

# 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), AbusIPDB (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 & Mohammad Jaafar:** 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., AbusIPDB, IRIS).
- Writing the readme section and overall evaluation of contents and reports.

#### 2. **Yehia Sobeh:** Integration of AbusIPDB for threat intelligence.

- Install and configure AbusIPDB as the threat intelligence platform.
- Integrate AbusIPDB 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 AbusIPDB.

#### • **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, AbusIPDB, and IRIS.
- Establish network connectivity and API integrations.

#### 2. Integration:

- Configure Wazuh to collect logs and generate alerts.
- Integrate AbusIPDB 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.

### 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 -s0 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.

```

lialli-workstation: ~ $ sudo curl -s0 https://packages.wazuh.com/4.9/wazuh-install.sh && sudo bash ./wazuh-install.sh -a
05/12/2024 14:35:01 INFO: Starting Wazuh Installation assistant. Wazuh version: 4.9.2
05/12/2024 14:35:01 INFO: Verbose logging redirected to /var/log/wazuh-install.log
05/12/2024 14:35:15 INFO: Verifying that your system meets the recommended minimum hardware requirements.
05/12/2024 14:35:15 INFO: Wazuh web interface port will be 443.
05/12/2024 14:35:34 INFO: Wazuh repository added.
05/12/2024 14:35:34 INFO: --- Configuration files ---
05/12/2024 14:35:34 INFO: Generating configuration files.
05/12/2024 14:35:35 INFO: Generating the root certificate.
05/12/2024 14:35:36 INFO: Generating Admin certificates.
05/12/2024 14:35:36 INFO: Generating Wazuh indexer certificates.
05/12/2024 14:35:36 INFO: Generating Filebeat certificates.
05/12/2024 14:35:36 INFO: Generating Wazuh dashboard certificates.
05/12/2024 14:35:37 INFO: Created wazuh-install-files.tar. It contains the Wazuh cluster key, certificates, and passwords necessary for installation.
05/12/2024 14:35:37 INFO: --- Wazuh Indexer ---
05/12/2024 14:39:28 INFO: Wazuh Indexer installation finished.
05/12/2024 14:39:28 INFO: Wazuh Indexer post-install configuration finished.
05/12/2024 14:39:28 INFO: Starting service wazuh-indexer.
05/12/2024 14:39:47 INFO: wazuh-indexer service started.
05/12/2024 14:39:47 INFO: Initializing Wazuh indexer cluster security settings.
05/12/2024 14:39:53 INFO: Wazuh indexer cluster security configuration initialized.
05/12/2024 14:39:53 INFO: Wazuh indexer cluster initialized.
05/12/2024 14:39:53 INFO: --- Wazuh server ---
05/12/2024 14:39:53 INFO: Starting the Wazuh manager installation.
05/12/2024 14:42:28 INFO: Wazuh manager installation finished.
05/12/2024 14:42:28 INFO: Wazuh manager vulnerability detection configuration finished.
05/12/2024 14:42:28 INFO: Starting service wazuh-manager.
05/12/2024 14:42:46 INFO: wazuh-manager service started.
05/12/2024 14:42:46 INFO: Starting Filebeat installation.
05/12/2024 14:43:21 INFO: Filebeat installation finished.
05/12/2024 14:43:26 INFO: Filebeat post-install configuration finished.
05/12/2024 14:43:26 INFO: Starting service filebeat.
05/12/2024 14:43:28 INFO: filebeat service started.
05/12/2024 14:43:28 INFO: --- Wazuh dashboard ---

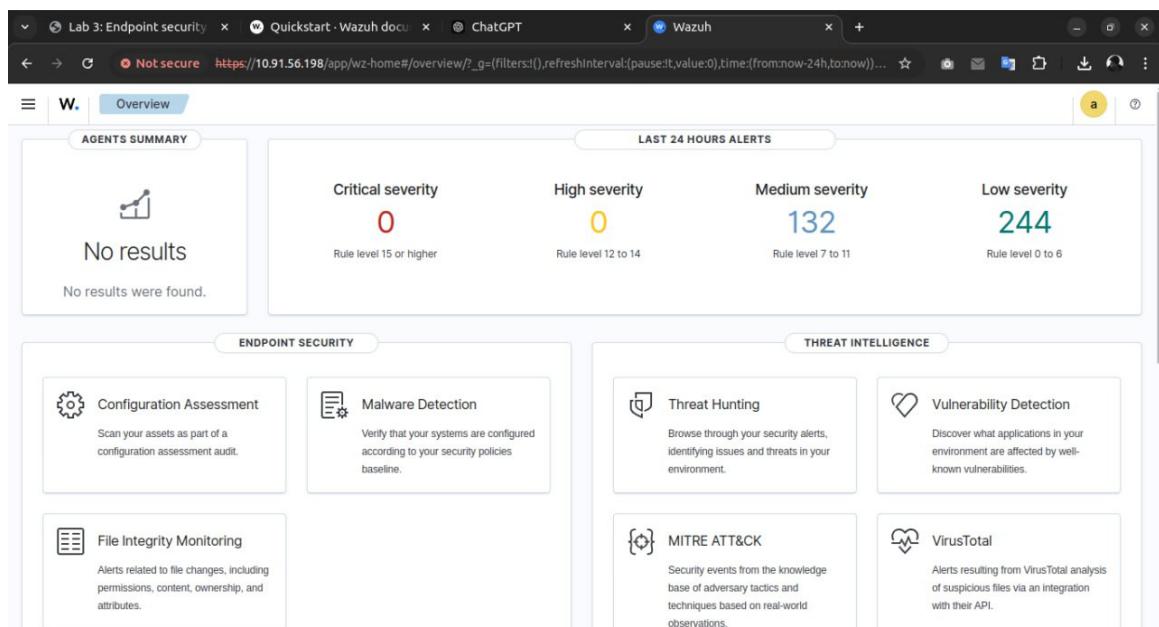
```

```

05/12/2024 14:43:28 INFO: --- Wazuh dashboard ---
05/12/2024 14:43:28 INFO: Starting Wazuh dashboard installation.
05/12/2024 14:45:33 INFO: Wazuh dashboard installation finished.
05/12/2024 14:45:33 INFO: Wazuh dashboard post-install configuration finished.
05/12/2024 14:45:33 INFO: Starting service wazuh-dashboard.
05/12/2024 14:45:34 INFO: wazuh-dashboard service started.
05/12/2024 14:45:36 INFO: Updating the internal users.
05/12/2024 14:45:42 INFO: A backup of the internal users has been saved in the /etc/wazuh-Indexer/internalusers-backup folder.
05/12/2024 14:45:54 INFO: The filebeat.yml file has been updated to use the Filebeat Keystore username and password.
05/12/2024 14:46:29 INFO: Initializing Wazuh dashboard web application.
05/12/2024 14:46:29 INFO: Wazuh dashboard web application not yet initialized. Waiting...
05/12/2024 14:46:45 INFO: Wazuh dashboard web application not yet initialized. Waiting...
05/12/2024 14:47:00 INFO: Wazuh dashboard web application initialized.
05/12/2024 14:47:00 INFO: --- Summary ---
05/12/2024 14:47:00 INFO: You can access the web interface https://<wazuh-dashboard-ip>:443
    User: admin
    Password: scZ.E3BKusUFJ8ZqKfTiyrxH4CRkLBL+
05/12/2024 14:47:00 INFO: Installation finished.
lialli-workstation: ~

