Implementation of Security Operations Center Based on Wazuh Tool

1. Goal and Tasks of the Project

Goal

- To build a fully functional Security Operations Center (SOC) using open-source tools (Wazuh (SIEM), MISP (Threat Intelligence Platform), IRIS (Ticketing System)).
- Enhance capabilities for detecting, analyzing, and responding to security incidents using SIEM, threat intelligence, and a ticketing system.

Tasks and Responsibilities

1. **Mohamad Nour Shahin:** Infrastructure Setup.
    - Set up the virtual environment for the SOC.
    - Install and configure the Wazuh SIEM tool.
    - Document the installation and configuration process for Wazuh.
    - Create a basic incident dashboard in Wazuh for tracking events.
    - Ensure connectivity between Wazuh and other tools (e.g., MISP, IRIS).
2. **Yehia Sobeh:** Integration of MISP for threat intelligence.
    - Install and configure MISP as the threat intelligence platform.
    - Integrate MISP with Wazuh for contextual threat information.
    - Document the integration process and configuration steps.
    - Test threat intelligence data flow into Wazuh and provide sample scenarios.
    - Propose mechanisms to enrich threat data using MISP.

- **Ammar Meslmani:** Configuration of IRIS for case management and incident automation.
    - Install and configure IRIS as the ticketing system for incident tracking.
    - Link IRIS with Wazuh to automatically log security incidents.
    - Develop and test automated responses to at least two security incidents (e.g., failed login attempts, malware detection).
    - Document automation workflows and response mechanisms.
    - Simulate the incident management process for the demo.

- **Ali Hamdan:** Testing, incident simulation, and documentation.
    - Test the overall integration and functionality of the SOC setup.
    - Identify and document potential issues during integration and solutions implemented.
    - Write scripts to simulate security incidents for testing (e.g., running a vulnerability scanner, mock phishing attacks).
    - Prepare the final demo presentation and record it for submission.
    - Organize and compile the documentation from all members into a cohesive report.

---

2. Execution Plan and Methodology

Plan for the Solution

1. **Setup:**
    - Deploy virtual machines or Docker containers for Wazuh, MISP, and IRIS.
    - Establish network connectivity and API integrations.
2. **Integration:**
    - Configure Wazuh to collect logs and generate alerts.
    - Integrate MISP for threat intelligence enrichment.
    - Set up IRIS for incident tracking and automation workflows.
3. **Testing:**
    - Simulate real-world security incidents (e.g., brute force attacks, malware detection).
    - Automate responses and document workflows.

Planned Infrastructure

- Diagram illustrating connectivity between Wazuh, MISP, IRIS, and client systems.

Methodology

- Follow official documentation for tool installations and configurations.
- Implement and test workflows for data flow, alert generation, and automated responses.
- Document challenges and solutions.

---

3. Development of Solution and Tests (Proof of Concept)

Environment Preparation

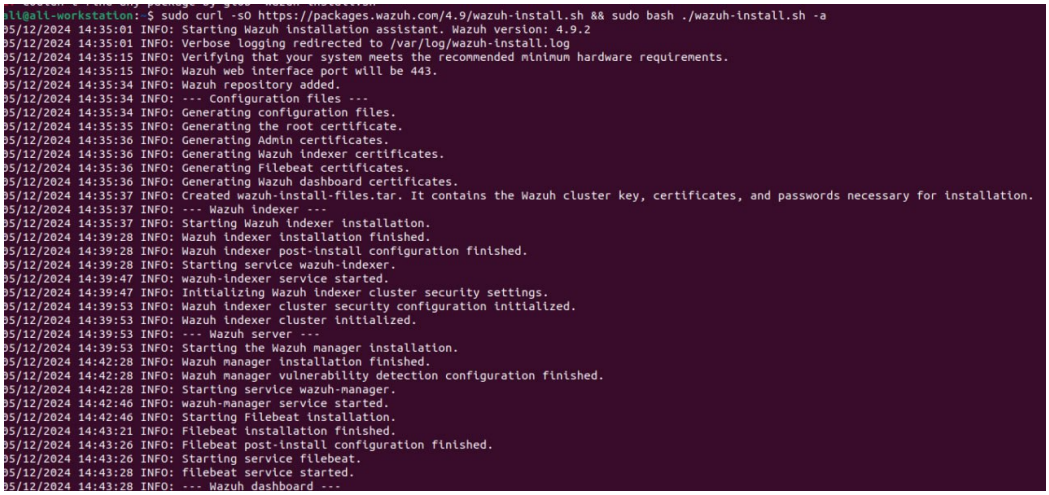
- Description of the SOC setup: VMs/containers, OS versions, and configurations.

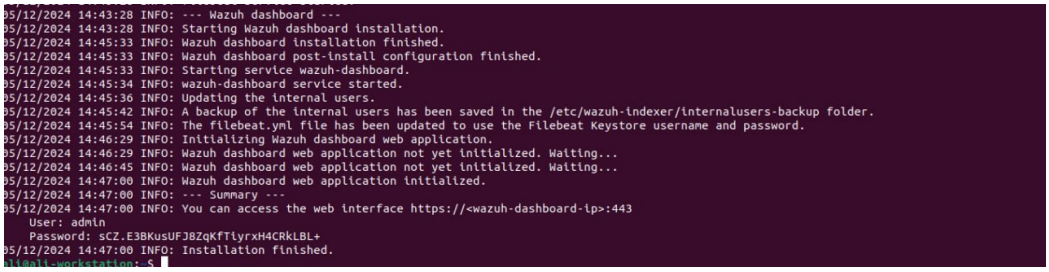
    1. Setup Wazuh server on Ubuntu 22.04 (Ali Device number one):

