ip: 192.168.100.125 agent.name: DESKTOP-D5V55I3 agent.id: 001
                            manager.name: safeov-VirtualBox data.win.eventdata.error Description: The operation comple
                            ted successfully. data.win.eventdata.source ID: 3 data.win.eventdata.origin ID: 1
                            data.win.eventdata.threat ID: 2147519003 data.win.eventdata.action Name: Quarantine
                            data.win.eventdata.additional Actions String: No additional actions required
```

This image provides a deep dive into the raw log data captured by Wazuh. Key fields confirm the nature of the event, showing that the execution Name was Suspended, the log includes a reference to the **file.com**, and the operation Description states "The operation completed successfully." This final verification confirms the full integration of Windows Defender logs into the Wazuh security platform

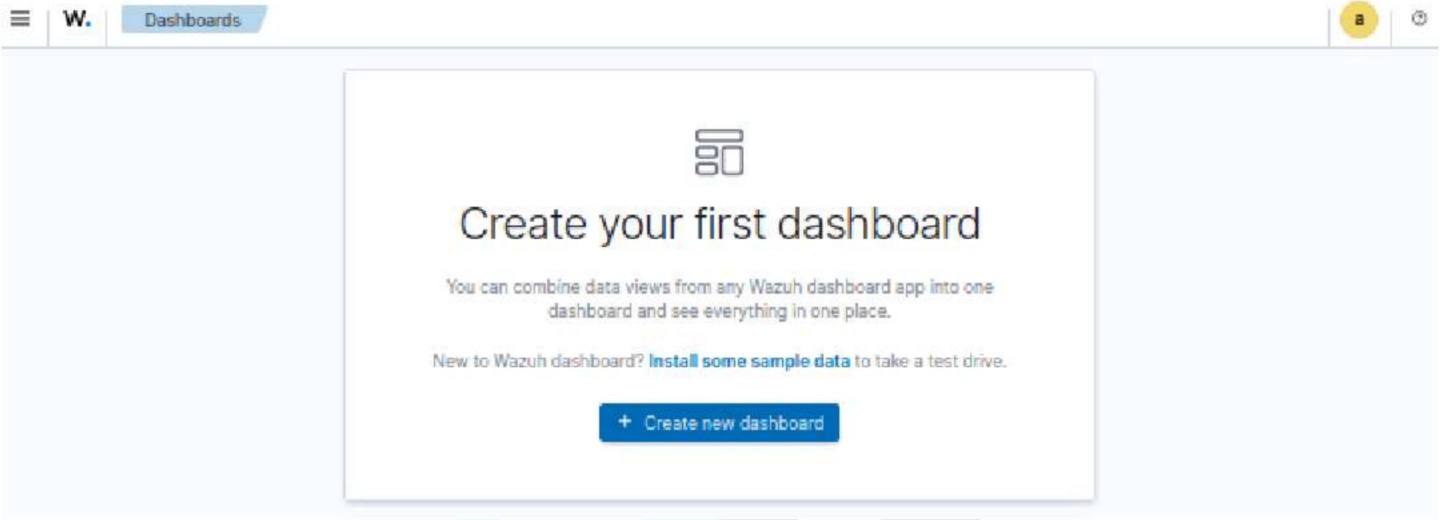


t data.win.eventdata.error Description	The operation completed successfully.
t data.win.eventdata.execution ID	1
t data.win.eventdata.execution Name	Suspended
t data.win.eventdata.fwLink	https://go.microsoft.com/fwlink/?linkid=37020&amp;name=Virus:DOS/EICAR_Test_File&amp;threatid=2147519003&amp;enterprise=0
t data.win.eventdata.origin ID	1
t data.win.eventdata.origin Name	Local machine
t data.win.eventdata.path	file:_C:\\Users\\PC\\OneDrive\\Desktop\\file.com
t data.win.eventdata.post Clean Status	0
t data.win.eventdata.pre Execution Status	0
t data.win.eventdata.process Name	C:\\Program Files\\WindowsApps\\Microsoft.WindowsNotepad_11.2507.26.0_x64__8wekyb3d8bbwe\\Notepad\\Notepad.exe

Lab 07: Basic Dashboard Navigation and Configuration

- Explore the Wazuh Dashboard and view prebuilt dashboards.
- Create a custom visualization.

This screenshot shows the initial landing page after accessing the Wazuh Dashboard. It indicates that no custom dashboards have been created yet, presenting the option to either "Create your first dashboard" or "Install some sample data" to begin visualizing the collected security information