```

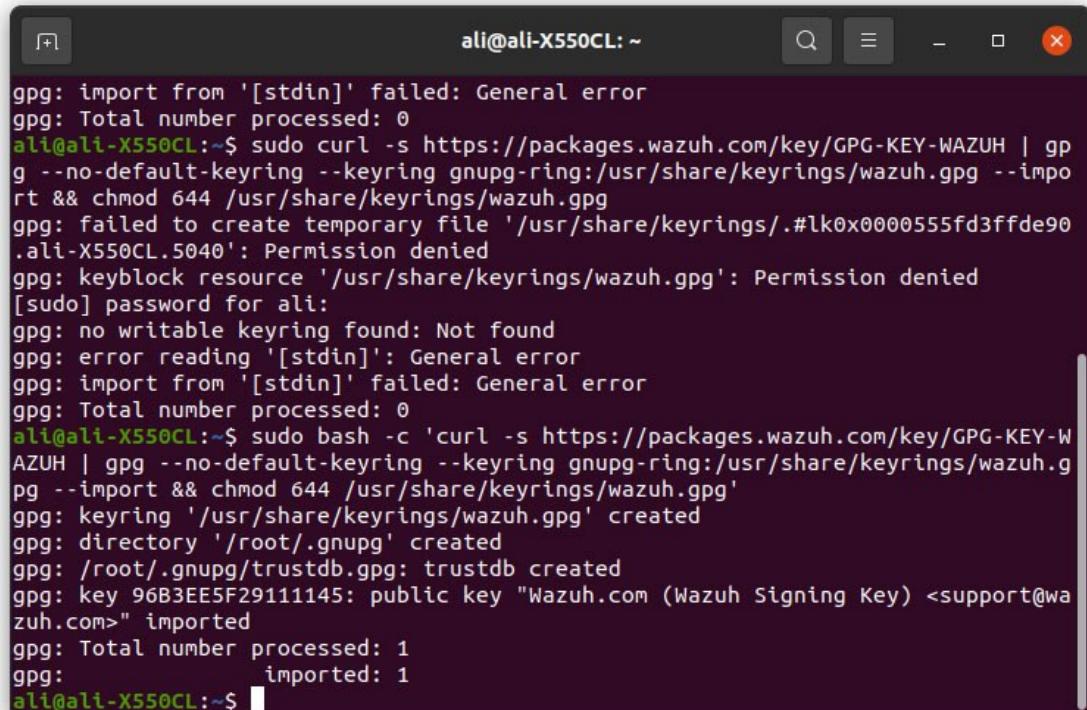
- Access the Wazuh web interface with <https://10.91.56.198:443> and my credentials:



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

- Install the GPG key:

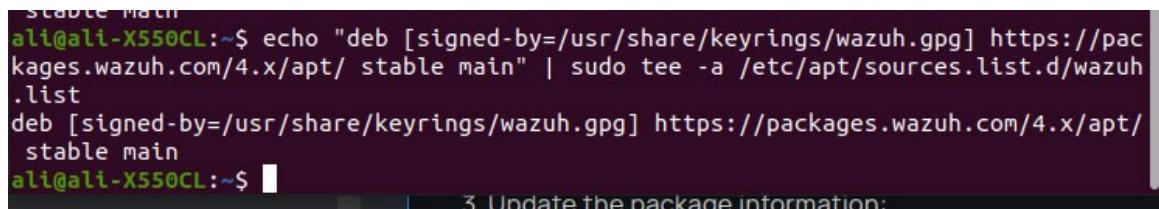
```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/share/keyrings/wazuh.gpg
```



```
ali@ali-X550CL:~$ 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
gpg: failed to create temporary file '/usr/share/keyrings/.#lk0x0000555fd3ffde90.ali-X550CL.5040': Permission denied
gpg: keyblock resource '/usr/share/keyrings/wazuh.gpg': Permission denied
[sudo] password for ali:
gpg: no writable keyring found: Not found
gpg: error reading '[stdin]': General error
gpg: import from '[stdin]' failed: General error
gpg: Total number processed: 0
ali@ali-X550CL:~$ sudo bash -c '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'
gpg: keyring '/usr/share/keyrings/wazuh.gpg' created
gpg: directory '/root/.gnupg' created
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 96B3EE5F29111145: public key "Wazuh.com (Wazuh Signing Key) <support@wazuh.com>" imported
gpg: Total number processed: 1
gpg: imported: 1
ali@ali-X550CL:~$
```

- Add the repository:

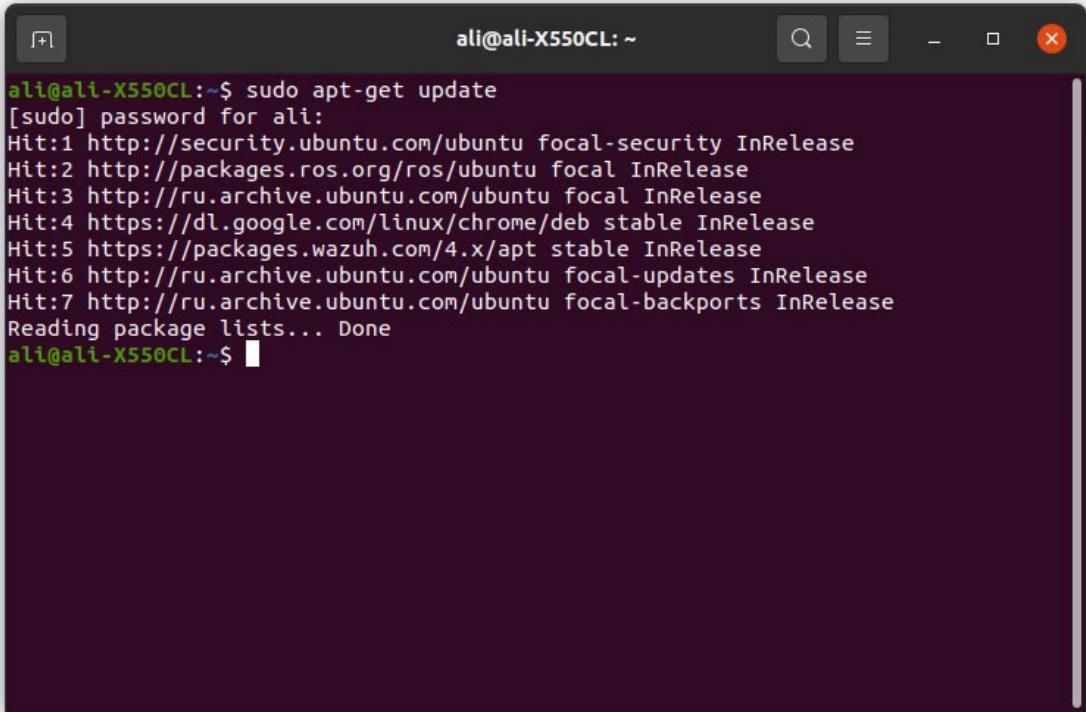
```
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg]
https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
/etc/apt/sources.list.d/wazuh.list
```



```
ali@ali-X550CL:~$ echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee -a /etc/apt/sources.list.d/wazuh.list
deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/
stable main
ali@ali-X550CL:~$
```

