Wazuh POC Report

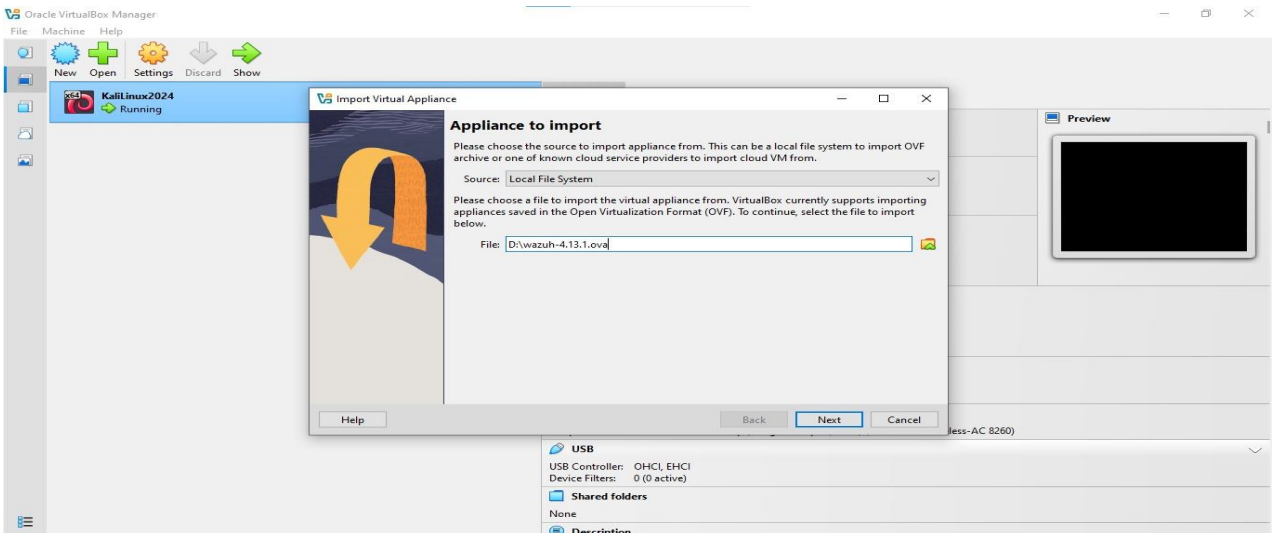
Submitted by: Hiba Zahid
Submitted to: Sir Basit
Task: 01 — Wazuh POC Setup and Use Case Demonstration
Environment: VirtualBox (OVA), Wazuh Manager + Dashboard, two endpoint VMs (Target and Attacker)

1. Objective

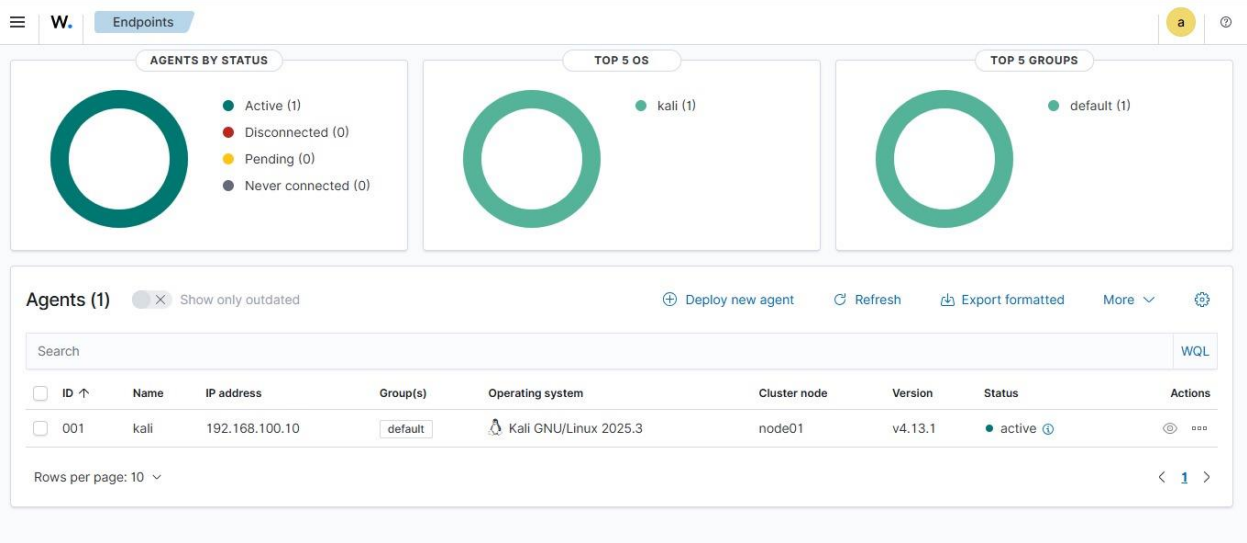
The objective of this POC was to set up a Wazuh lab environment and demonstrate a basic use case for threat detection and response. The goal was to deploy Wazuh central components and at least two monitored endpoints, then simulate a controlled security incident (brute-force SSH login attempts) to validate detection and alerting capabilities.