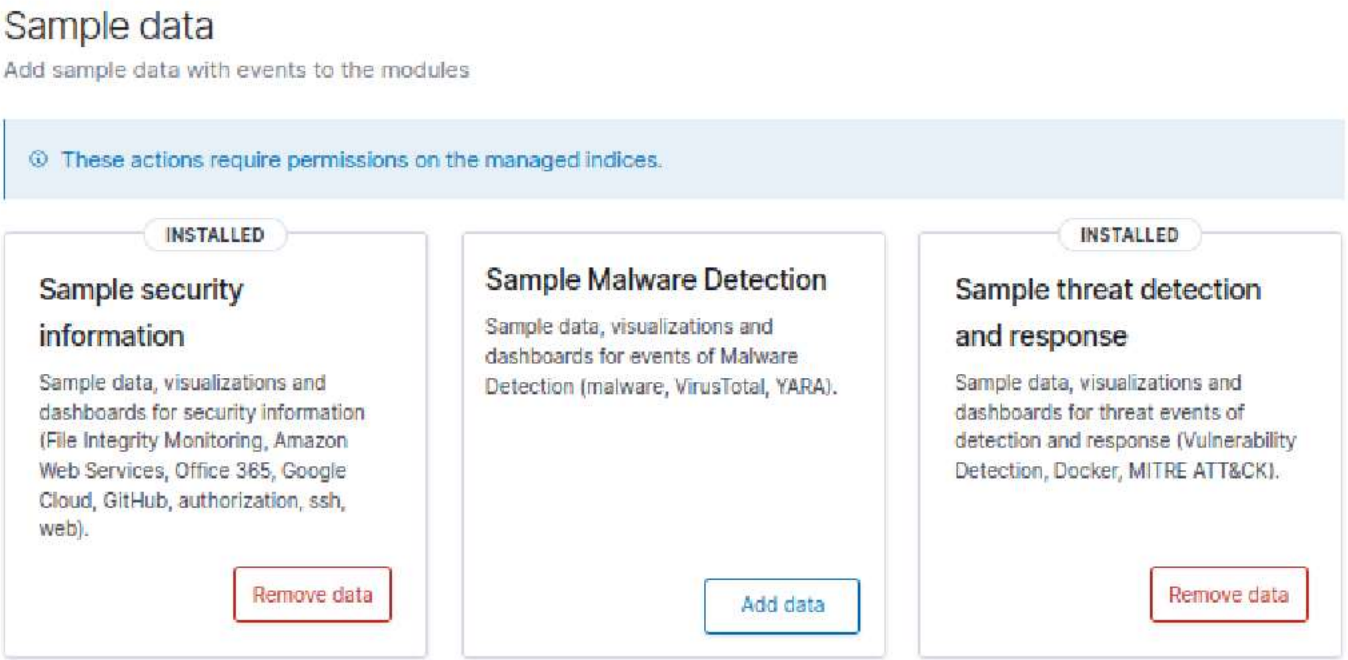


Create your first dashboard

You can combine data views from any Wazuh dashboard app into one dashboard and see everything in one place.

New to Wazuh dashboard? Install some sample data to take a test drive.

This image displays the menu for installing sample data into the Wazuh modules (security information, malware detection, threat detection). This optional step allows new users to immediately populate their dashboards with events for testing and learning, providing instant visualizations even without active agents generating logs.



Sample data

Add sample data with events to the modules

These actions require permissions on the managed indices.

Sample security information

Sample data, visualizations and dashboards for security information (File Integrity Monitoring, Amazon Web Services, Office 365, Google Cloud, GitHub, authorization, ssh, web).

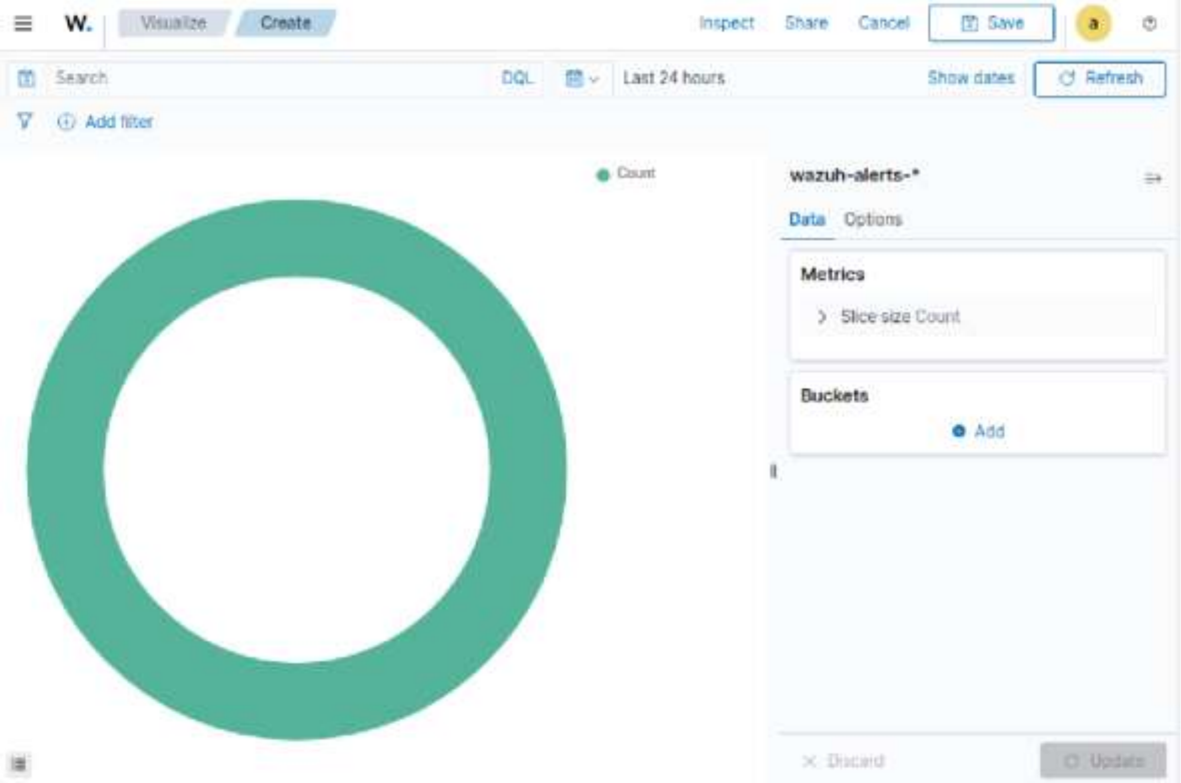
Sample Malware Detection

Sample data, visualizations and dashboards for events of Malware Detection (malware, VirusTotal, YARA).

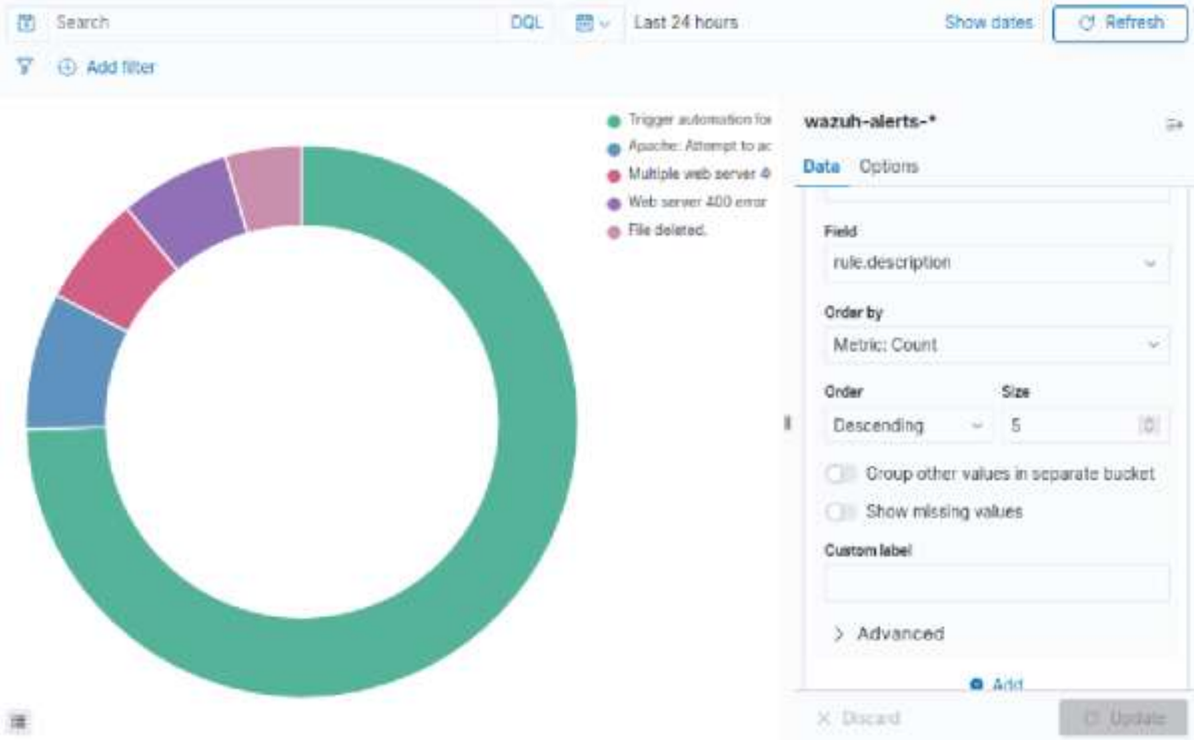
Sample threat detection and response

Sample data, visualizations and dashboards for threat events of detection and response (Vulnerability Detection, Docker, MITRE ATT&CK).

This screenshot captures the beginning of the process for creating a custom visualization. The user is in the "Create" mode, preparing to define a new chart or graph. Currently, it shows the default view for a blank visualization, ready for the user to select the metrics and buckets for the security data they wish to display.

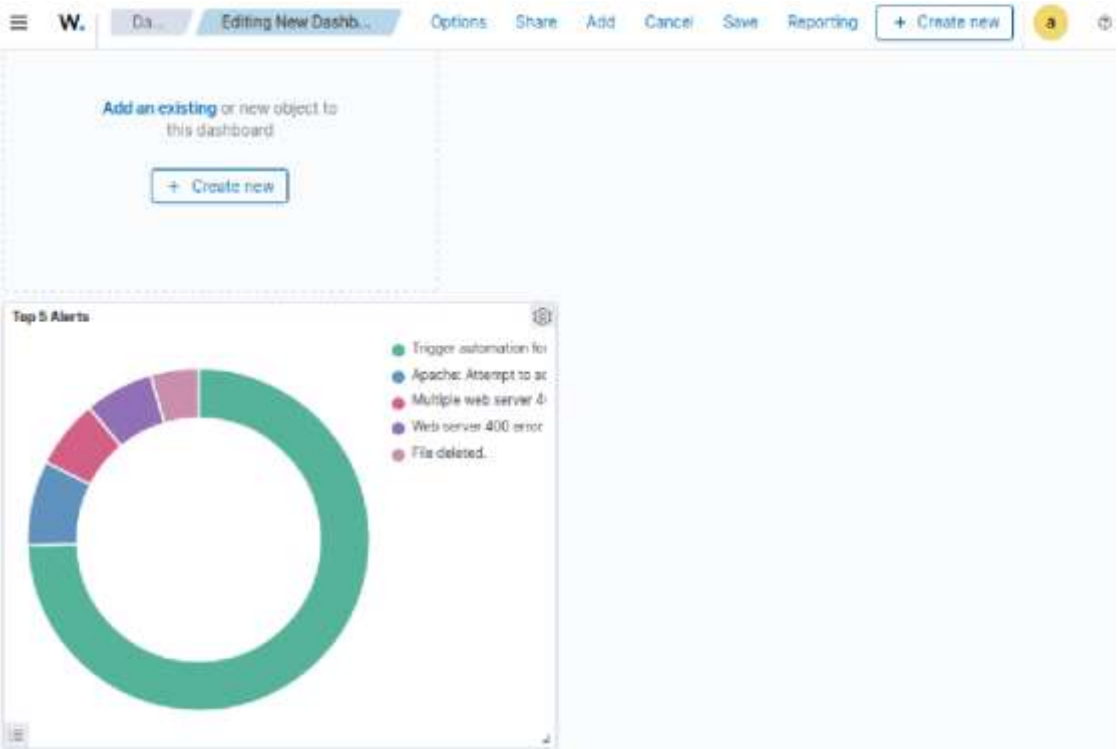


This image shows the custom visualization being actively configured, likely as a Pie Chart. The chart is being built from the wazuh-alerts-* index, and the aggregation is being defined by the rule.description field. This setup is designed to show the distribution and count of the most frequent types of security alerts captured by Wazuh.