- Update the package information:

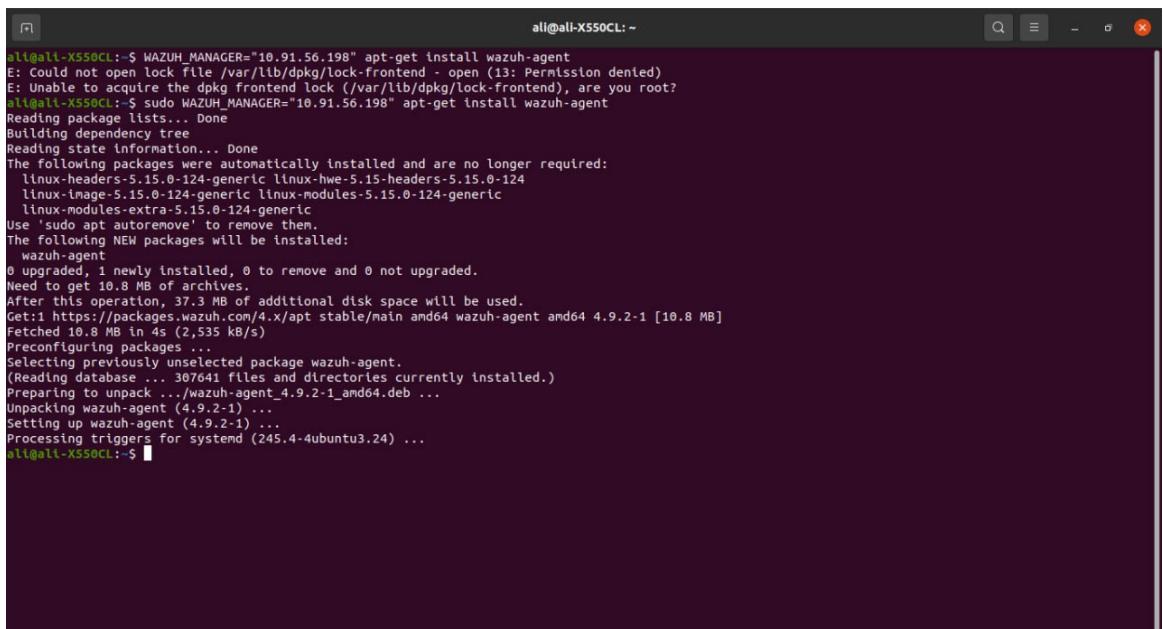
```
sudo apt-get update
```



```
ali@ali-X550CL:~$ sudo apt-get update
[sudo] password for ali:
Hit:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:2 http://packages.ros.org/ros/ubuntu focal InRelease
Hit:3 http://ru.archive.ubuntu.com/ubuntu focal InRelease
Hit:4 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:5 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:6 http://ru.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://ru.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
ali@ali-X550CL:~$
```

- Edit the WAZUH\_MANAGER variable to contain our Wazuh manager IP address or hostname **10.91.56.198**:

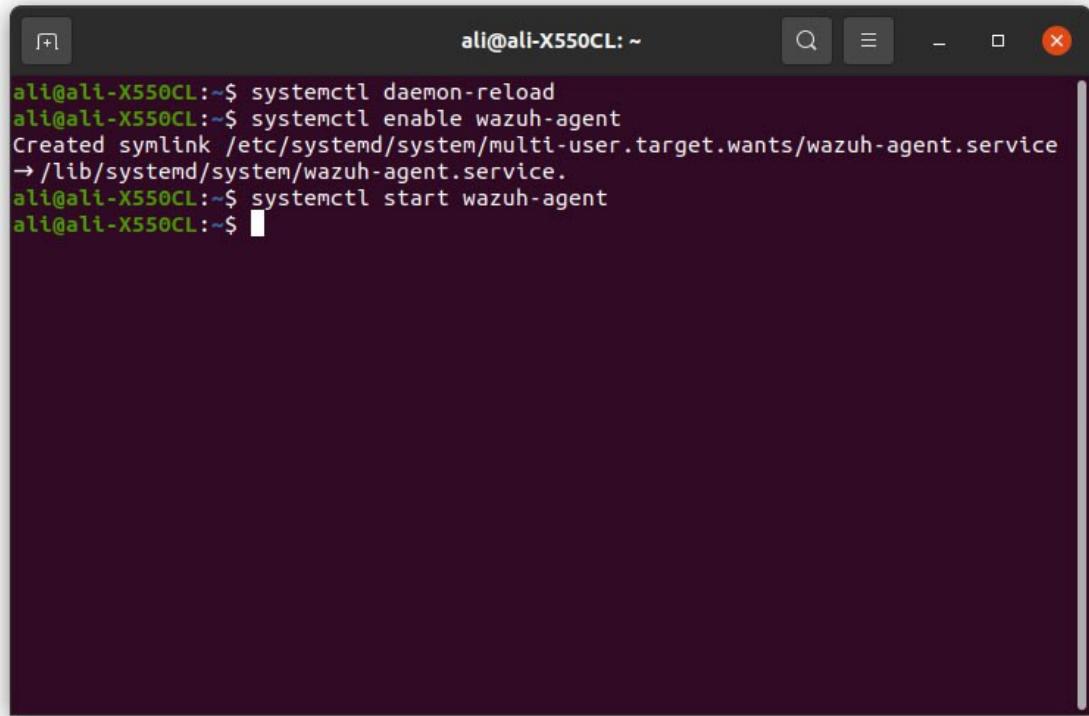
```
WAZUH_MANAGER="10.91.56.198" apt-get install wazuh-agent
```



```
ali@ali-X550CL:~$ WAZUH_MANAGER="10.91.56.198" apt-get install wazuh-agent
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
ali@ali-X550CL:~$ sudo WAZUH_MANAGER="10.91.56.198" apt-get install wazuh-agent
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-5.15.0-124-generic linux-hwe-5.15-headers-5.15.0-124
  linux-image-5.15.0-124-generic linux-modules-5.15.0-124-generic
  linux-modules-extra-5.15.0-124-generic
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  wazuh-agent
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 10.8 MB of archives.
After this operation, 37.3 MB of additional disk space will be used.
Get:1 https://packages.wazuh.com/4.x/apt stable/main amd64 wazuh-agent amd64 4.9.2-1 [10.8 MB]
Fetched 10.8 MB in 4s (2,535 kB/s)
Preconfiguring packages ...
Selecting previously unselected package wazuh-agent.
(Reading database ... 307641 files and directories currently installed.)
Preparing to unpack .../wazuh-agent_4.9.2-1_amd64.deb ...
Unpacking wazuh-agent (4.9.2-1) ...
Setting up wazuh-agent (4.9.2-1) ...
Processing triggers for systemd (245.4-4ubuntu3.24) ...
ali@ali-X550CL:~$
```