2. Lab Setup and Preparation

- **Virtualization platform:** VirtualBox (imported preconfigured Wazuh OVA).



- **Central components:** Wazuh manager.
- **Endpoint VMs:**
  - **Target VM (monitored endpoint):** Wazuh agent installed
  - **Attacker VM:** Kali Linux is used to simulate attacks.



- **Tools on attacker:** `sshpass` installed to perform scripted SSH authentication attempts.

Verification performed prior to testing:

- Confirmed SSH service running on target VM: `sudo systemctl status ssh`
- Confirmed Wazuh agent running on target VM: `sudo systemctl status wazuh-agent`
- Verified agents listed and connected in Wazuh Dashboard (Agent Management / Monitoring).

3. Agent Deployment and Registration

1. Imported Wazuh OVA and booted the central components.
2. Deployed endpoint VMs and installed the Wazuh agent on each using the official Wazuh installer steps.
3. Registered each agent with the Wazuh manager and validated connectivity in the dashboard. Agents displayed as connected and reporting.

4. Use Case Selection and Safety Measures

Selected use case: Controlled brute-force SSH login attempts against a test user on the monitored target VM.
Rationale: SSH brute-force attempts are common and easily simulated in an isolated lab; detection is expected by default Wazuh authentication rules.

Safety measures implemented:

- Performed tests only within isolated lab VMs (never on external or production networks).
- Created a dedicated test user (`testuser`) on the target VM.
- Limited the number of attempts to 15.
- Took pre-test snapshots for rollback.

- Removed the test user after completion.

5. Test Implementation

Create test user on target VM

```
sudo useradd -m testuser
echo "testuser:Test@1234" | sudo chpasswd
```

```
┌──(root㉿kali)-[~]
└─# sudo useradd -m testuser

┌──(root㉿kali)-[~]
└─# sudo useradd -m testuser
useradd: user 'testuser' already exists

┌──(root㉿kali)-[~]
└─# sudo passwd testuser
New password:
Retype new password:
passwd: password updated successfully
```

Install sshpass on attacker VM (Kali)

```
sudo apt update
sudo apt install -y sshpass
```

```
┌──(root㉿kali)-[~]
└─# sudo apt update
sudo apt install -y sshpass

Hit:1 https://packages.wazuh.com/4.x/apt stable InRelease
Ign:2 http://http.kali.org kali-rolling InRelease
Err:3 http://http.kali.org kali-rolling Release
  Redirection loop encountered
Error: The repository 'http://http.kali.org/kali kali-rolling Release' no lon
ger has a Release file.
Notice: Updating from such a repository can't be done securely, and is theref
ore disabled by default.
Notice: See apt-secure(8) manpage for repository creation and user configurat
ion details.
Installing:
  sshpass

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 617
  Download size: 12.7 kB
  Space needed: 41.0 kB / 35.1 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 sshpass amd64 1.10-0.
1 [12.7 kB]
Fetched 12.7 kB in 6s (2,062 B/s)
Selecting previously unselected package sshpass.
(Reading database ... 417103 files and directories currently installed.)
Preparing to unpack .../sshpass_1.10-0.1_amd64.deb ...
Unpacking sshpass (1.10-0.1) ...
Setting up sshpass (1.10-0.1) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for kali-menu (2025.3.2) ...

┌──(root㉿kali)-[~]
```

Controlled brute-force loop (attacker VM — Kali)