Field: rule.description
Order by: Metric: Count
Order: Descending
Size: 5

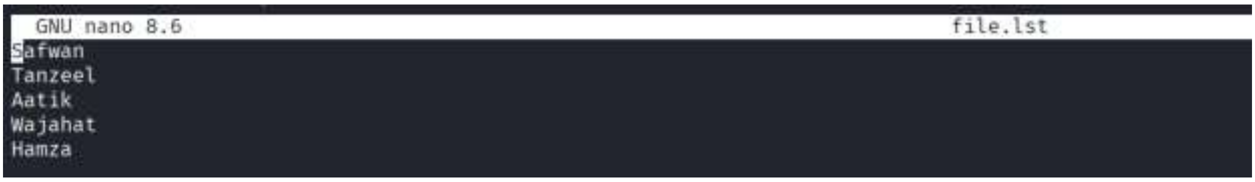
This final screenshot confirms the successful creation and placement of the custom visualization. The newly created chart, titled **"Top 5 Alerts"** is visible on the "Editing New Dashboard" page. This demonstrates the completion of the lab's objective to build a custom visualization and add it to a personal dashboard for quick, focused monitoring.



Lab 08: SSH Brute Force Detection and Basic Response

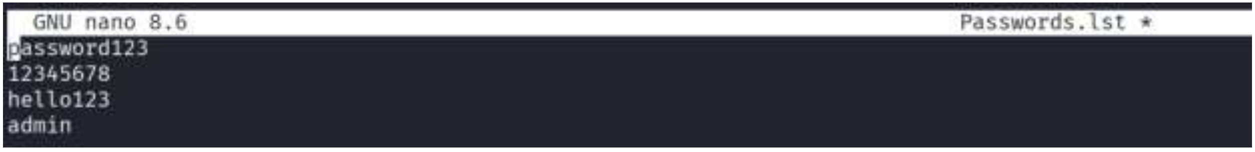
- Simulate SSH brute force attacks on a Linux system.
- Observe alerts generated by Wazuh and configure active response.
- Provide proof of IP blocking after repeated failed logins.

This image displays the file.lst file being edited with nano. This list contains several usernames (Safwan, Tanzeel, Aatik, etc.) which will be used by the brute force tool (Medusa) to simulate an attacker guessing different account names.



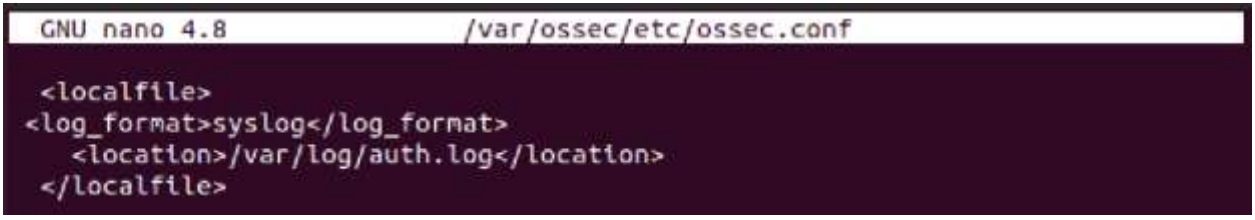
```
  GNU nano 8.6                                              file.lst
Safwan
Tanzeel
Aatik
Wajahat
Hamza
```

This screenshot shows the Passwords.lst file, which contains a short list of common passwords (password123, 12345678, hello123, admin). This list will be combined with the user list to conduct the dictionary attack against the target system



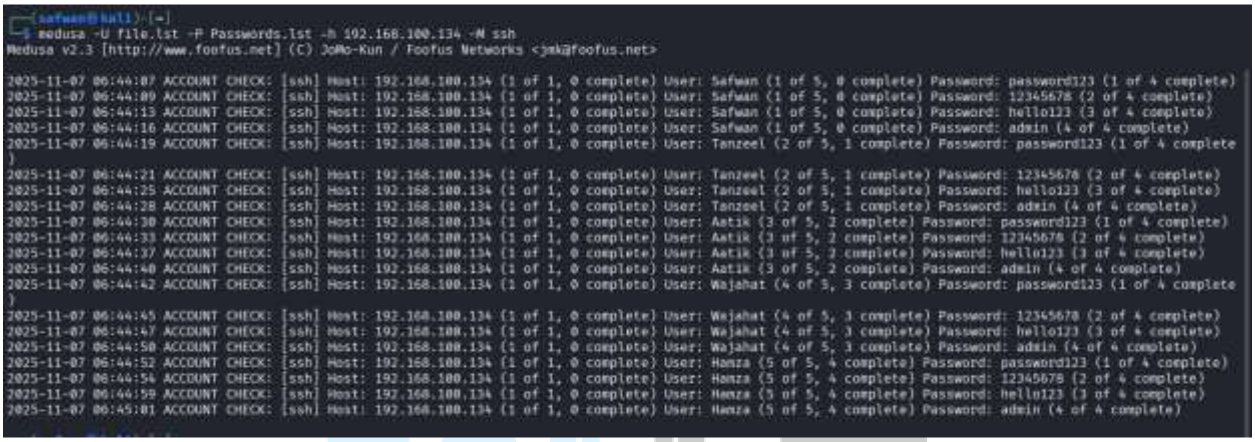
```
  GNU nano 8.6                                              Passwords.lst *
password123
12345678
hello123
admin
```

This screenshot shows the Wazuh Manager configuration (ossec.conf) for log collection. The <localfile> entry is configured to read the /var/log/auth.log file, which is essential because it contains all Linux login attempts, including the failed ones necessary for detecting SSH brute force attacks.



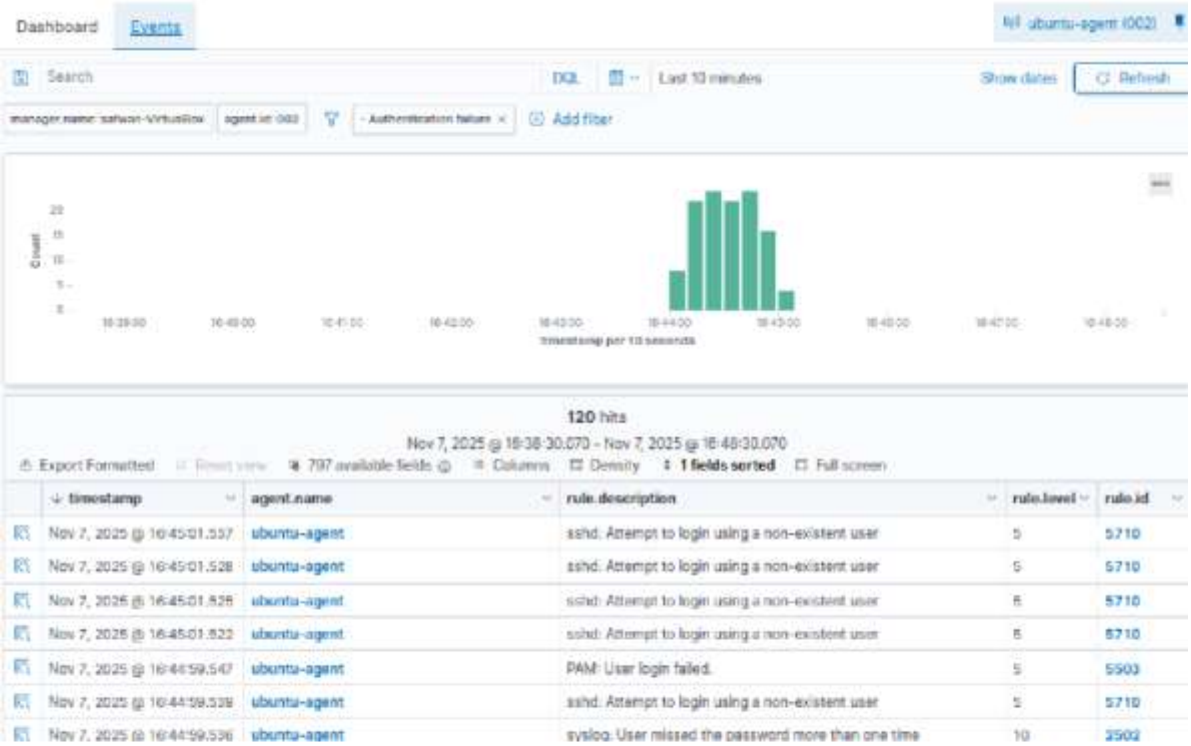
```
  GNU nano 4.8                /var/ossec/etc/ossec.conf