- Enable and start the Wazuh agent service:

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



The screenshot shows a terminal window titled "ali@ali-X550CL: ~". The user has run several commands to manage a Wazuh agent service:

```
ali@ali-X550CL:~$ systemctl daemon-reload
ali@ali-X550CL:~$ systemctl enable wazuh-agent
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-agent.service
→ /lib/systemd/system/wazuh-agent.service.
ali@ali-X550CL:~$ systemctl start wazuh-agent
ali@ali-X550CL:~$
```

### 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

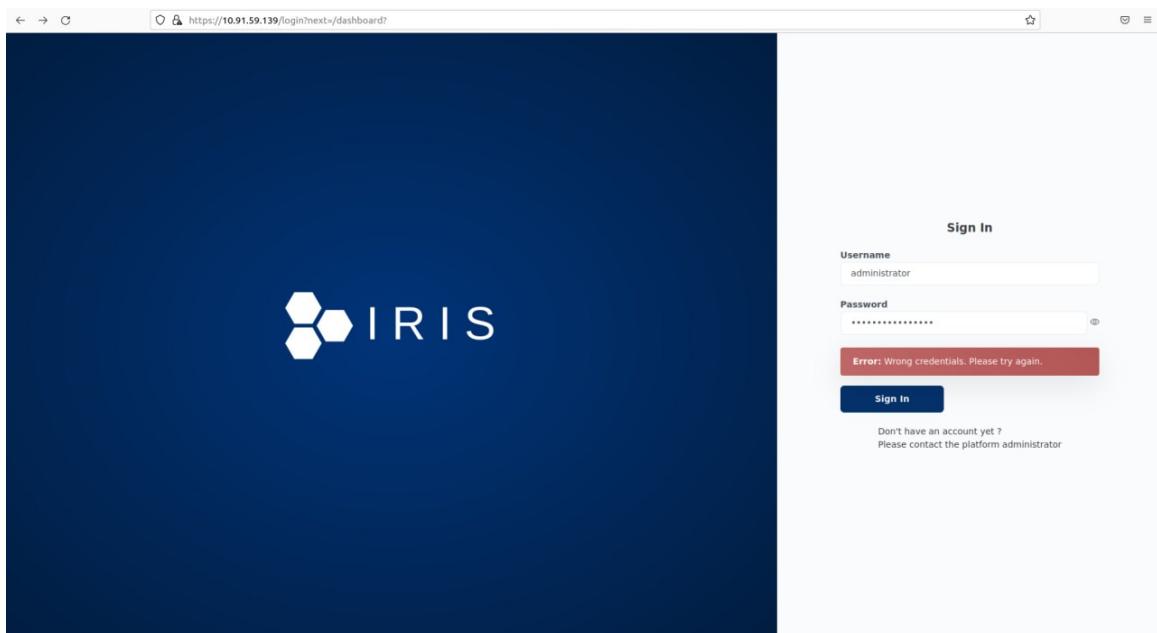
# Copy the environment file
cp .env.model .env

# Pull the docker
docker compose pull

# Run IRIS
docker compose up

```

- After Installing, Access the web interface:

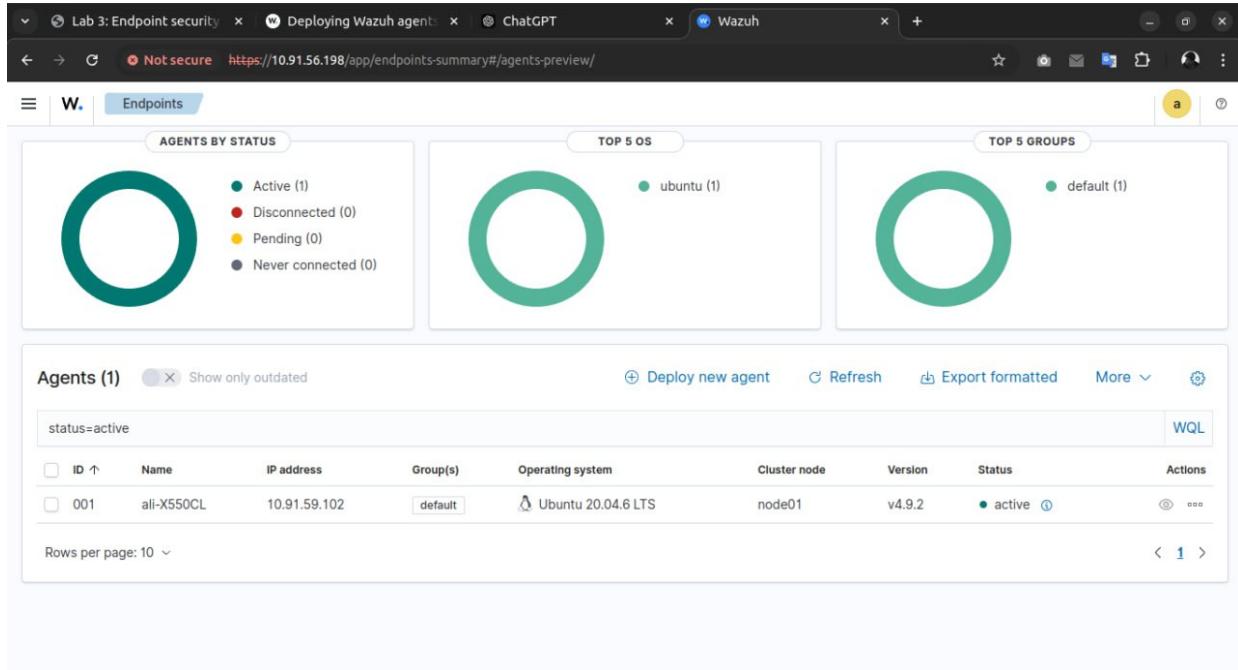


- Login and see the Dashboard of IRIS:

## 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:



- **AbusIPDB:** Integrated with Wazuh for threat intelligence enrichment.

- First I registered in AbusIPDB and creat a API key.
- Then we follow this [documentaion](#) to configure wazuh to send request to check if an IP address is exist as a malicious.

### steps

1. We add rules into var/ossec/etc/rules/local\_rules.xml for trigger alert when using public ip and rules to accept the answer from AbusIPDB but we do it from the dashboard

The screenshot shows the Wazuh dashboard with the local\_rules.xml file open in a code editor. The XML content includes rules for SSH authentication and abuse detection:

```
<group name="local.syslog.sshd">
  <rule id="100002" level="5">
    <if_sid>5715</if_sid>
    <match type="pcre2">\b(?!(10|192\.\d{2}\.\d{1}|168\.\d{2}\.\d{1}|172\.(2[0-9]|1[6-9]|3[0-1])|(25[6-9]|2[6-9]|2[6-9]|3[9-0]|3[9-0]|99[1-9]))[0-9]{1,3}\.((25[0-5]|2[0-4]|0[1-9])?[0-9]{0-9})\.(25[0-5]|2[0-4]|0[1-9])?[0-9]{0-9}\.((25[0-5]|2[0-4]|0[1-9])?[0-9]{0-9})</match>
    <description><sshd: Authentication failed from a public IP address $srcip.></description>
  </rule>
  <group>authentication_failed,authentication_success,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>
<rule id="100003" level="5">
  <if_sid>5715</if_sid>
  <match type="pcre2">\b(?!(10|192\.\d{2}\.\d{1}|168\.\d{2}\.\d{1}|172\.(2[0-9]|1[6-9]|3[0-1])|(25[6-9]|2[6-9]|2[6-9]|3[9-0]|3[9-0]|99[1-9]))[0-9]{1,3}\.((25[0-5]|2[0-4]|0[1-9])?[0-9]{0-9})\.(25[0-5]|2[0-4]|0[1-9])?[0-9]{0-9}\.((25[0-5]|2[0-4]|0[1-9])?[0-9]{0-9})</match>
  <description><sshd: Authentication succeeded from a public IP address $srcip.></description>
  </rule>
  <rule id="100004" level="10">
    <field name="abuseipdb_source_rule" type="pcre2">>^100002$</field>
    <if_sid name="abuseipdb_abuse_confidence_score" type="pcre2" negate="yes">>^0\$</if_sid>
    <description>AbuseIPDB: SSH Authentication failed from a public IP address $(abuseipdb.source.srcip) with $(abuseipdb.abuse_confidence_score)% confidence of abuse.</description>
    </rule>
    <rule id="100005" level="14">
      <field name="abuseipdb_source_rule" type="pcre2">>^100003$</field>
      <if_sid name="abuseipdb_abuse_confidence_score" type="pcre2" negate="yes">>^0\$</if_sid>
      <description>AbuseIPDB: SSH Authentication succeeded from a public IP address $(abuseipdb.source.srcip) with $(abuseipdb.abuse_confidence_score)% confidence of abuse.</description>
      </rule>
    </group>
  </group>
</group>
```

## 2. we create the custom-abuseipdb.py in </var/ossec/integrations/custom-abuseipdb.py>

```
#!/var/ossec/framework/python/bin/python3
# Copyright (C) 2015-2022, Wazuh Inc.
import json
import sys
import time
import os
from socket import socket, AF_UNIX, SOCK_DGRAM
try:
    import requests
    from requests.auth import HTTPBasicAuth
except Exception as e:
    print("No module 'requests' found. Install: pip install requests")
    sys.exit(1)
# Global vars
debug_enabled = False
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
json_alert = {}
now = time.strftime("%a %b %d %H:%M:%S %Z %Y")
# Set paths
log_file = '{0}/logs/integrations.log'.format(pwd)
socket_addr = '{0}/queue/sockets/queue'.format(pwd)
def main(args):
    debug("# Starting")
    # Read args
    alert_file_location = args[1]
    apikey = args[2]
    debug("# API Key")
    debug(apikey)

```

```
#!/var/ossec/framework/python/bin/python3
# Copyright (C) 2015-2022, Wazuh Inc.

import json
import sys
import time
import os
from socket import socket, AF_UNIX, SOCK_DGRAM

try:
    import requests
    from requests.auth import HTTPBasicAuth
except Exception as e:
    print("No module 'requests' found. Install: pip install requests")
    sys.exit(1)

# Global vars

debug_enabled = False
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
json_alert = {}
now = time.strftime("%a %b %d %H:%M:%S %Z %Y")

# Set paths
log_file = '{0}/logs/integrations.log'.format(pwd)
socket_addr = '{0}/queue/sockets/queue'.format(pwd)

def main(args):
    debug("# Starting")

    # Read args
    alert_file_location = args[1]
    apikey = args[2]
```

```

        debug("# API Key")
        debug(apikey)

        debug("# File location")
        debug(alert_file_location)

        # Load alert. Parse JSON object.
        with open(alert_file_location) as alert_file:
            json_alert = json.load(alert_file)
        debug("# Processing alert")
        debug(json_alert)

        # Request AbuseIPDB info
        msg = request_abuseipdb_info(json_alert,apikey)

        # If positive match, send event to Wazuh Manager
        if msg:
            send_event(msg, json_alert["agent"])

def debug(msg):
    if debug_enabled:
        msg = "{0}: {1}\n".format(now, msg)

        print(msg)

        f = open(log_file,"a")
        f.write(msg)
        f.close()

def collect(data):
    abuse_confidence_score = data['abuseConfidenceScore']
    country_code = data['countryCode']
    usage_type = data['usageType']
    isp = data['isp']
    domain = data['domain']
    total_reports = data['totalReports']
    last_reported_at = data['lastReportedAt']
    return abuse_confidence_score, country_code, usage_type, isp, domain,
total_reports, last_reported_at

def in_database(data, srcip):
    result = data['totalReports']
    if result == 0:
        return False
    return True

def query_api(srcip, apikey):
    params = {'maxAgeInDays': '90', 'ipAddress': srcip,}
    headers = {
        "Accept-Encoding": "gzip, deflate",
        'Accept': 'application/json',
        "Key": apikey
    }

```

```

        response =
    requests.get('https://api.abuseipdb.com/api/v2/check', params=params,
headers=headers)
    if response.status_code == 200:
        json_response = response.json()
        data = json_response["data"]
        return data
    else:
        alert_output = {}
        alert_output["abuseipdb"] = {}
        alert_output["integration"] = "custom-abuseipdb"
        json_response = response.json()
        debug("# Error: The AbuseIPDB encountered an error")
        alert_output["abuseipdb"]["error"] = response.status_code
        alert_output["abuseipdb"]["description"] = json_response["errors"]
[0]["detail"]
        send_event(alert_output)
        exit(0)

def request_abuseipdb_info(alert, apikey):
    alert_output = {}
    # If there is no source ip address present in the alert. Exit.
    if not "srcip" in alert["data"]:

        return(0)

    # Request info using AbuseIPDB API
    data = query_api(alert["data"]["srcip"], apikey)

    # Create alert
    alert_output["abuseipdb"] = {}
    alert_output["integration"] = "custom-abuseipdb"
    alert_output["abuseipdb"]["found"] = 0
    alert_output["abuseipdb"]["source"] = {}
    alert_output["abuseipdb"]["source"]["alert_id"] = alert["id"]
    alert_output["abuseipdb"]["source"]["rule"] = alert["rule"]["id"]
    alert_output["abuseipdb"]["source"]["description"] = alert["rule"]
["description"]
    alert_output["abuseipdb"]["source"]["full_log"] = alert["full_log"]
    alert_output["abuseipdb"]["source"]["srcip"] = alert["data"]["srcip"]
    srcip = alert["data"]["srcip"]
    # Check if AbuseIPDB has any info about the srcip
    if in_database(data, srcip):
        alert_output["abuseipdb"]["found"] = 1

    # Info about the IP found in AbuseIPDB
    if alert_output["abuseipdb"]["found"] == 1:
        abuse_confidence_score, country_code, usage_type, isp, domain,
total_reports, last_reported_at = collect(data)