Replace `TARGET_IP` with the target VM IP:

```
TARGET_IP=192.168.56.101
for i in {1..15}; do
```

```
  echo "Attempt $i"
  sshpass -p 'WrongPass123!' ssh -o StrictHostKeyChecking=no -o
ConnectTimeout=5 -o BatchMode=no testuser@$TARGET_IP 'exit'
  sleep 1
done
```

```
Session Actions Edit View Help
┌──(root㉿kali)-[~]
└─# TARGET_IP=192.168.100.9
for i in {1..15}; do
  echo "Attempt $i"
  sshpass -p 'WrongPass123!' ssh -o StrictHostKeyChecking=no -o ConnectTimeou
t=5 -o BatchMode=no testuser@$TARGET_IP 'exit'
  sleep 1
done
Attempt 1
Warning: Permanently added '192.168.100.9' (ECDSA) to the list of known hosts
.
Permission denied, please try again.
Attempt 2
Permission denied, please try again.
Attempt 3
Permission denied, please try again.
Attempt 4
Permission denied, please try again.
Attempt 5
Permission denied, please try again.
Attempt 6
Permission denied, please try again.
Attempt 7
Permission denied, please try again.
Attempt 8
Permission denied, please try again.
Attempt 9
Permission denied, please try again.
Attempt 10
Permission denied, please try again.
Attempt 11
Permission denied, please try again.
Attempt 12
Permission denied, please try again.
```