        - Download and run the Wazuh installation assistant:

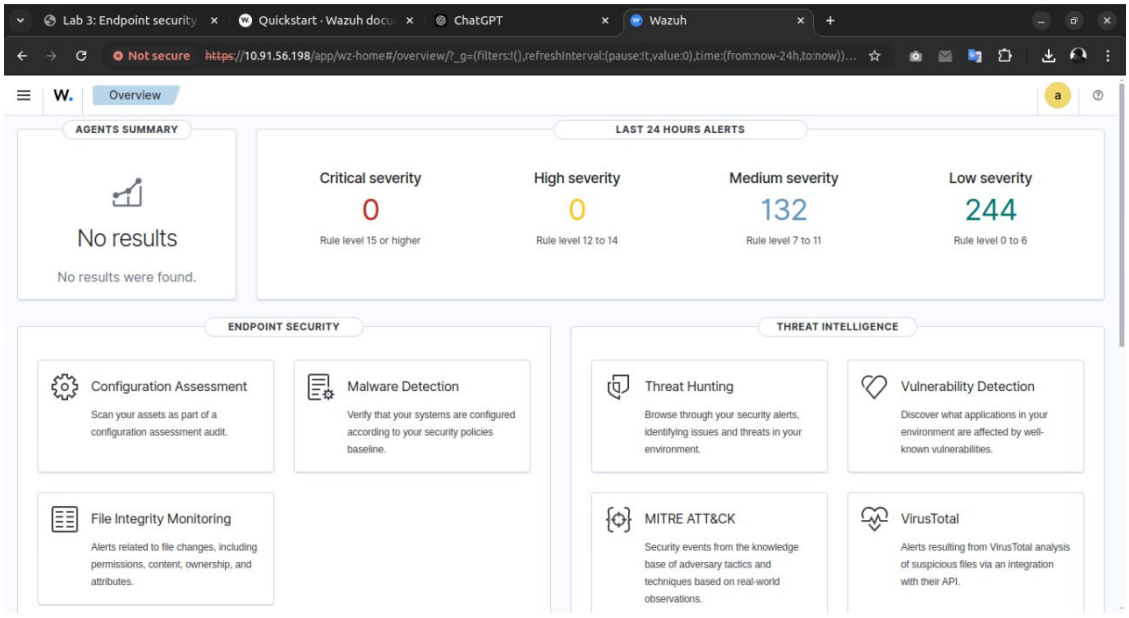
```
  curl -sO https://packages.wazuh.com/4.9/wazuh-install.sh
&& sudo bash ./wazuh-install.sh -a -o
```

- Configure Wazuh, install Wazuh Indexer, install Wazuh Server, and install Wazuh Dashboard.





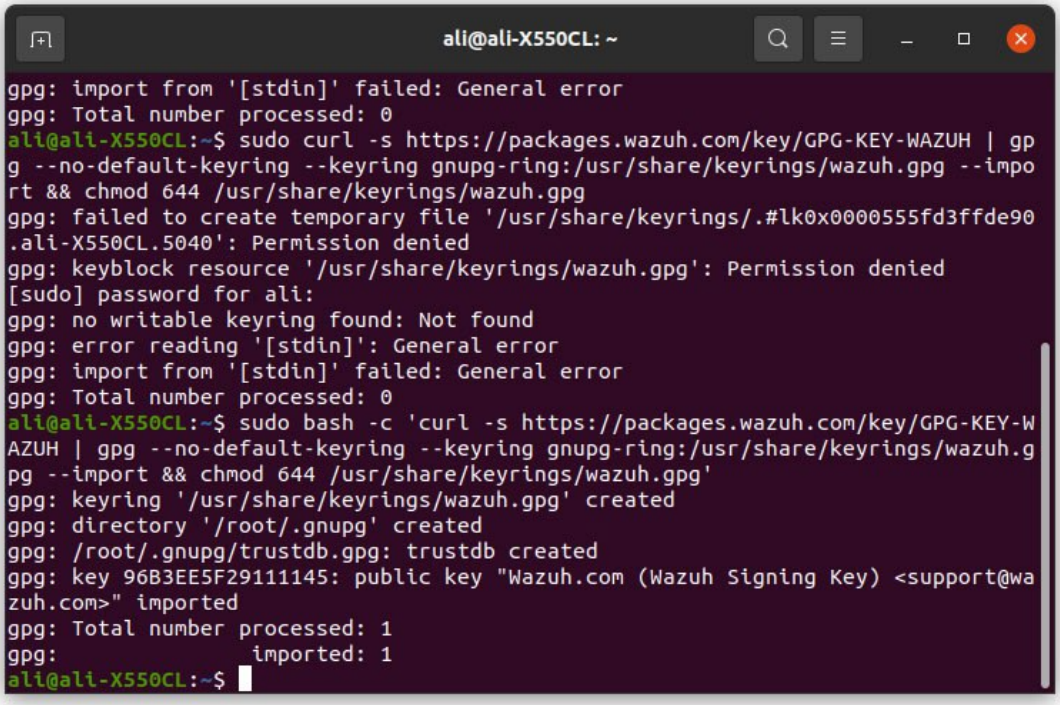
- Access the Wazuh web interface with `https://10.91.56.198:443` and my credentials:



2. Setup Wazuh Agent on Ubuntu 20.04 (Ali Device number two):

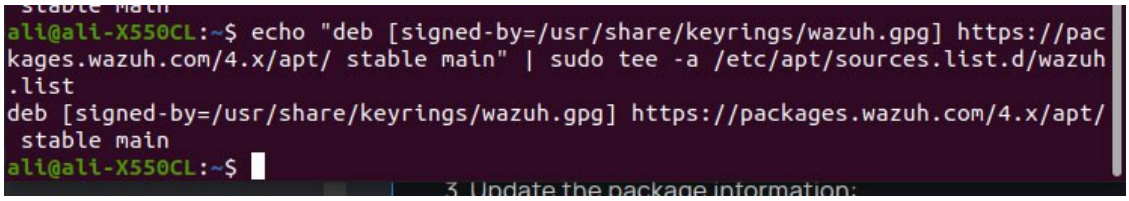
- Install the GPG key:

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg
--no-default-keyring --keyring gnupg-
ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644
/usr/share/keyrings/wazuh.gpg
```



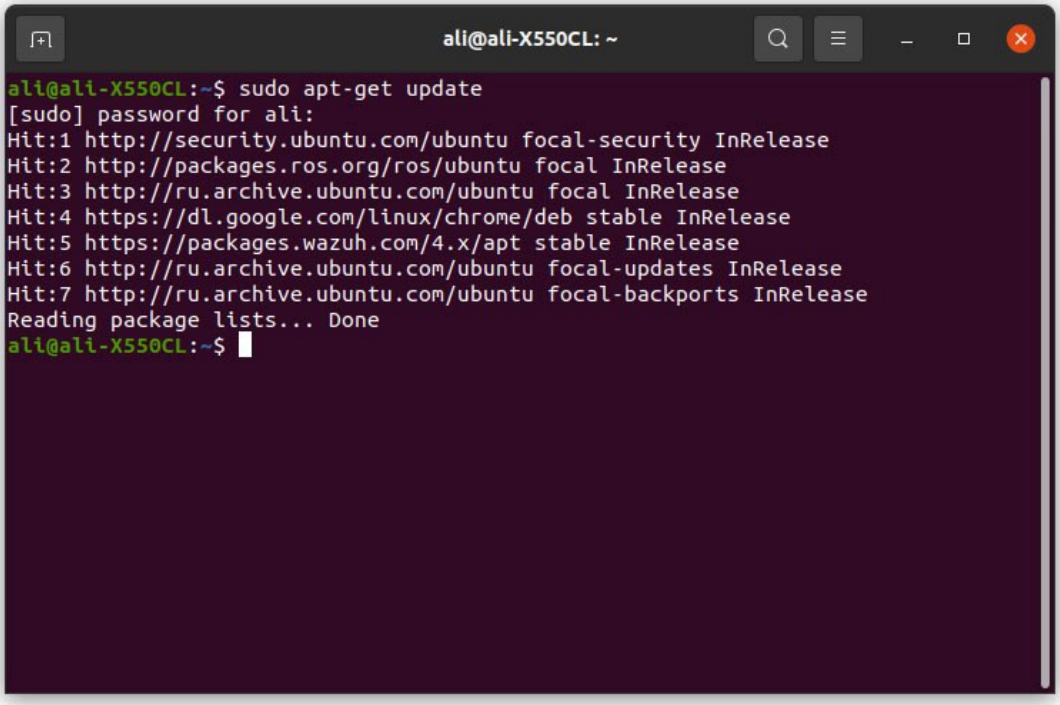
- Add the repository:

```
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg]
https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
/etc/apt/sources.list.d/wazuh.list
```



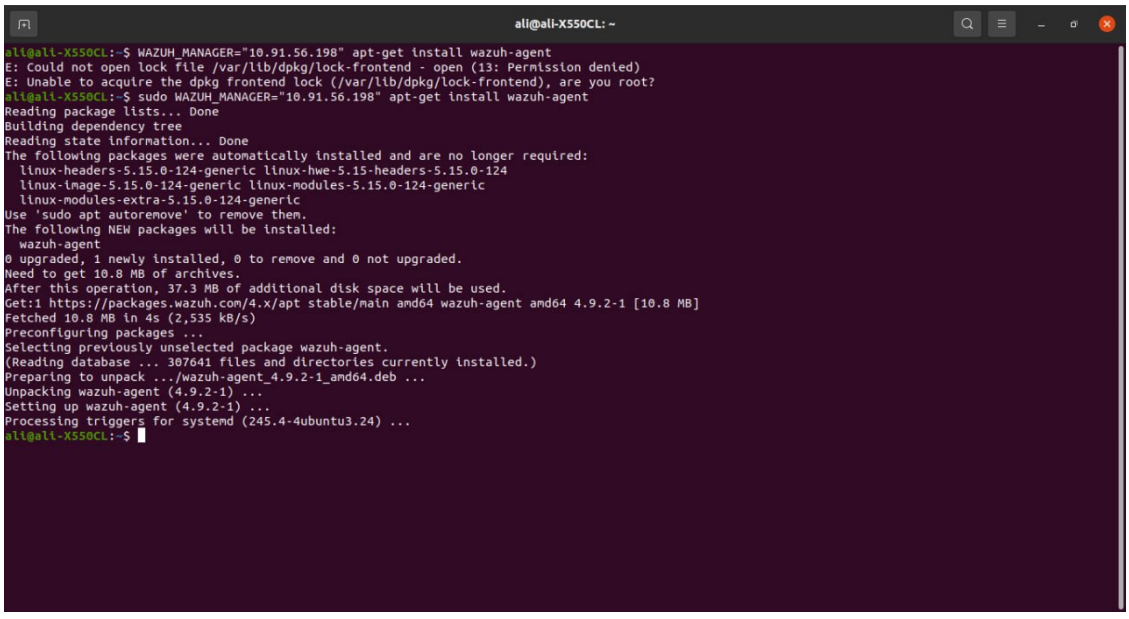
- Update the package information:

```
sudo apt-get update
```



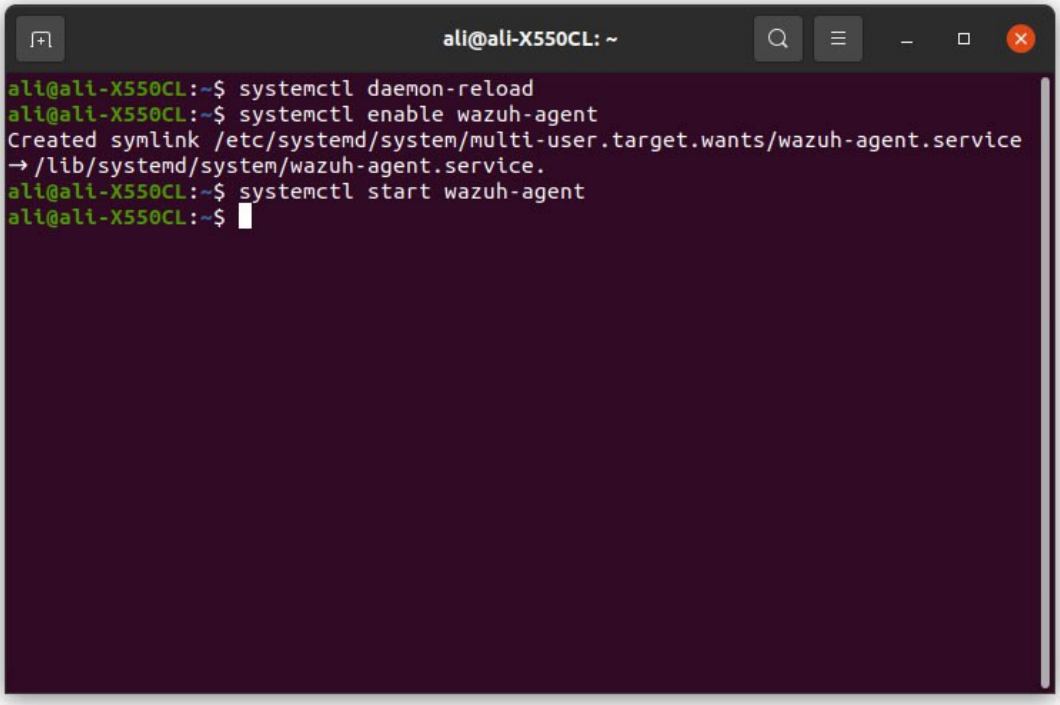
- Edit the WAZUH_MANAGER variable to contain our Wazuh manager IP address or hostname `10.91.56.198`:

```
WAZUH_MANAGER="10.91.56.198" apt-get install wazuh-agent
```



- Enable and start the Wazuh agent service:

```
systemctl daemon-reload
systemctl enable wazuh-agent
systemctl start wazuh-agent
```



3. Setup IRIS on Ubuntu 22.04 (Ammar Device):

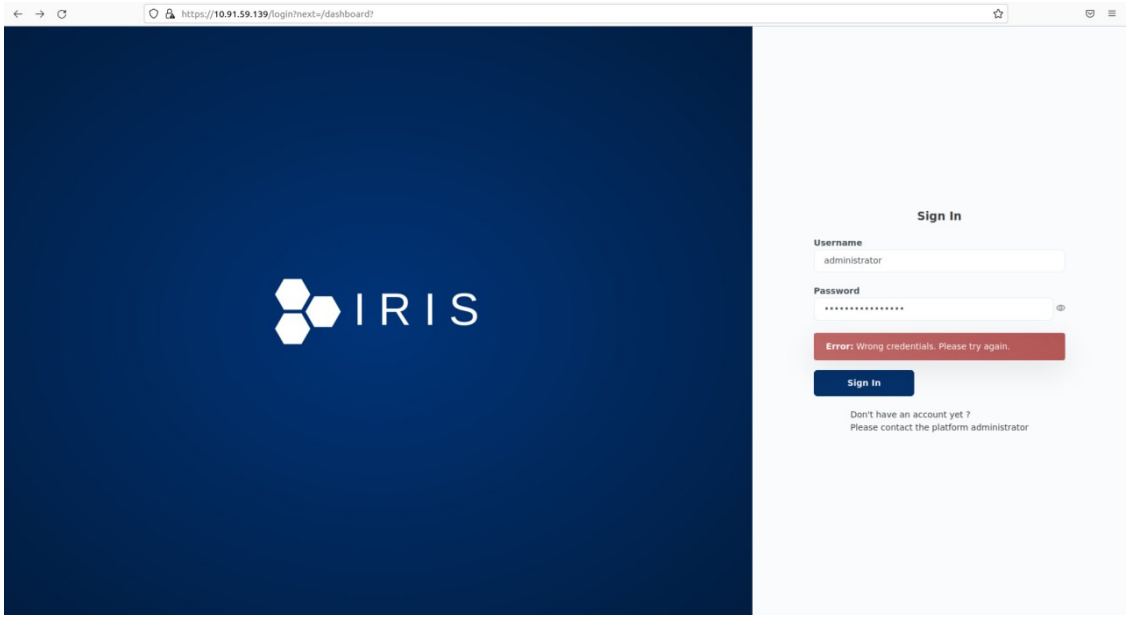
   - Setup Docker on the device:

```
(DOCKER PART)