        # Populate JSON Output object with AbuseIPDB request
    alert_output["abuseipdb"]["abuse_confidence_score"] =

```

```

abuse_confidence_score
    alert_output["abuseipdb"]["country_code"] = country_code
    alert_output["abuseipdb"]["usage_type"] = usage_type
    alert_output["abuseipdb"]["isp"] = isp
    alert_output["abuseipdb"]["domain"] = domain
    alert_output["abuseipdb"]["total_reports"] = total_reports
    alert_output["abuseipdb"]["last_reported_at"] = last_reported_at

    debug(alert_output)

    return(alert_output)

def send_event(msg, agent = None):
    if not agent or agent["id"] == "000":
        string = '1:abuseipdb:{0}'.format(json.dumps(msg))
    else:
        string = '1:[{0}] ({1}) {2}->abuseipdb:{3}'.format(agent["id"],
agent["name"], agent["ip"] if "ip" in agent else "any", json.dumps(msg))

    debug(string)
    sock = socket(AF_UNIX, SOCK_DGRAM)
    sock.connect(socket_addr)
    sock.send(string.encode())
    sock.close()

if __name__ == "__main__":
    try:
        # Read arguments
        bad_arguments = False
        if len(sys.argv) >= 4:
            msg = '{0} {1} {2} {3} {4}'.format(now, sys.argv[1],
sys.argv[2], sys.argv[3], sys.argv[4] if len(sys.argv) > 4 else '')
            debug_enabled = (len(sys.argv) > 4 and sys.argv[4] ==
'debug')
        else:
            msg = '{0} Wrong arguments'.format(now)
            bad_arguments = True

        # Logging the call
        f = open(log_file, 'a')
        f.write(msg + '\n')
        f.close()

        if bad_arguments:
            debug("# Exiting: Bad arguments.")
            sys.exit(1)

        # Main function
        main(sys.argv)

    except Exception as e:
        debug(str(e))
        raise

```

### 3. we give the script the right permissions

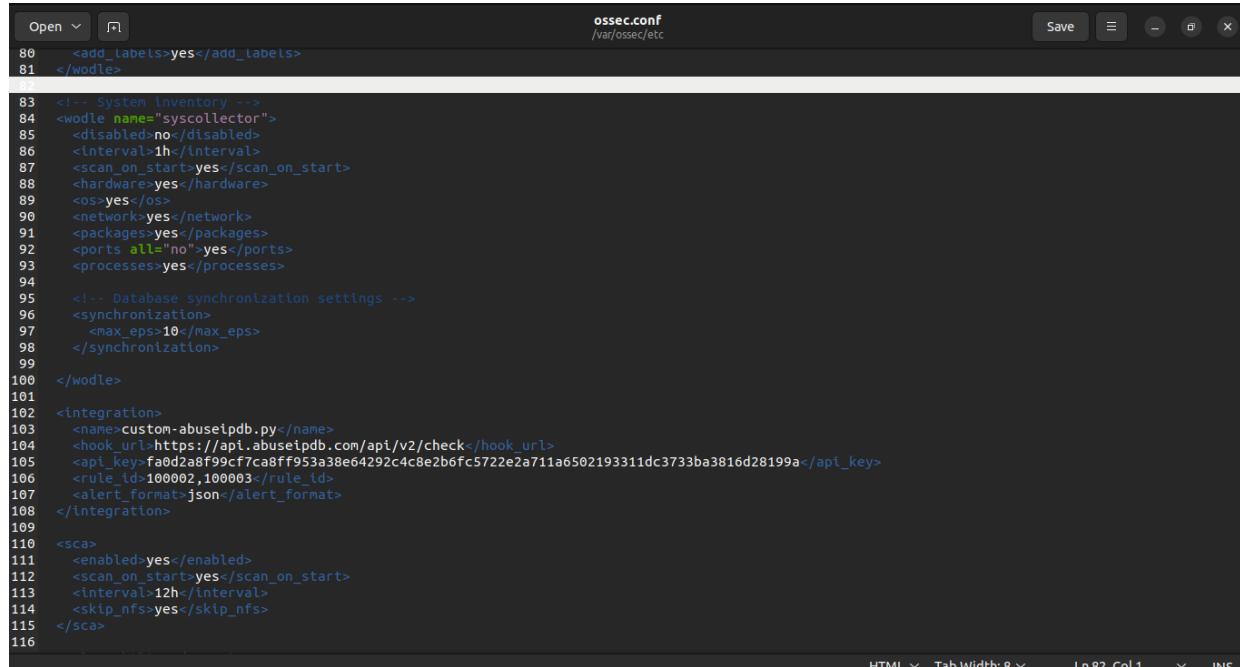
```
chmod 750 /var/ossec/integrations/custom-abuseipdb.py
chown root:wazuh /var/ossec/integrations/custom-abuseipdb.py
```

```
chown root:wazuh /var/ossec/integrations/custom-abuseipdb.py
root@ali-workstation:/home/ali# chmod 750 /var/ossec/integrations/custom-abuseipdb.py
root@ali-workstation:/home/ali# chown root:wazuh /var/ossec/integrations/custom-abuseipdb.py
root@ali-workstation:/home/ali# ls -llh /var/ossec/integrations
total 84K
-rwxr-x--- 1 root wazuh 5.6K Dec 11 20:24 custom-abuseipdb.py
-rw-r--r-- 1 root wazuh 1.1K Oct 28 18:31 maliiverse
-rw-r--r-- 1 root wazuh 20K Oct 28 18:31 maliiverse.py
-rw-r--r-- 1 root wazuh 1.1K Oct 28 18:31 pagerduty
-rw-r--r-- 1 root wazuh 6.3K Oct 28 18:31 pagerduty.py
-rw-r--r-- 1 root wazuh 1.1K Oct 28 18:31 shuffle
-rw-r--r-- 1 root wazuh 7.1K Oct 28 18:31 shuffle.py
-rw-r--r-- 1 root wazuh 1.1K Oct 28 18:31 slack
-rw-r--r-- 1 root wazuh 6.7K Oct 28 18:31 slack.py
-rw-r--r-- 1 root wazuh 1.1K Oct 28 18:31 virustotal
-rw-r--r-- 1 root wazuh 11K Oct 28 18:31 virustotal.py
root@ali-workstation:/home/ali#
```

### 4. we ad the configuration /var/ossec/etc/ossec.conf

```
<integration>
  <name>custom-abuseipdb.py</name>
  <hook_url>https://api.abuseipdb.com/api/v2/check</hook_url>
  <api_key><YOUR_ABUSEIPDB_API_KEY></api_key>
  <rule_id>100002,100003</rule_id>
  <alert_format>json</alert_format>
</integration>
```

note i replaced <YOUR\_ABUSEIPDB\_API\_KEY> with a valid one