  <localfile>
 <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>
```

This terminal output shows the initial attempt to launch the brute force attack using **Medusa**. The command targets the specified host **ubuntu-agent (192.168.100.134)** using the prepared user and password lists.



```
┌──(safwan㉿kali)-[~]
└─$ medusa -U file.lst -P Passwords.lst -h 192.168.100.134 -M ssh
Medusa v2.3 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

2025-11-07 06:44:07 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Safwan (1 of 5, 0 complete) Password: password123 (1 of 4 complete)
2025-11-07 06:44:09 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Safwan (1 of 5, 0 complete) Password: 12345678 (2 of 4 complete)
2025-11-07 06:44:13 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Safwan (1 of 5, 0 complete) Password: hello123 (3 of 4 complete)
2025-11-07 06:44:16 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Safwan (1 of 5, 0 complete) Password: admin (4 of 4 complete)
2025-11-07 06:44:19 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Tanzeel (2 of 5, 1 complete) Password: password123 (1 of 4 complete
)
2025-11-07 06:44:21 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Tanzeel (2 of 5, 1 complete) Password: 12345678 (2 of 4 complete)
2025-11-07 06:44:25 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Tanzeel (2 of 5, 1 complete) Password: hello123 (3 of 4 complete)
2025-11-07 06:44:28 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Tanzeel (2 of 5, 1 complete) Password: admin (4 of 4 complete)
2025-11-07 06:44:30 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Aatik (3 of 5, 2 complete) Password: password123 (1 of 4 complete)
2025-11-07 06:44:33 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Aatik (3 of 5, 2 complete) Password: 12345678 (2 of 4 complete)
2025-11-07 06:44:37 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Aatik (3 of 5, 2 complete) Password: hello123 (3 of 4 complete)
2025-11-07 06:44:40 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Aatik (3 of 5, 2 complete) Password: admin (4 of 4 complete)
2025-11-07 06:44:42 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Wajahat (4 of 5, 3 complete) Password: password123 (1 of 4 complete
)
2025-11-07 06:44:45 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Wajahat (4 of 5, 3 complete) Password: 12345678 (2 of 4 complete)
2025-11-07 06:44:47 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Wajahat (4 of 5, 3 complete) Password: hello123 (3 of 4 complete)
2025-11-07 06:44:50 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Wajahat (4 of 5, 3 complete) Password: admin (4 of 4 complete)
2025-11-07 06:44:52 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Hamza (5 of 5, 4 complete) Password: password123 (1 of 4 complete)
2025-11-07 06:44:54 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Hamza (5 of 5, 4 complete) Password: 12345678 (2 of 4 complete)
2025-11-07 06:44:59 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Hamza (5 of 5, 4 complete) Password: hello123 (3 of 4 complete)
2025-11-07 06:45:01 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Hamza (5 of 5, 4 complete) Password: admin (4 of 4 complete)
```

This dashboard shows the successful execution of the brute force attack after the network issue was resolved. Medusa systematically cycles through the user and password combinations, generating many failed SSH login attempts, which are immediately sent to the Wazuh Manager for detection.

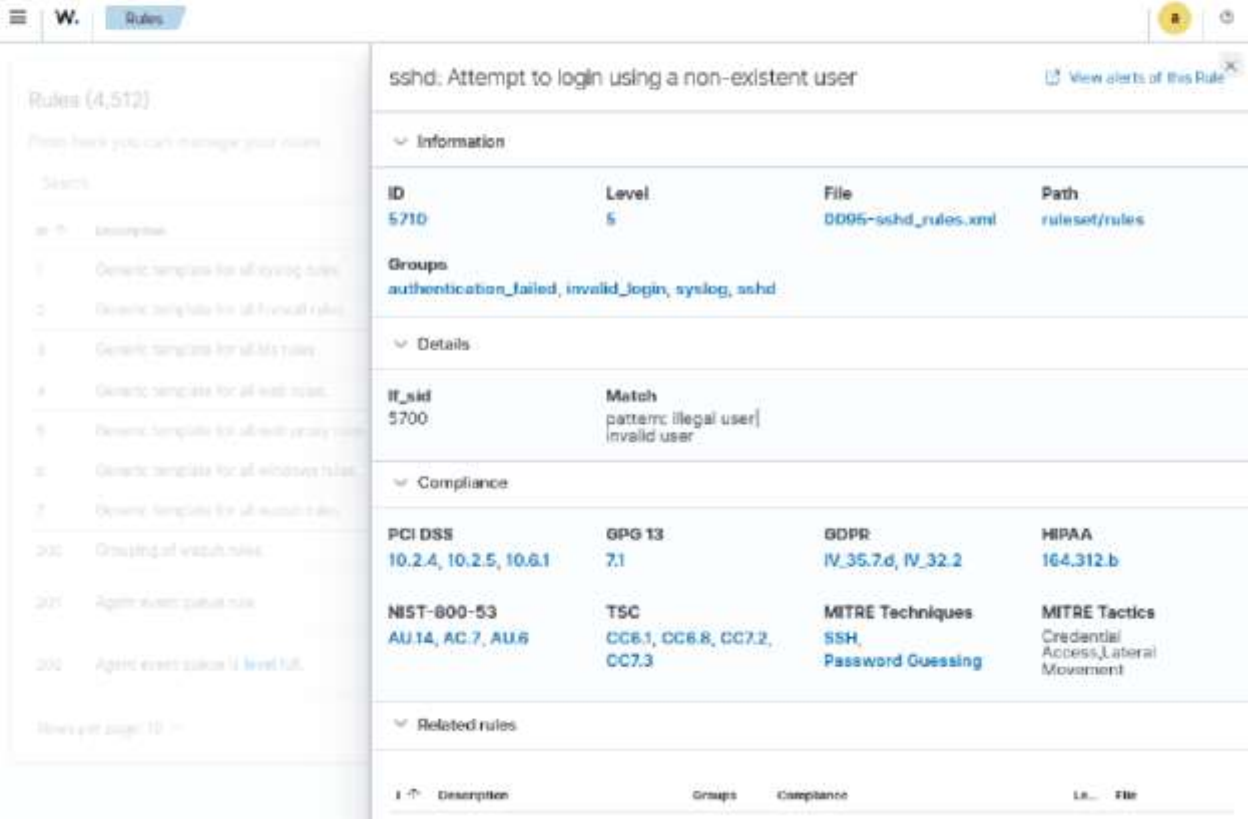


120 hits

Nov 7, 2025 @ 16:38:30.070 - Nov 7, 2025 @ 16:48:30.070

↓ timestamp	agent.name	rule.description	rule.level	rule.id
Nov 7, 2025 @ 16:45:01.537	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 16:45:01.528	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 16:45:01.526	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 16:45:01.522	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 16:44:59.547	ubuntu-agent	PAM: User login failed.	5	5503
Nov 7, 2025 @ 16:44:59.539	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 16:44:59.536	ubuntu-agent	syslog: User missed the password more than one time	10	2502

This Wazuh Dashboard view confirms the details of the specific rule being tested: **ID 5710,** described as "sshd: Attempt to login using a non-existent user." This rule is triggered by failed login attempts and is a key component for detecting the initial phase of the brute force attack.



sshd: Attempt to login using a non-existent user

Information

ID	Level	File	Path
5710	5	0095-sshd_rules.xml	ruleset/rules

Groups
authentication_failed, invalid_login, syslog, sshd

Details

If_sid	Match
5700	pattern: illegal user| invalid user

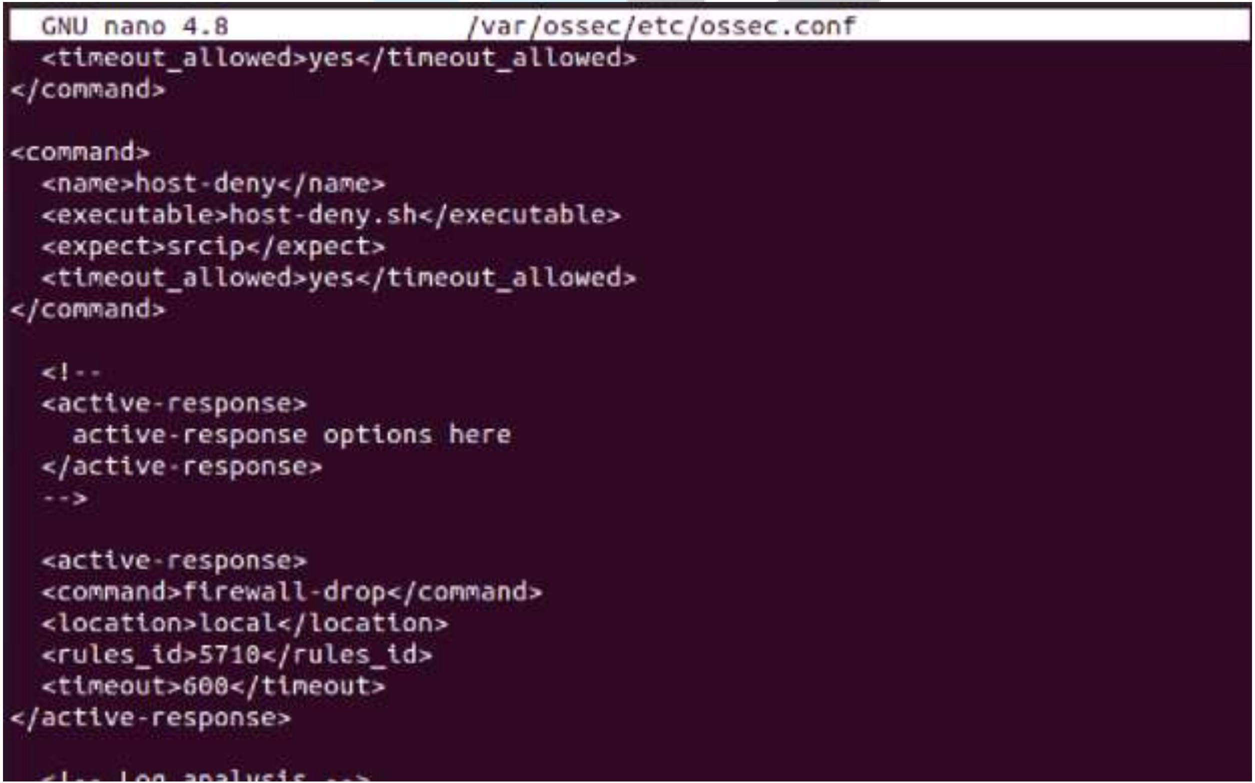
Compliance

PCI DSS	GPG 13	GDPR	HIPAA
10.2.4, 10.2.5, 10.6.1	7.1	IV_35.7.d, IV_32.2	164.312.b
NIST-800-53	**TSC**	**MITRE Techniques**	**MITRE Tactics**
AU.14, AC.7, AU.6	CC6.1, CC6.8, CC7.2, CC7.3	SSH, Password Guessing	Credential Access,Lateral Movement

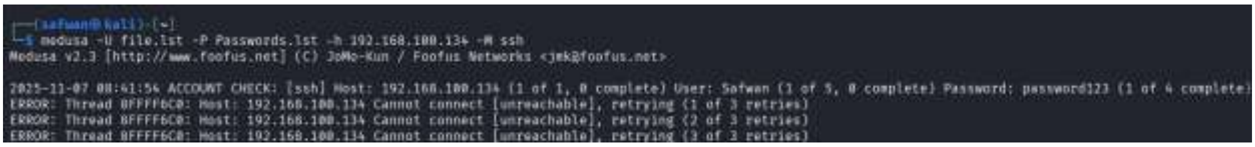
Related rules

I	Description	Groups	Compliance	Le...	File

This screenshot shows the Wazuh Manager's Active Response configuration within ossec.conf. It defines the action: firewall-drop should be executed when Rule ID 5710 (failed SSH login) is triggered, with a timeout of 600 seconds (10 minutes). This is the key configuration that automates the blocking of the attacker's IP address.



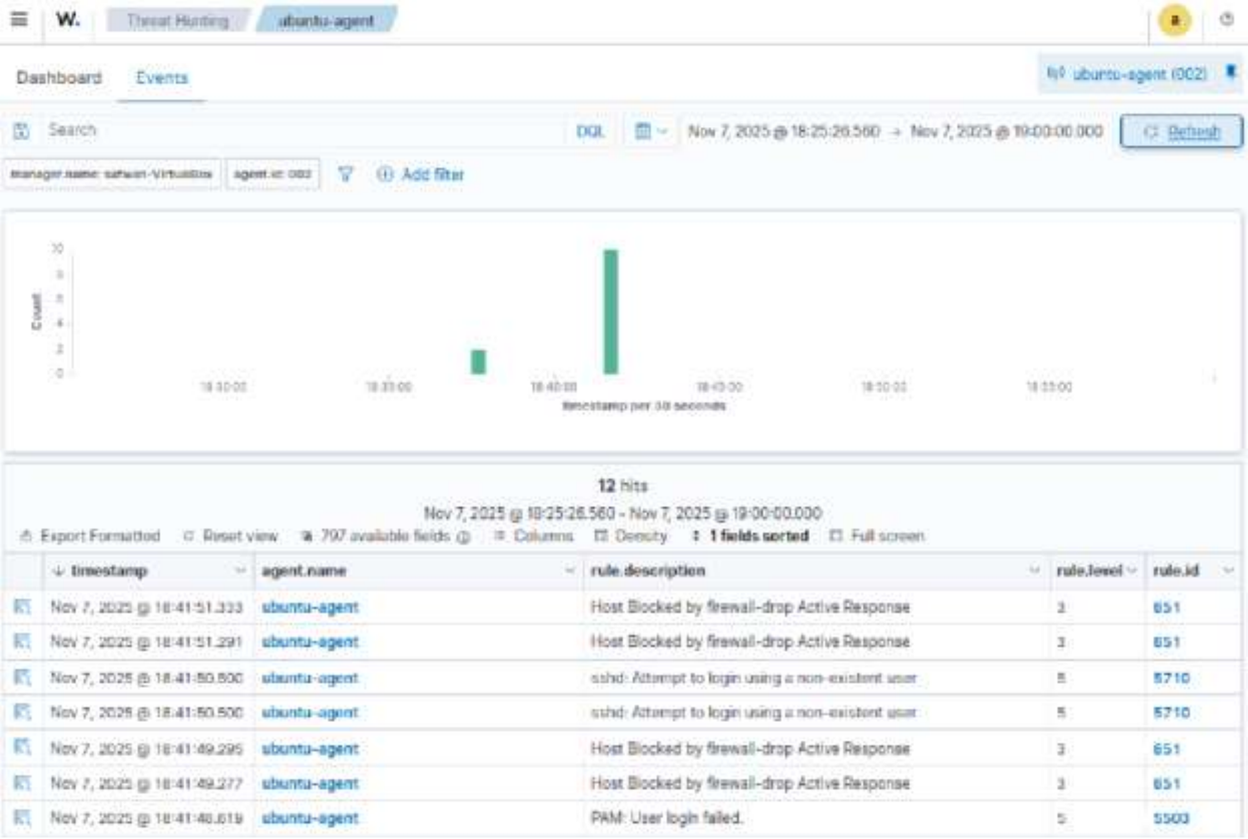
```
 GNU nano 4.8              /var/ossec/etc/ossec.conf
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>host-deny</name>
  <executable>host-deny.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