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo docker run hello-world
sudo groupadd docker
sudo gpasswd -a $USER docker
newgrp docker
docker run hello-world
```

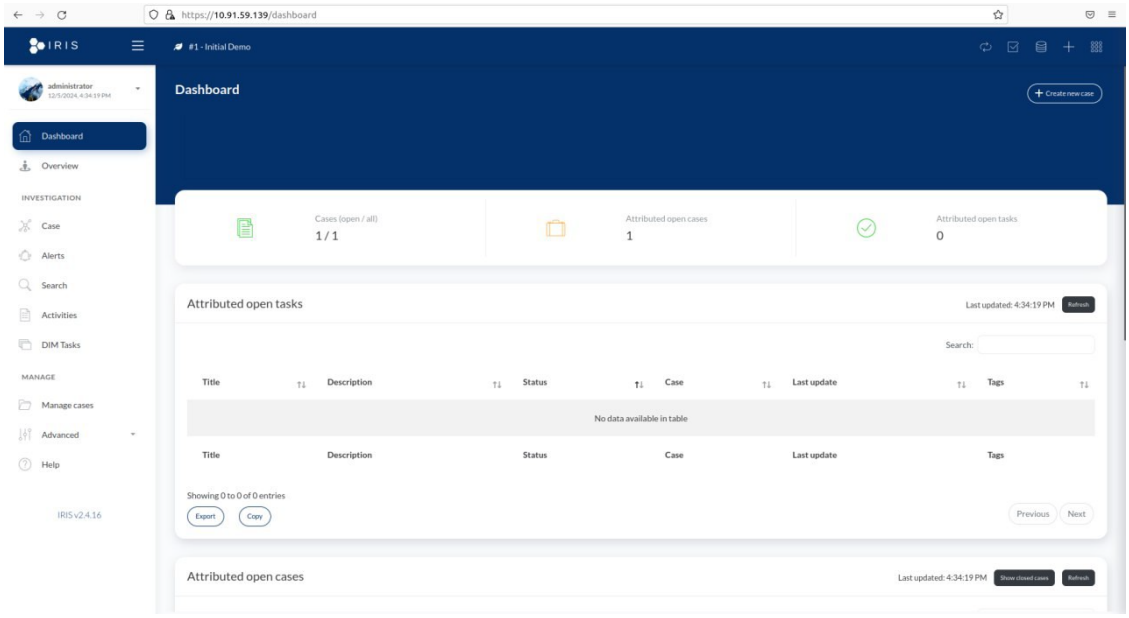
   - Intalling the IRIS on the device:

```
#  Clone the iris-web repository
git clone https://github.com/dfir-iris/iris-web.git
cd iris-web

# Checkout to the last tagged version
git checkout v2.4.16

# Copy the environment file
cp .env.model .env

# Pull the dockers
docker compose pull

# Run IRIS
docker compose up
```

- After Installing, Access the web interface:



- Login and see the Dashboard of IRIS:

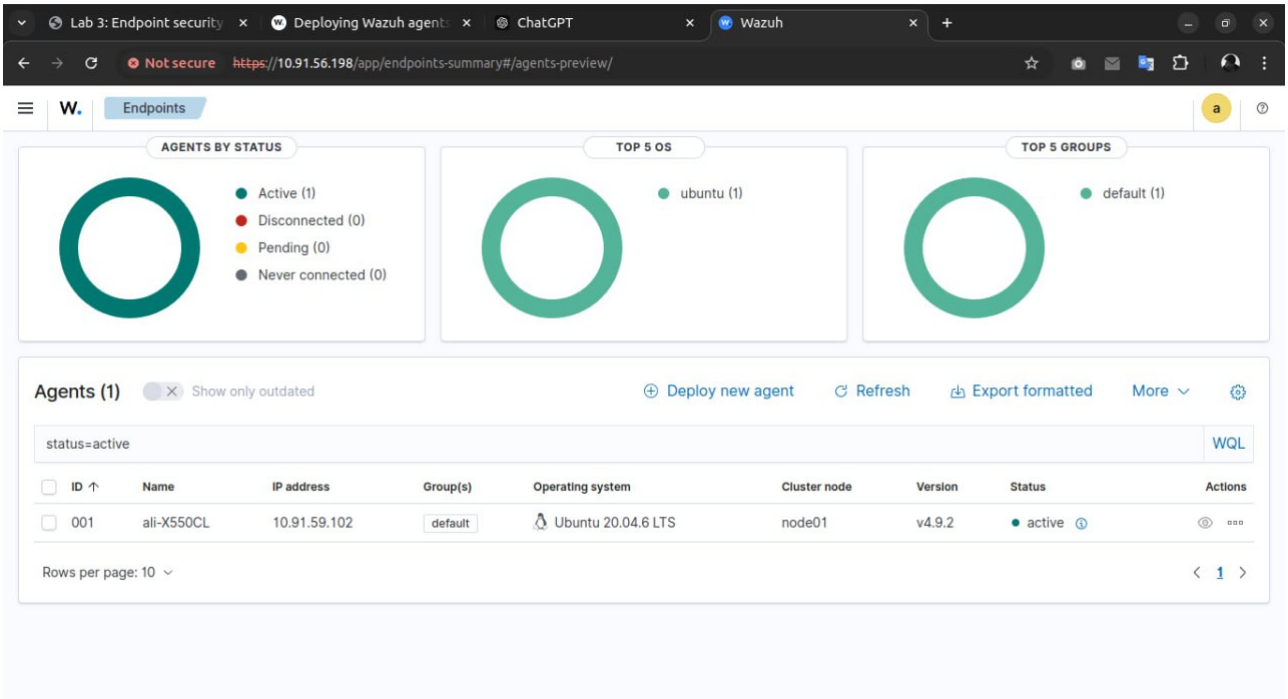


- Network architecture with diagrams showing relationships between components.

Working Instances of Tools

- **Wazuh:** Configured to collect logs, generate alerts, and display dashboards.

  here you can see the connection between the Wazuh Server and Wazuh Agent



- **MISP:** Integrated with Wazuh for threat intelligence enrichment.
  - Clone the repository of IRIS-Wazuh integration and update `verify=false` inside the script `custom-iris.py`.
  - Copy the updated script to `custom-iris.py` to the directory `/var/ossec/integrations/custom-iris.py`, Sets permissions for the file so the owner can read, write, and execute, and Changes the ownership of the file to the user root and group wazuh.