```
Open ⌂ Save ⌂ ⌂ ⌂ ⌂ ⌂ ⌂
ossec.conf
/var/ossec/etc
80  <add_labels>yes</add_labels>
81  </wodle>
83  <!-- System Inventory -->
84  <wodle name="syscollector">
85  <disabled>no</disabled>
86  <Interval>1h</Interval>
87  <scan_on_start>yes</scan_on_start>
88  <hardware>yes</hardware>
89  <os>yes</os>
90  <network>yes</network>
91  <packages>yes</packages>
92  <ports all="no" yes</ports>
93  <processes>yes</processes>
94
95  <!-- Database synchronization settings -->
96  <synchronization>
97  <max_eps>10</max_eps>
98  </synchronization>
99
100 </wodle>
101
102 <integration>
103  <name>custom-abuseipdb.py</name>
104  <hook_url>https://api.abuseipdb.com/api/v2/check</hook_url>
105  <api_key>fa0d2a8f99cf7ca8ff953a38e64292c4c8e2b6fc5722e2a711a6502193311dc3733ba3816d28199a</api_key>
106  <rule_id>100002,100003</rule_id>
107  <alert_format>json</alert_format>
108 </integration>
109
110 <sca>
111  <enabled>yes</enabled>
112  <scan_on_start>yes</scan_on_start>
113  <interval>12h</interval>
114  <skip_nfs>yes</skip_nfs>
115 </sca>
116
```

5. now we restart the wazuh manager to apply the new configuration

```
systemctl restart wazuh-manager
```

```
[root]#  
[root@ali-workstation:/home/ali# systemctl restart wazuh-manager  
[root@ali-workstation:/home/ali# ]
```

6. we test from the agent

we add

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

to the file `/var/log/test.log`

A screenshot of a terminal window titled "Text Editor". The status bar shows the date and time as "Dec 11 22:48". The file being edited is "ossec.conf" located at "/var/ossec/etc". The code in the editor is:

```
143      <max_eps>10</max_eps>  
144      </synchronization>  
145      </syscheck>  
146  <!-- Log analysis -->  
147      <localfile>  
148        <log_format>command</log_format>  
149        <command>df -P</command>  
150        <frequency>360</frequency>  
151      </localfile>  
152      <localfile>  
153        <log_format>syslog</log_format>  
154        <location>/var/log/test.log</location>  
155      </localfile>  
156
```

now we restart the agent

```
/var/ossec/bin/wazuh-control restart
```

```
root@ali-X550CL:/home/ali# /var/ossec/bin/wazuh-control restart  
2024/12/11 22:49:24 wazuh-agentd: INFO: Max time to reconnect can't be less than notify_time(10), using notify_time*3 (30)  
Killing wazuh-modulesd...  
Killing wazuh-logcollector...  
Killing wazuh-syscheckd...  
Killing wazuh-agentd...  
Killing wazuh-execd...  
Wazuh v4.9.2 Stopped  
Starting Wazuh v4.9.2...  
Started wazuh-execd...  
2024/12/11 22:49:27 wazuh-agentd: INFO: Max time to reconnect can't be less than notify_time(10), using notify_time*3 (30)  
Started wazuh-agentd...  
Started wazuh-syscheckd...  
Started wazuh-logcollector...  
Started wazuh-modulesd...  
Completed.  
root@ali-X550CL:/home/ali# ]
```

now

```
echo "Dec 10 01:02:02 host sshd[1234]: Failed none for root from  
45.159.112.120 port 1066 ssh2" >> injector
```

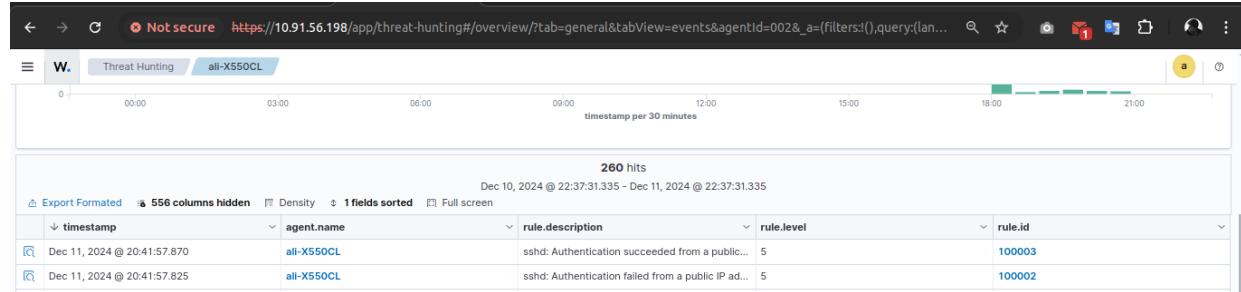
```
echo "Dec 10 01:02:02 host sshd[1234]: Accepted none for root from
64.62.197.132 port 1066 ssh2" >> injector
```

then Inject the log into `/var/log/test.log` to trigger the alert.

```
cat injector >> /var/log/test.log
```

```
root@ali-X550CL:/home/ali# echo "Dec 10 01:02:02 host sshd[1234]: Failed none for root from 45.159.112.120 port 1066 ssh2" >> injector
root@ali-X550CL:/home/ali#
root@ali-X550CL:/home/ali# echo "Dec 10 01:02:02 host sshd[1234]: Accepted none for root from 64.62.197.132 port 1066 ssh2" >> injector
root@ali-X550CL:/home/ali# cat injector >> /var/log/test.log
root@ali-X550CL:/home/ali#
```

now lets check on the dashboard



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

- 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 ... -->

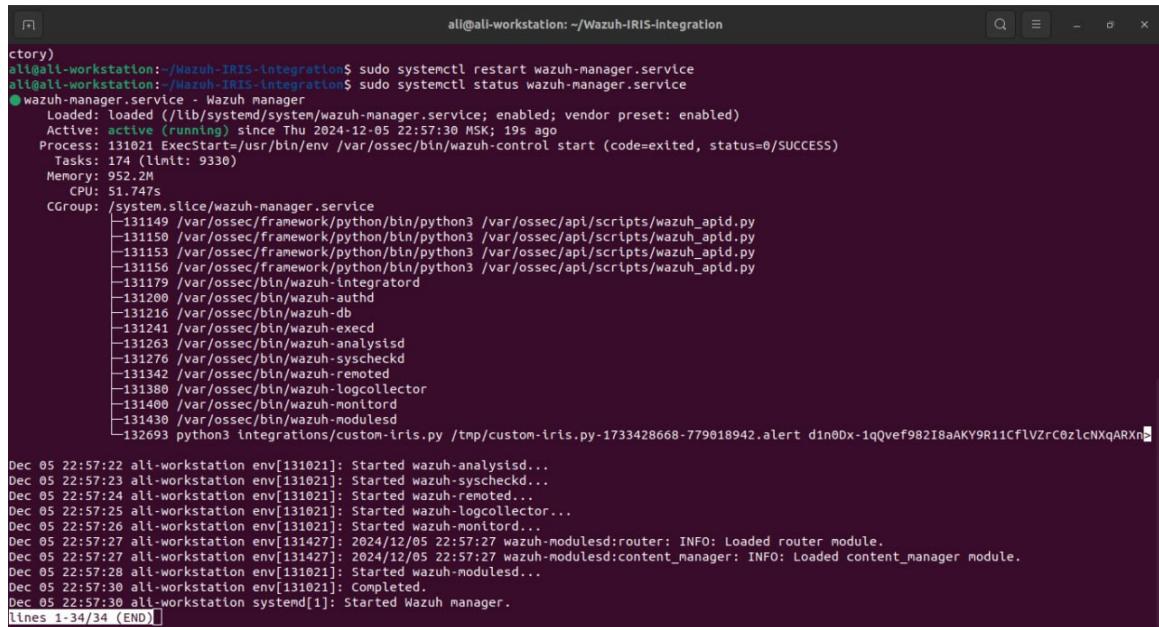
```

```

328 <!-- IRIS integration -->
329 <integration>
330 <name>custom-iris.py</name>
331 <hook_url>https://10.91.59.139/alerts/add</hook_url>
332 <level></level>
333 <api_key>d1n0Dx-1qQvef982I8aAKY9R11CflVZrC0zlcNXqARXn5-
IdcOUNHZSiPbgBTStzmDDSGYw6Z8m3_KZ8p1P5Ww</api_key>
334 <alert_format>json</alert_format>
335 </integration>