  <!--
  <active-response>
    active-response options here
  </active-response>
  -->

  <active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <rules_id>5710</rules_id>
  <timeout>600</timeout>
</active-response>

  <!-- Log analysis -->
```



```
┌──(safwan㉿kali)-[~]
└─$ medusa -U file.lst -P Passwords.lst -h 192.168.100.134 -M ssh
Medusa v2.3 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

2025-11-07 08:41:54 ACCOUNT CHECK: [ssh] Host: 192.168.100.134 (1 of 1, 0 complete) User: Safwan (1 of 5, 0 complete) Password: password123 (1 of 4 complete
ERROR: Thread 8FFFF6C0: Host: 192.168.100.134 Cannot connect [unreachable], retrying (1 of 3 retries)
ERROR: Thread 8FFFF6C0: Host: 192.168.100.134 Cannot connect [unreachable], retrying (2 of 3 retries)
ERROR: Thread 8FFFF6C0: Host: 192.168.100.134 Cannot connect [unreachable], retrying (3 of 3 retries)
```

This image from the Wazuh Dashboard shows a large spike of "sshd: Attempt to login using a non-existent user" alerts (Rule ID 5710) generated by the agent during the attack.

The Dashboard now displays **"Host Blocked by firewall-drop Active Response"** alerts (Rule ID 651), generated immediately after the initial SSH failed logins passed the trigger threshold. This proves that the attacker's IP was successfully blocked, stopping the brute force attempt.



Nov 7, 2025 @ 18:25:26.560 → Nov 7, 2025 @ 19:00:00.000

manager.name: safwan-VirtualBox

agent.id: 002

12 hits

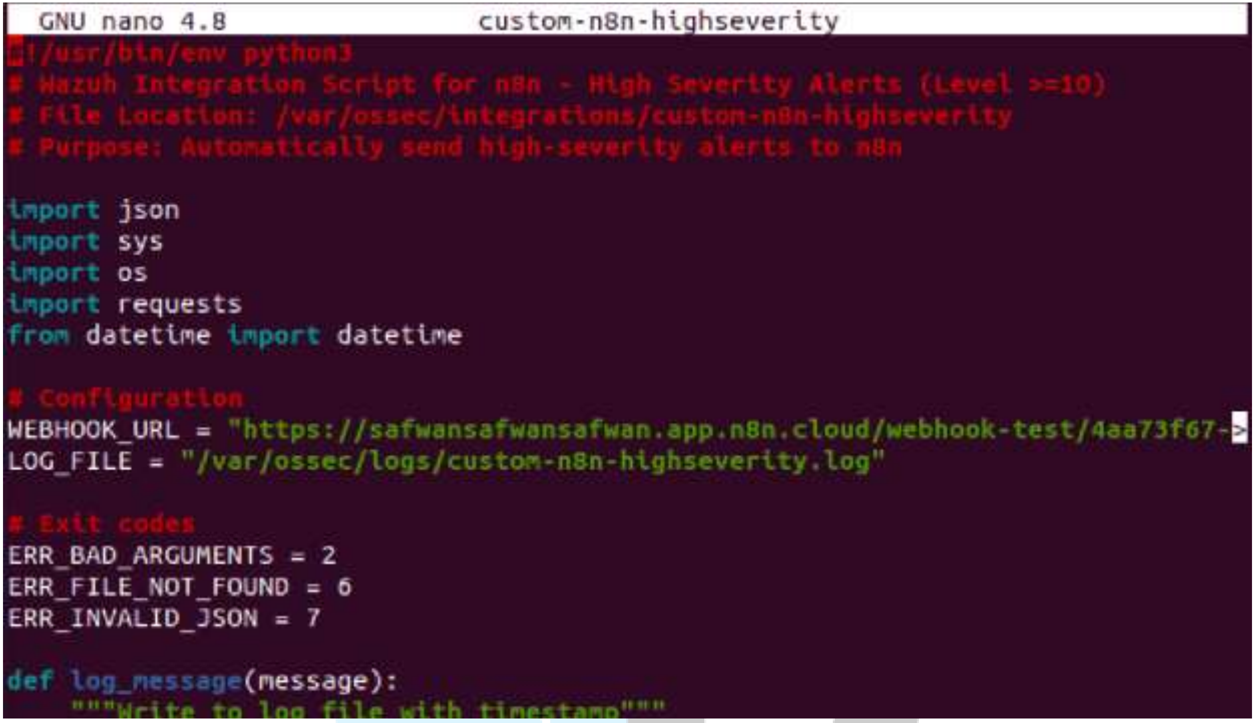
Nov 7, 2025 @ 18:25:26.560 - Nov 7, 2025 @ 19:00:00.000

timestamp	agent.name	rule.description	rule.level	rule.id
Nov 7, 2025 @ 18:41:51.333	ubuntu-agent	Host Blocked by firewall-drop Active Response	3	651
Nov 7, 2025 @ 18:41:51.291	ubuntu-agent	Host Blocked by firewall-drop Active Response	3	651
Nov 7, 2025 @ 18:41:50.500	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 18:41:50.500	ubuntu-agent	sshd: Attempt to login using a non-existent user	5	5710
Nov 7, 2025 @ 18:41:49.295	ubuntu-agent	Host Blocked by firewall-drop Active Response	3	651
Nov 7, 2025 @ 18:41:49.277	ubuntu-agent	Host Blocked by firewall-drop Active Response	3	651
Nov 7, 2025 @ 18:41:48.619	ubuntu-agent	PAM: User login failed.	5	5503

Lab 09: Email Integration with Gmail Using N8N

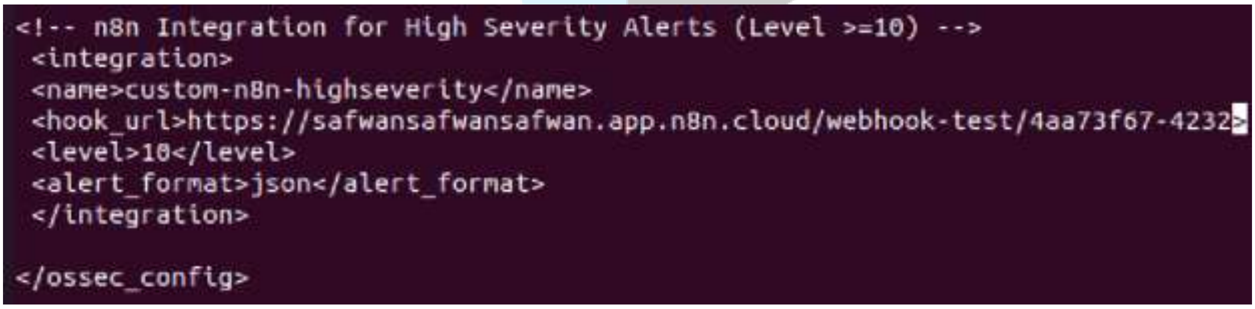
- Filter events for high-priority alerts and trigger tests.
- Submit evidence of received alert emails.

This screenshot displays the Python integration script (custom-n8n-highseverity) placed on the Wazuh Manager. This script is designed to handle alerts and includes the WEBHOOK_URL—the unique address used to send high-severity alerts (Level 10 and above) from the Wazuh Manager to the n8n automation platform.

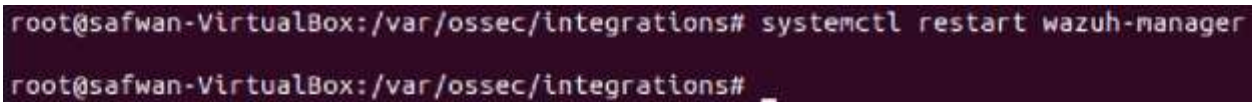


```
  GNU nano 4.8                    custom-n8n-highseverity
#!/usr/bin/env python3
# Wazuh Integration Script for n8n - High Severity Alerts (Level >=10)
# File Location: /var/ossec/integrations/custom-n8n-highseverity
# Purpose: Automatically send high-severity alerts to n8n

import json
import sys
import os
import requests
from datetime import datetime

# Configuration
WEBHOOK_URL = "https://safwansafwansafwan.app.n8n.cloud/webhook-test/4aa73f67->
LOG_FILE = "/var/ossec/logs/custom-n8n-highseverity.log"

# Exit codes
ERR_BAD_ARGUMENTS = 2
ERR_FILE_NOT_FOUND = 6
ERR_INVALID_JSON = 7

def log_message(message):
    """Write to log file with timestamp"""
```

This image shows the Wazuh Manager's ossec.conf file, where the n8n integration is defined. The <integration> block specifies the script's name, provides the webhook_url, and is specifically set to trigger any alert with a <level> of 10 or greater, ensuring only critical events are sent to n8n.

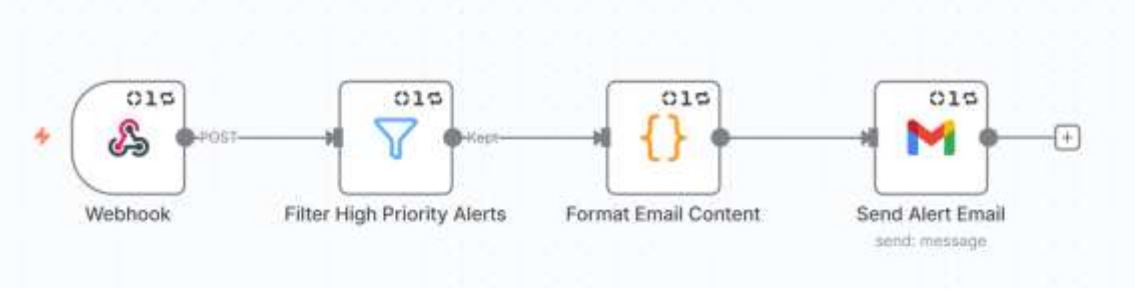