```
git clone https://github.com/nateuribe/Wazuh-IRIS-integration.git
cd Wazuh-IRIS-integration/
cp custom-iris.py /var/ossec/integrations/custom-iris.py
chmod 750 /var/ossec/integrations/custom-iris.py
chown root:wazuh /var/ossec/integrations/custom-iris.py
```

```
78 # Send request to IRIS
79 response = requests.post(hook_url, verify=False, data=payload, headers={"Authorization": "Bearer " + api_key, "content-type": "application/json"})
80
```

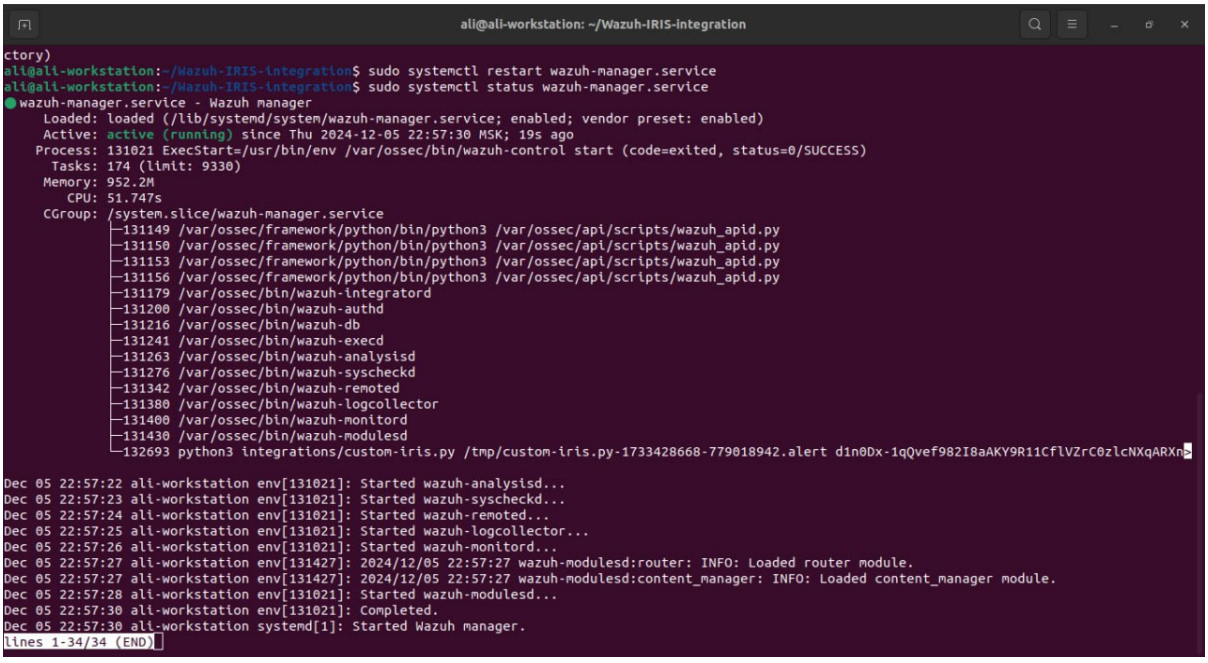
  - Add the following snippet into the `/var/ossec/etc/ossec.conf` config file, Adjust `<hook_url>` and `<api_key>` to your environment, and change `<level>` to the desired threshold for alerts.

```
<!-- ... Rest of config ... -->
    <!-- IRIS integration -->
```

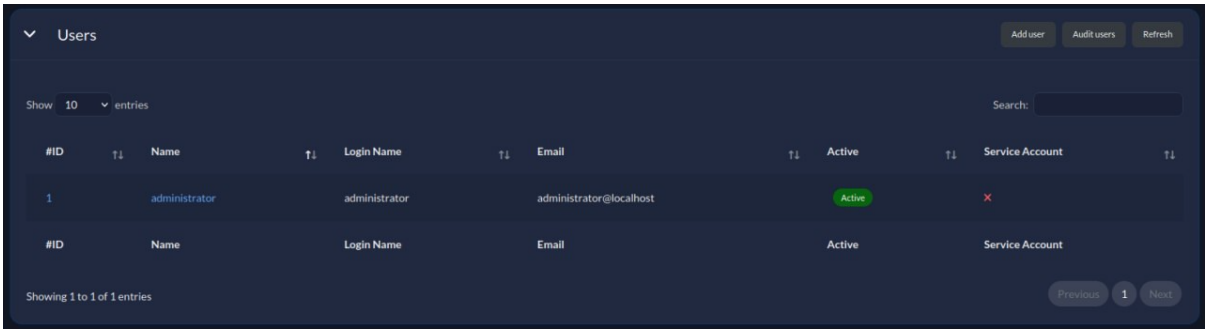
```
    <integration>
      <name>custom-iris.py</name>
      <hook_url>http://10.91.59.139/alerts/add</hook_url>
      <level>4</level>
      <api_key>d1n0Dx-1qQvef982I8aAKY9R11CflVZrC0zlcNXqARXn5-
IdcOUNHZSiPbgBTStzmDDSGYw6Z8m3_KZ8p1P5Ww</api_key>
      <alert_format>json</alert_format>
    </integration>