Optional: Monitor authentication logs on target VM

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

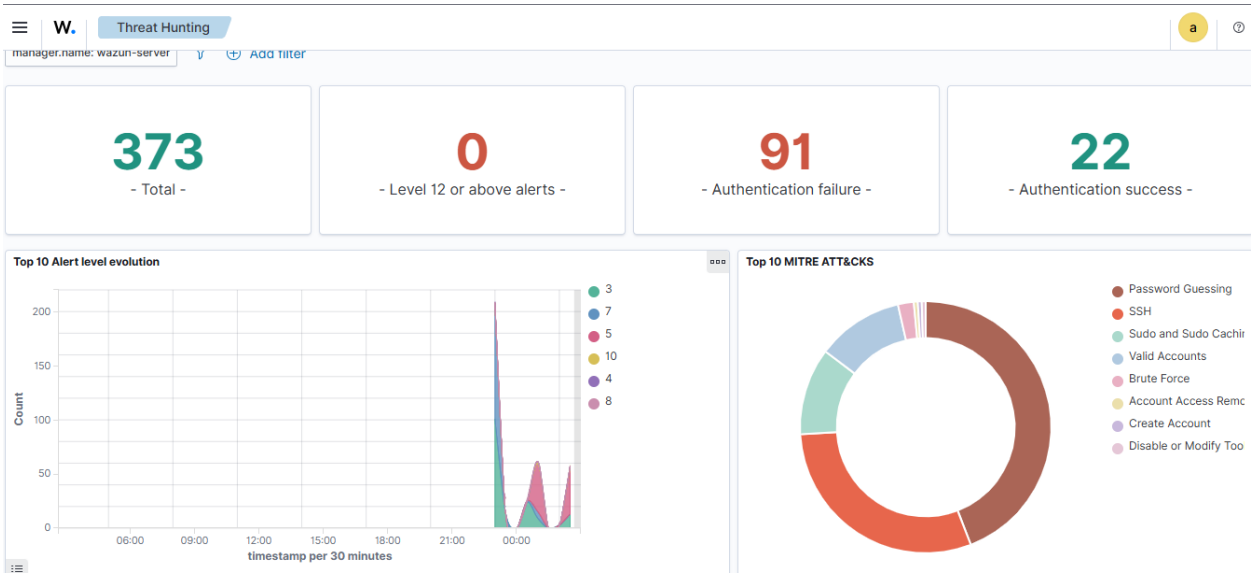
---

6. Detection and Dashboard Analysis

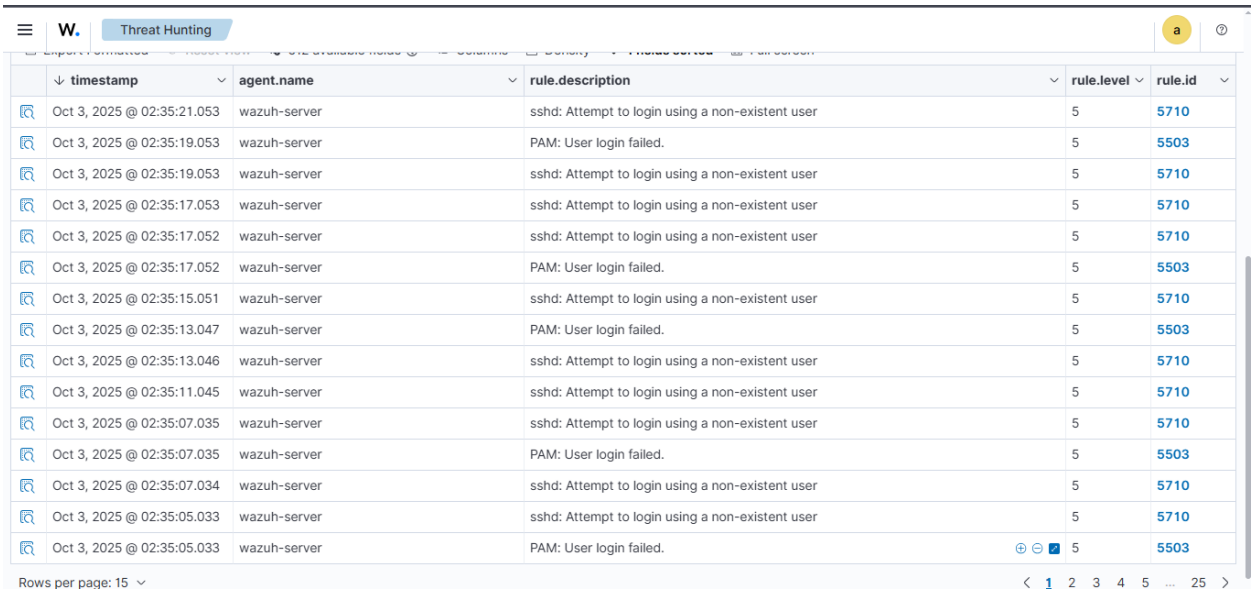
Dashboard navigation: Wazuh Dashboard → *Explore* → *Discover* (Index pattern: `wazuh-alerts-*`)
Time filter used: Last 24 hours(adjusted as needed).

Observed results:

- New alert rows appeared shortly after the brute-force loop completed. Alerts corresponded to failed SSH authentication events.
- Visible alert fields



- Alerts were accessible via Threat Hunting and Discover, and details showed the originating agent and timestamps, enabling straightforward triage.



Cleanup

After testing the following cleanup actions were executed:

```
sudo userdel -r testuser
```

Alternatively, the test user password can be changed to a secure value. If any unexpected changes occurred, VMs can be restored from snapshots.

Conclusion

The Wazuh POC successfully demonstrated that the platform can detect and alert on a simulated brute-force SSH attack in an isolated lab environment. Agents were properly registered and visible in the dashboard; the dashboard ingested and displayed relevant authentication failure alerts with adequate context (agent identity, timestamps, rule metadata).

Recommendations

1. **Tune rule thresholds and correlation** to consolidate repeated failed attempts into manageable incidents.
2. **Implement automated containment** (e.g., Fail2Ban integration or automatic firewall rules) to block suspicious sources upon detection.
3. **Harden SSH configuration** on endpoints (disable password authentication, use key-based auth, disable root login, and consider non-default ports for additional obscurity).
4. **Create saved searches / dashboards** for common detections (SSH brute-force, privilege escalation, suspicious file access) to speed up triage.
5. **Document test playbooks** and retain snapshots before tests to ensure safe, repeatable experimentation.

References

- Wazuh Proof of Concept Guide — Wazuh Documentation: https://documentation.wazuh.com/current/proof-of-concept-guide/index.html

Summary

A Wazuh lab (manager, indexer, dashboard) was set up using a preconfigured OVA and snapshots were created. Two endpoint VMs were deployed and the Wazuh agent was installed and registered. A controlled brute-force SSH attack was simulated from a Kali attacker VM against a test user on a monitored target VM (15 incorrect attempts). Wazuh detected the failed authentication events and generated alerts visible in the dashboard (Discover/Threat Hunting).