```
<!-- n8n Integration for High Severity Alerts (Level >=10) -->
 <integration>
 <name>custom-n8n-highseverity</name>
 <hook_url>https://safwansafwansafwan.app.n8n.cloud/webhook-test/4aa73f67-4232>
 <level>10</level>
 <alert_format>json</alert_format>
 </integration>

</ossec_config>
```

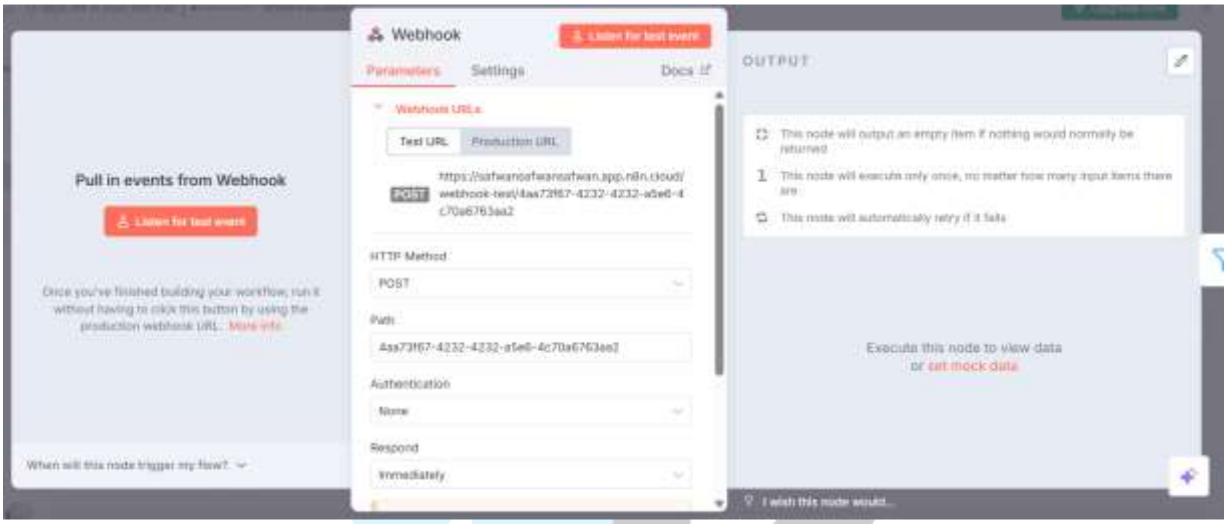


```
root@safwan-VirtualBox:/var/ossec/integrations# systemctl restart wazuh-manager

root@safwan-VirtualBox:/var/ossec/integrations# _
```

This is the high-level view of the entire n8n automation workflow. It clearly shows the four sequential nodes: Webhook (receiving the alert from Wazuh), Filter High Priority Alerts, Format Email Content, and finally, Send Alert Email. This flow processes the raw alert into a deliverable email.



This screenshot details the Webhook node setup within n8n. It displays the unique Test URL that corresponds to the hook_url configured on the Wazuh Manager. This node is configured to listen for the incoming POST request containing the high-severity alert data from the Wazuh integration.


Webhook
Listen for test event
Parameters
Settings
Docs
OUTPUT
I wish this node would...
When will this node trigger my flow?

Pull in events from Webhook

Listen for test event

Once you've finished building your workflow, run it without having to click this button by using the production webhook URL. More info

Webhook URLs

Test URL | Production URL

POST https://safwansafwansafwan.app.n8n.cloud/webhook-test/4aa73f67-4232-4232-a5e6-4c70a6763aa2

HTTP Method: POST

Path: 4aa73f67-4232-4232-a5e6-4c70a6763aa2

Authentication: None

Respond: Immediately

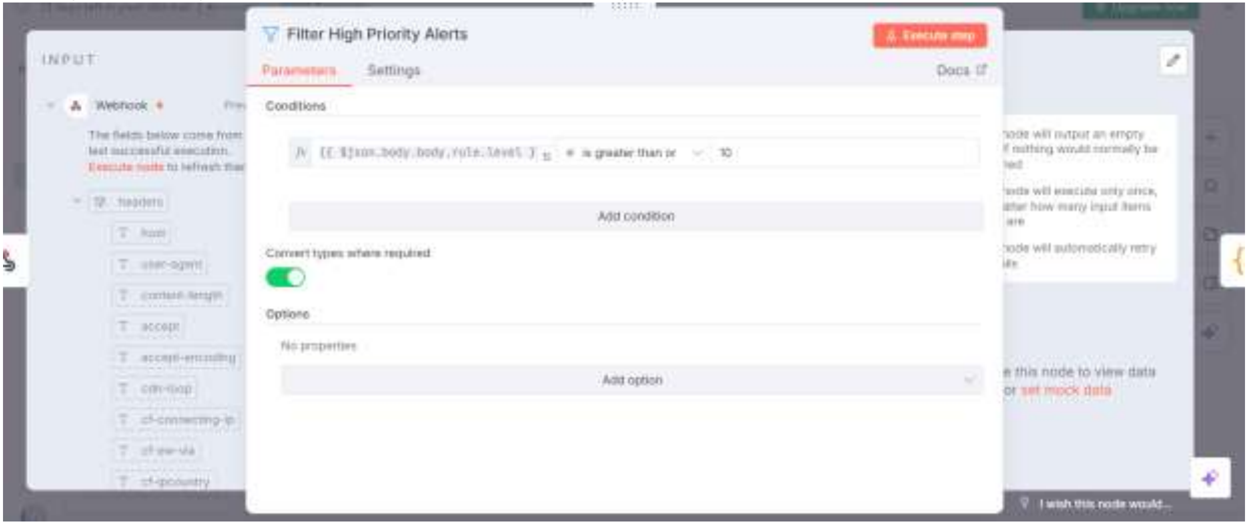
This node will output an empty item if nothing would normally be returned

1 This node will execute only once, no matter how many input items there are

This node will automatically retry if it fails

Execute this node to view data
or set mock data

This image displays the configuration for the Filter High Priority Alerts node. The central logic is shown: the node only allows alerts to pass if the rule.level field in the incoming JSON is is greater than or equal to 10. This ensures the system only sends emails for critical security incidents.



Webhook

The fields below come from last successful execution. Execute node to refresh the

headers
host
user-agent
content-length
accept
accept-encoding
cdn-loop
cf-connecting-ip
cf-ew-via
cf-ipcountry

Conditions

{{ $json.body.body.rule.level }} is greater than or 10

Convert types where required

Options

No properties

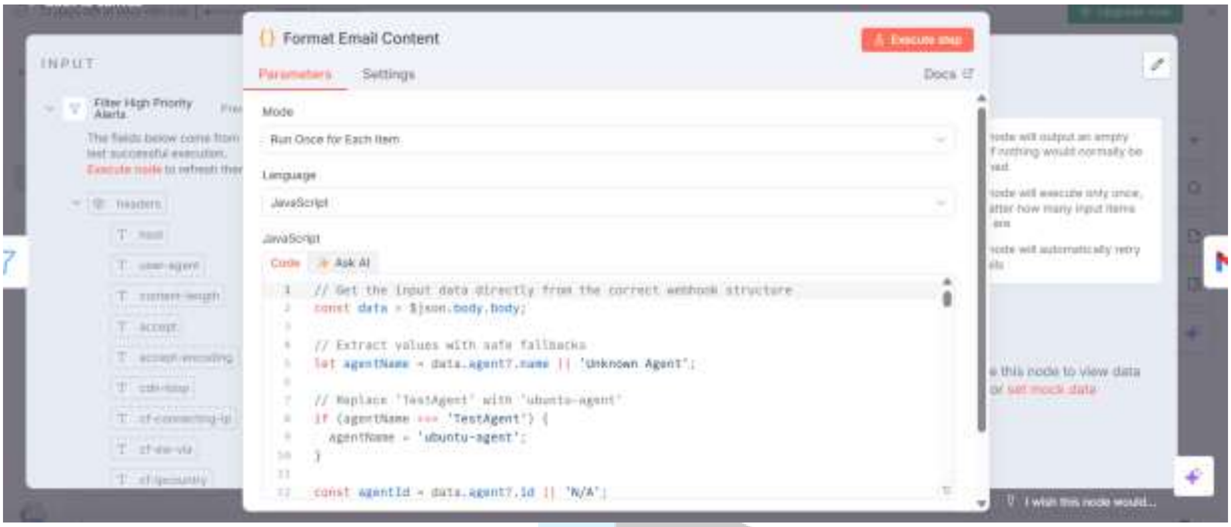
node will output an empty if nothing would normally be ned

node will execute only once, atter how many input items are

node will automatically retry ils

e this node to view data
or set mock data

This screenshot shows the logic used by the Format Email Content node. It utilizes JavaScript to parse the raw alert JSON, safely extracting key variables like the agentName and agentId for use in the final email body, ensuring the notification is readable and contains essential security information.



Format Email Content

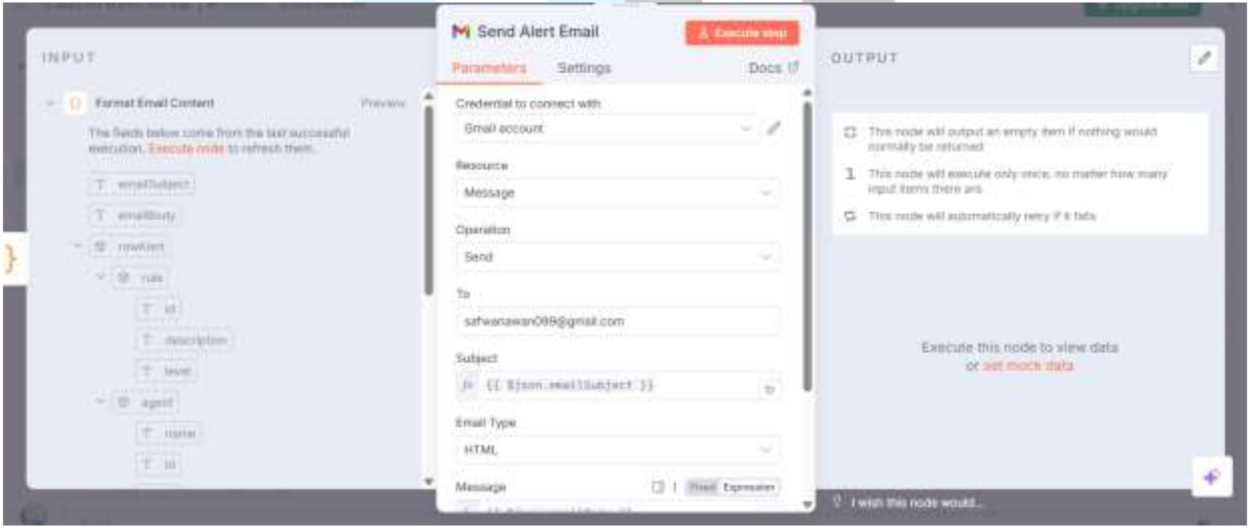
Mode: Run Once for Each Item

Language: JavaScript

JavaScript