<!-- ... Rest of config ... -->
```

```
<!-- IRIS integration -->
  <integration>
  <name>custom-iris.py</name>
  <hook_url>https://10.91.59.139/alerts/add</hook_url>
  <level>4</level>
  <api_key>d1n0Dx-1qQvef982I8aAKY9R11CflVZrC0zlcNXqARXn5-
IdcOUNHZSiPbgBTStzmDDSGYw6Z8m3_KZ8p1P5Ww</api_key>
  <alert_format>json</alert_format>
  </integration>
```

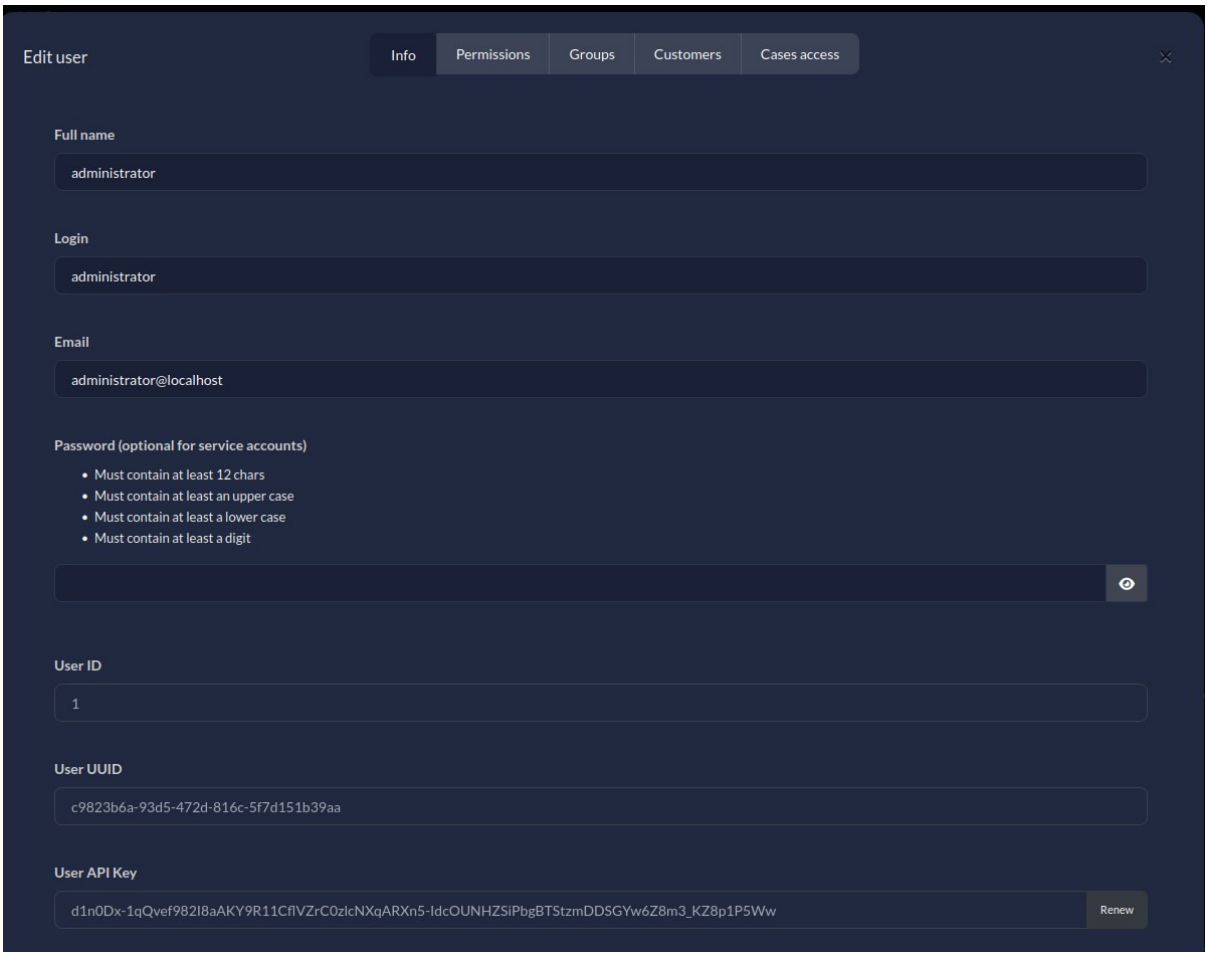
- Restart the wazuh-manager service after making the above settings, and check the status:



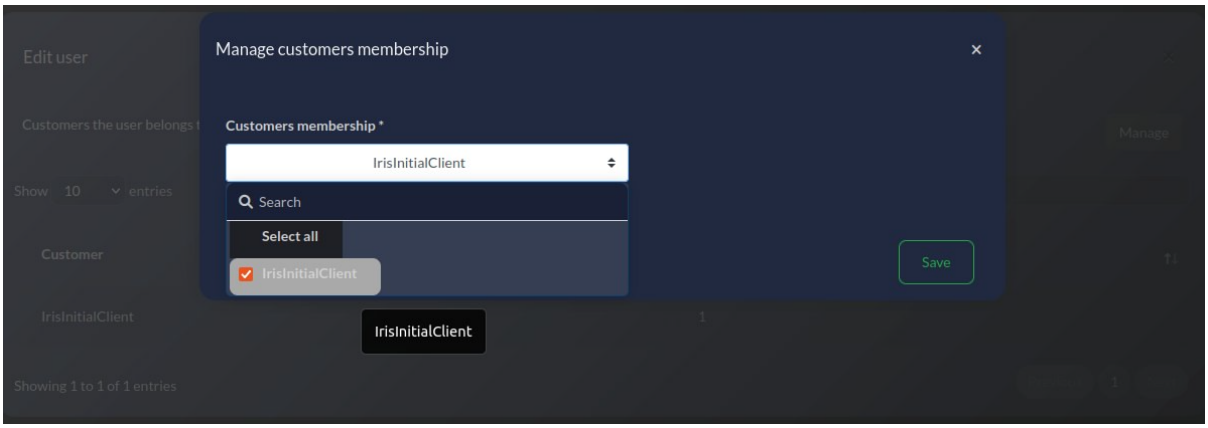
- Now eveything configured from Wazuh server side and we need to check it on the IRIS side:
- Navigate to the Advanced Settings in Iris, then Access Control Section:



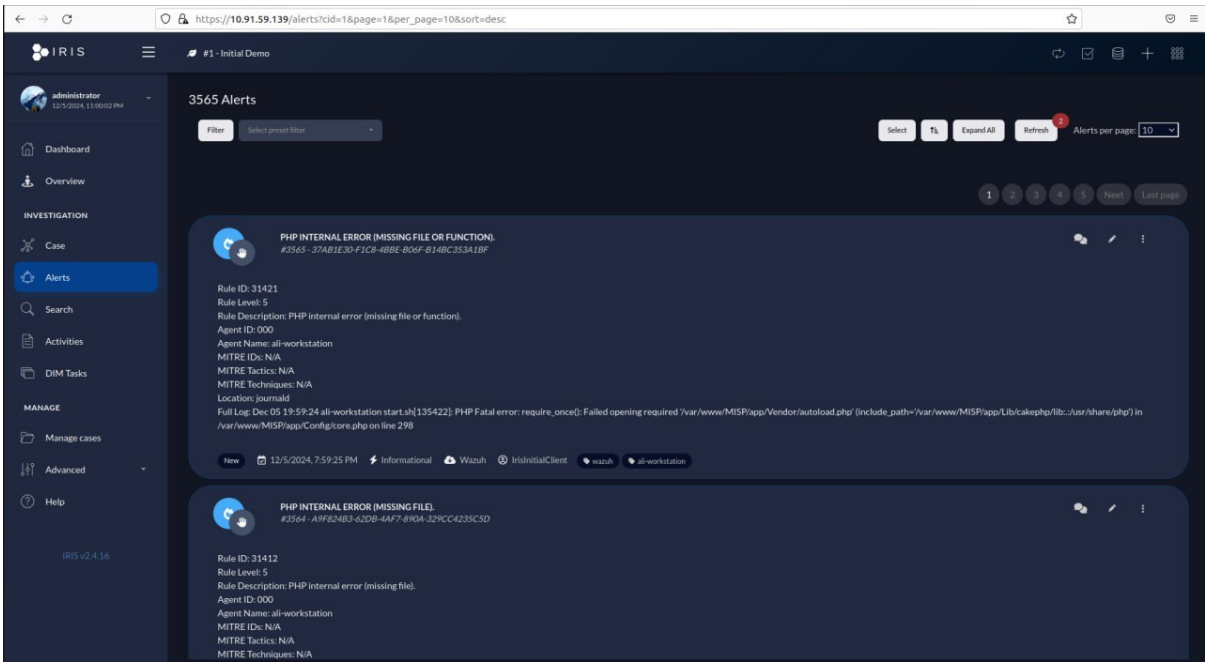
- Locate Customers Management:



- Assign Wazuh Server to a Customer:



- Showing the alerts of thats coming from wazuh agent:



- **IRIS:** Configured for case management and linked with Wazuh.

Testing Scenarios

- **Incident 1:** Brute force attack simulation detected by Wazuh, automated IP block as response.
- **Incident 2:** Malware detection with alert escalation to IRIS.

Test Results

- Logs or screenshots demonstrating successful detection and automated responses.

---

4. Difficulties Faced and New Skills Acquired

Difficulties Faced

- Challenges in tool integration and resolving API incompatibilities.
- Troubleshooting network configurations and performance issues.

New Skills Acquired

- Hands-on experience with Wazuh, MISP, and IRIS.
- Skills in integrating SOC components and automating responses.
- Enhanced understanding of incident workflows and response automation.

---

5. Conclusion, Contemplations, and Judgement