```
// Get the input data directly from the correct webhook structure
const data = $json.body.body;

// Extract values with safe fallbacks
let agentName = data.agent?.name || 'Unknown Agent';

// Replace 'TestAgent' with 'ubuntu-agent'
if (agentName === 'TestAgent') {
  agentName = 'ubuntu-agent';
}

const agentId = data.agent?.id || 'N/A';
```

This final configuration screenshot details the Send Alert Email node. It is configured to use a Gmail account, set to the Send operation, and specifies the recipient email address. This confirms the final action of the workflow is to deliver the processed high-priority security alert.



Send Alert Email

Credential to connect with: Gmail account

Resource: Message

Operation: Send

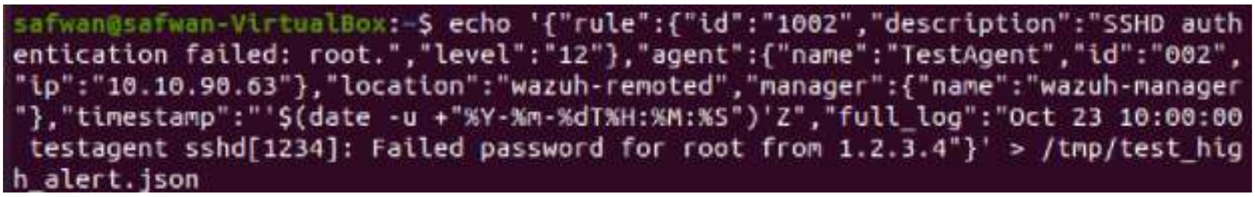
To: safwanawan098@gmail.com

Subject: {{ $json.emailSubject }}

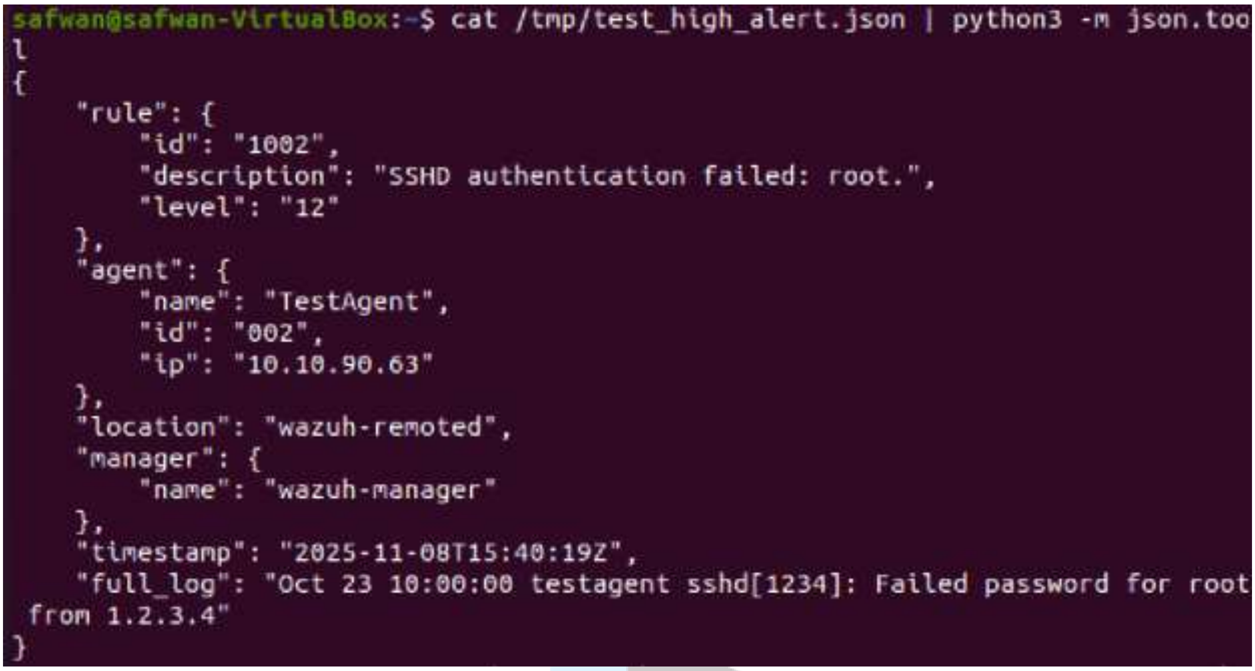
Email Type: HTML

Message

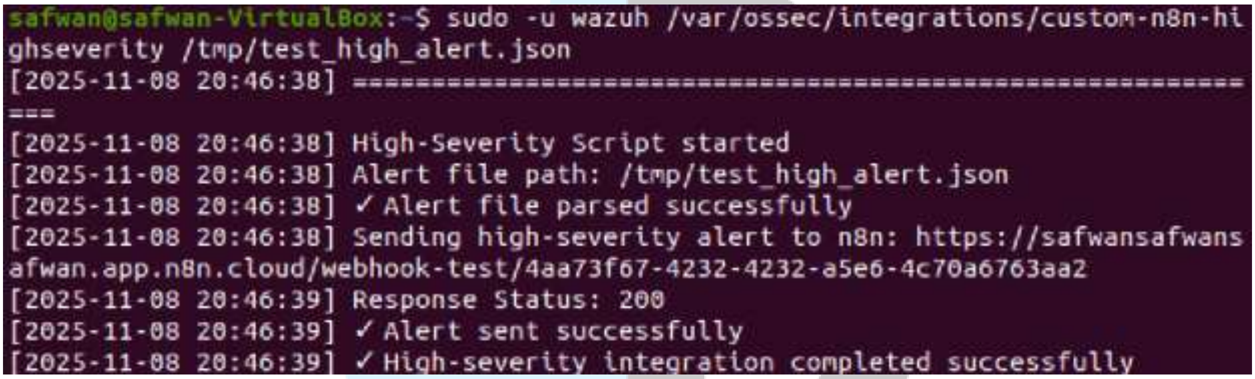
INPUT
Format Email Content
Preview
The fields below come from the last successful execution. Execute node to refresh them.
emailSubject
emailBody
rowAlert
rule
id
description
level
agent
name
id
Execute step
Parameters
Settings
Docs
Fixed
Expression
OUTPUT
This node will output an empty item if nothing would normally be returned
This node will execute only once, no matter how many input items there are
This node will automatically retry if it fails
Execute this node to view data
or set mock data
I wish this node would...



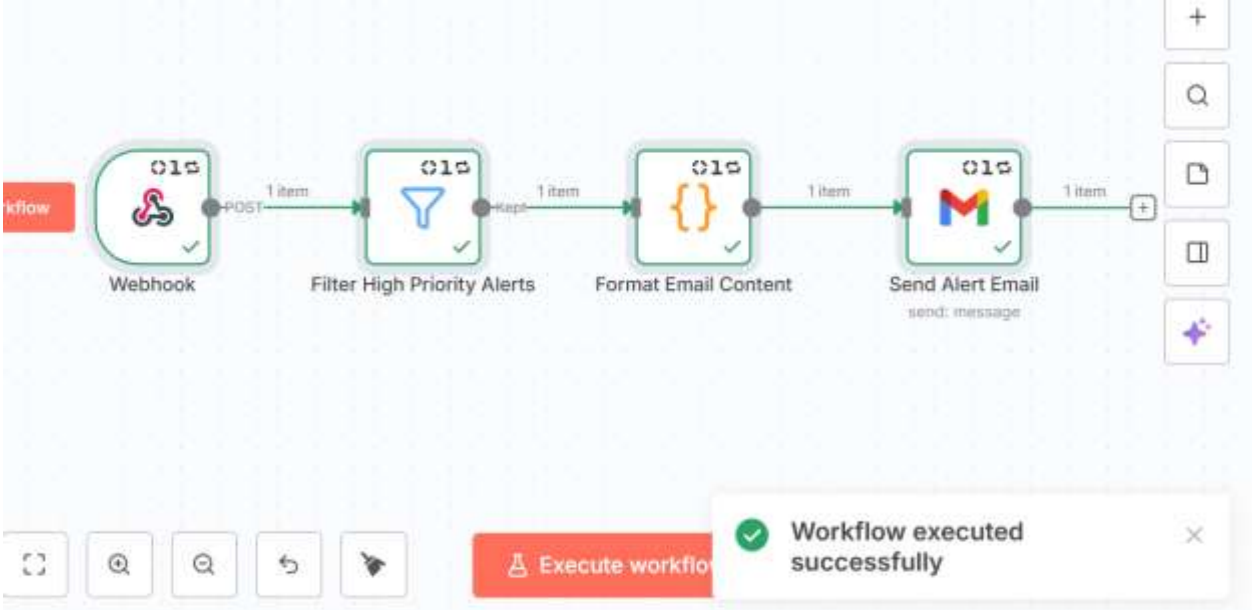
```
safwan@safwan-VirtualBox:~$ echo '{"rule":{"id":"1002","description":"SSHD auth
entication failed: root.","level":"12"},"agent":{"name":"TestAgent","id":"002",
"ip":"10.10.90.63"},"location":"wazuh-remoted","manager":{"name":"wazuh-manager
"},"timestamp":"'$(date -u +"%Y-%m-%dT%H:%M:%S")'Z","full_log":"Oct 23 10:00:00
 testagent sshd[1234]: Failed password for root from 1.2.3.4"}' > /tmp/test_hig
h_alert.json
```



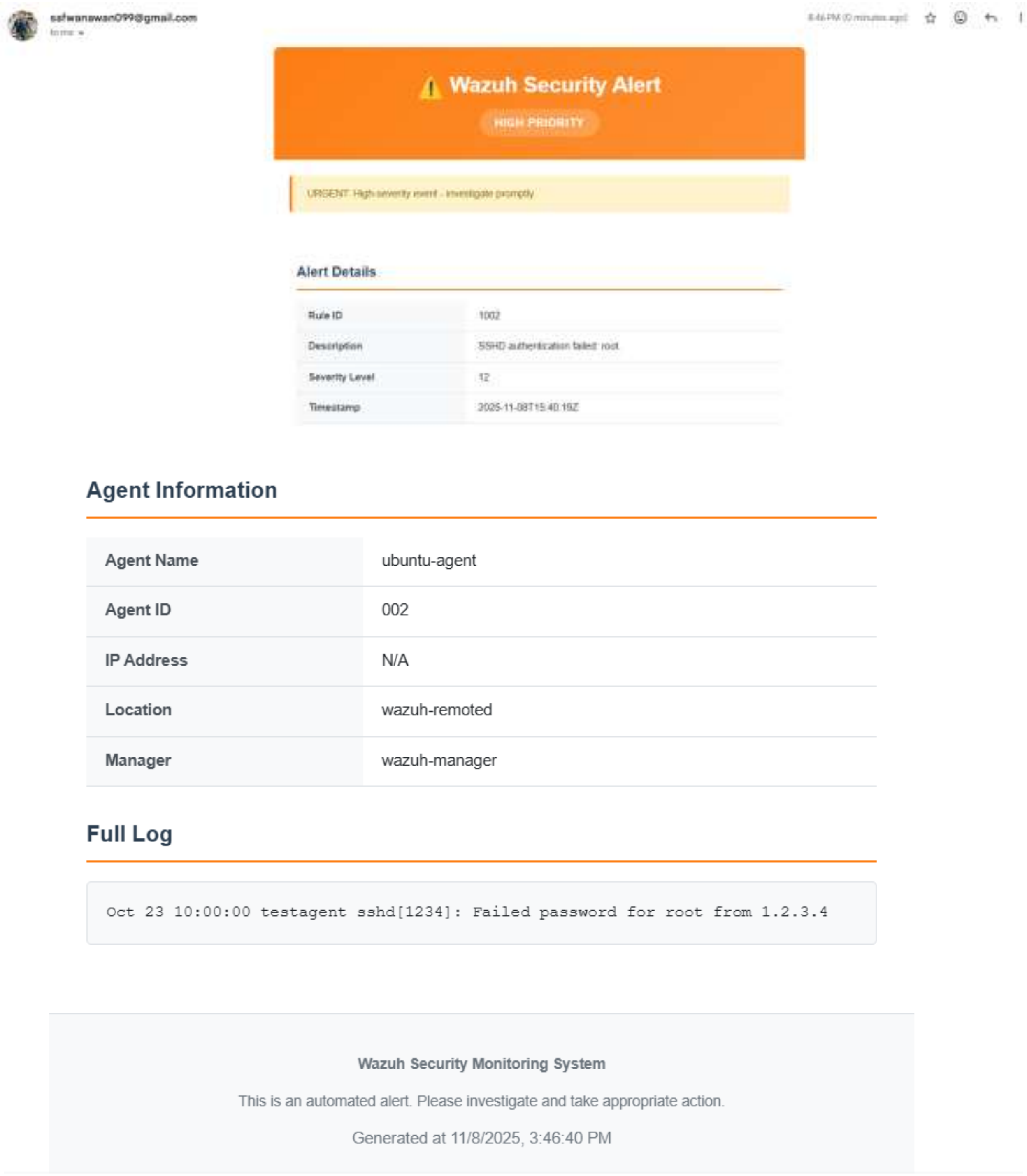
```
safwan@safwan-VirtualBox:~$ cat /tmp/test_high_alert.json | python3 -m json.too
l
{
    "rule": {
        "id": "1002",
        "description": "SSHD authentication failed: root.",
        "level": "12"
    },
    "agent": {
        "name": "TestAgent",
        "id": "002",
        "ip": "10.10.90.63"
    },
    "location": "wazuh-remoted",
    "manager": {
        "name": "wazuh-manager"
    },
    "timestamp": "2025-11-08T15:40:19Z",
    "full_log": "Oct 23 10:00:00 testagent sshd[1234]: Failed password for root
 from 1.2.3.4"
}
```



```
safwan@safwan-VirtualBox:~$ sudo -u wazuh /var/ossec/integrations/custom-n8n-hi
ghseverity /tmp/test_high_alert.json
[2025-11-08 20:46:38] ==========================================================
===
[2025-11-08 20:46:38] High-Severity Script started
[2025-11-08 20:46:38] Alert file path: /tmp/test_high_alert.json
[2025-11-08 20:46:38] ✓ Alert file parsed successfully
[2025-11-08 20:46:38] Sending high-severity alert to n8n: https://safwansafwans
afwan.app.n8n.cloud/webhook-test/4aa73f67-4232-4232-a5e6-4c70a6763aa2
[2025-11-08 20:46:39] Response Status: 200
[2025-11-08 20:46:39] ✓ Alert sent successfully
[2025-11-08 20:46:39] ✓ High-severity integration completed successfully
```



Workflow executed successfully



safwanawan099@gmail.com
to me

8:46 PM (0 minutes ago)

⚠ Wazuh Security Alert

HIGH PRIORITY

URGENT: High severity event - investigate promptly

Alert Details

Rule ID	1002
Description	SSHD authentication failed: root
Severity Level	12
Timestamp	2025-11-08T15:40:19Z

Agent Information

Agent Name	ubuntu-agent
Agent ID	002
IP Address	N/A
Location	wazuh-remoted
Manager	wazuh-manager

Full Log

```
Oct 23 10:00:00 testagent sshd[1234]: Failed password for root from 1.2.3.4
```

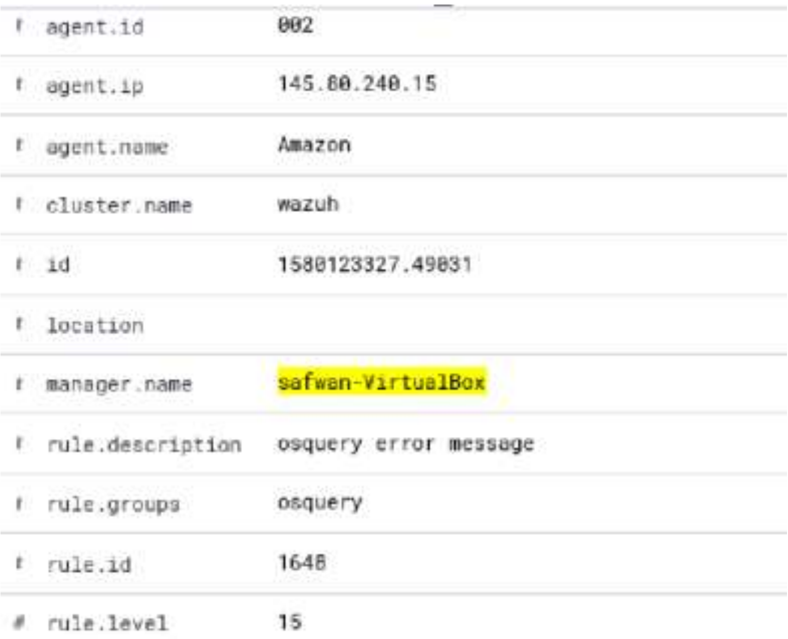
Wazuh Security Monitoring System

This is an automated alert. Please investigate and take appropriate action.

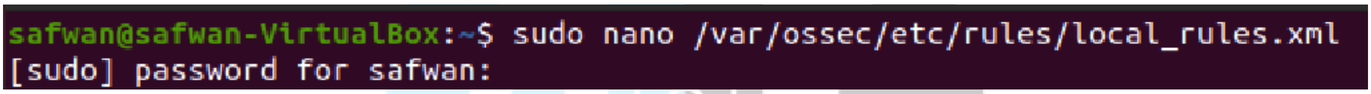
Generated at 11/8/2025, 3:46:40 PM

Lab 10: Basic Rule Modification Exercise

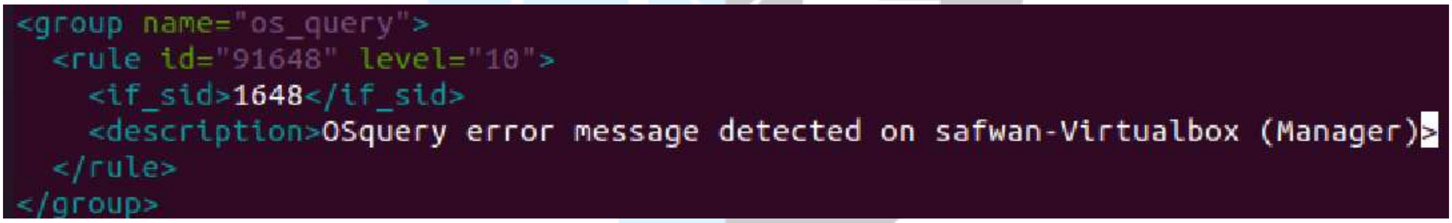
- Identify and modify an existing Wazuh detection rule.
- Use the Ruleset Test tool to verify custom rule functionality.
- Submit screenshots and a short note on your findings.



t	agent.id	002
t	agent.ip	145.80.240.15
t	agent.name	Amazon
t	cluster.name	wazuh
t	id	1580123327.49831
t	location	
t	manager.name	safwan-VirtualBox
t	rule.description	osquery error message
t	rule.groups	osquery
t	rule.id	1648
#	rule.level	15



```
safwan@safwan-VirtualBox:~$ sudo nano /var/ossec/etc/rules/local_rules.xml
[sudo] password for safwan:
```



```
<group name="os_query">
  <rule id="91648" level="10">
    <if_sid>1648</if_sid>
    <description>OSquery error message detected on safwan-Virtualbox (Manager)>
  </rule>
</group>
```