Conclusion

- Evaluation of the SOC setup's effectiveness in detecting and responding to incidents.
- Reflection on the strengths of the open-source tools used.
- Acknowledgment of the limitations and areas for improvement.

Recommendations

- Suggestions for future work, such as adding more tools or refining workflows.
- Potential use cases for scaling the solution in real-world scenarios.

---

6. Appendices

A. Scripts and Configurations

- Links to all scripts and configuration files in the repository:
  GitHub Repository

B. Proof of Concept Demonstration Video

- YouTube Link to Demo Video

C. Extended Documentation

- Step-by-step installation and configuration details for all tools.
- Troubleshooting steps for resolving challenges.

Integration of New Sections with Previous Requirements:

This report structure includes:

1. **Goal and Tasks of the Project:** Describes what the project aims to solve and specifies the division of tasks among members.
2. **Execution Plan/Methodology:** Contains the solution plan, architecture diagram, and methodology for implementing and testing the solution.
3. **Development of Solution/Tests:** Explains the setup and PoC testing, including results and insights from incident simulations.
4. **Difficulties Faced/New Skills:** Reflects on challenges and the knowledge gained during the project.
5. **Conclusion and Judgement:** Wraps up with reflections, evaluations, and suggestions for improvement.
6. **Appendices:** Contains all important links to code, configuration files, and demonstration materials for proof of concept.

Let me know if you need help with any specific section or step in implementation!