```

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

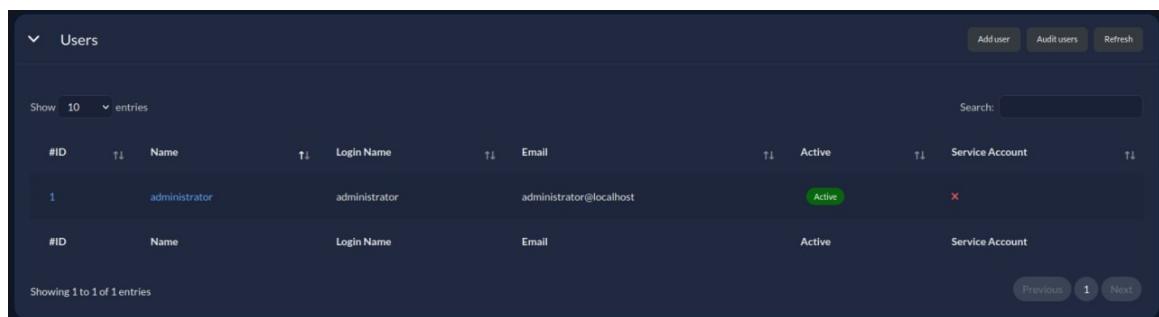


```

story)
ali@ali-workstation:~/Wazuh-IRIS-Integration$ sudo systemctl restart wazuh-manager.service
ali@ali-workstation:~/Wazuh-IRIS-Integration$ sudo systemctl status wazuh-manager.service
● wazuh-manager.service - Wazuh manager
    Loaded: loaded (/lib/systemd/system/wazuh-manager.service; enabled; vendor preset: enabled)
    Active: active (running) since Thu 2024-12-05 22:57:30 MSK; 19s ago
      Process: 131021 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
     Tasks: 174 (limit: 9330)
    Memory: 952.2M
       CPU: 51.747s
      CGroup: /system.slice/wazuh-manager.service
              └─131149 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
                  ├─131150 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
                  ├─131153 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
                  ├─131156 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh_apid.py
                  ├─131179 /var/ossec/bin/wazuh-Integratord
                  ├─131200 /var/ossec/bin/wazuh-authd
                  ├─131216 /var/ossec/bin/wazuh-db
                  ├─131241 /var/ossec/bin/wazuh-execd
                  ├─131263 /var/ossec/bin/wazuh-analysisd
                  ├─131276 /var/ossec/bin/wazuh-syscheckd
                  ├─131342 /var/ossec/bin/wazuh-remoted
                  ├─131388 /var/ossec/bin/wazuh-logcollector
                  ├─131400 /var/ossec/bin/wazuh-monitord
                  ├─131438 /var/ossec/bin/wazuh-modulesd
                  ├─132693 python3 integrations/custom-iris.py /tmp/custom-iris.py -1733428668-779018942.alert d1n0Dx-1qQvef982I8aAKY9R11CflVZrC0zlcNXqARXn5
Dec 05 22:57:22 ali-workstation env[131021]: Started wazuh-analysisd...
Dec 05 22:57:23 ali-workstation env[131021]: Started wazuh-syscheckd...
Dec 05 22:57:24 ali-workstation env[131021]: Started wazuh-remoted...
Dec 05 22:57:25 ali-workstation env[131021]: Started wazuh-logcollector...
Dec 05 22:57:26 ali-workstation env[131021]: Started wazuh-monitord...
Dec 05 22:57:27 ali-workstation env[131427]: 2024/12/05 22:57:27 wazuh-modulesd:router: INFO: Loaded router module.
Dec 05 22:57:27 ali-workstation env[131427]: 2024/12/05 22:57:27 wazuh-modulesd:content_manager: INFO: Loaded content_manager module.
Dec 05 22:57:28 ali-workstation env[131021]: Started wazuh-modulesd...
Dec 05 22:57:30 ali-workstation env[131021]: Completed.
Dec 05 22:57:30 ali-workstation systemd[1]: Started Wazuh manager.
lines 1-34/34 (END)

```

- Now everything 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:



#ID	Name	Login Name	Email	Active	Service Account
1	administrator	administrator	administrator@localhost	Active	

Showing 1 to 1 of 1 entries

- Locate Customers Management:

Edit user

- [Info](#)
- [Permissions](#)
- [Groups](#)
- [Customers](#)
- [Cases access](#)

---

Full name

Login

Email

Password (optional for service accounts)

- Must contain at least 12 chars
- Must contain at least an upper case
- Must contain at least a lower case
- Must contain at least a digit

User ID

User UUID

User API Key

```
d1n0Dx-1qQvef982l8aAKY9R11CflVZrC0zlcNxqARXn5-ldcOUNHZSiPbgBTStzmDDSGYw6Z8m3_KZ8p1P5Ww
```

[Renew](#)

- Assign Wazuh Server to a Customer:

Edit user

Customers the user belongs to

Show 10 entries

Customer

IrisInitialClient

Showing 1 to 1 of 1 entries

Manage customers membership

Customers membership \*

IrisInitialClient

Search

Select all

IrisInitialClient

Save

- Showing the alerts of that's coming from wazuh agent:

The screenshot shows the IRIS web interface with the URL [https://10.91.59.139/alerts?cid=1&page=1&per\\_page=10&sort=desc](https://10.91.59.139/alerts?cid=1&page=1&per_page=10&sort=desc). The left sidebar includes options like Dashboard, Overview, INVESTIGATION (Case, Alerts, Search, Activities, DIM Tasks), MANAGEMENT (Manage cases, Advanced), and Help. The version shown is IRIS v2.4.16. The main content area displays 3565 alerts. Two specific alerts are highlighted:

- PHP INTERNAL ERROR (MISSING FILE OR FUNCTION).** #3565 - 37A81E30-F1C8-4B8E-B06F-B14BC353A1BF  
Rule ID: 31421  
Rule Level: 5  
Rule Description: PHP internal error (missing file or function).  
Agent ID: 000  
Agent Name: all-workstation  
MITRE IDs: N/A  
MITRE Tactics: N/A  
MITRE Techniques: N/A  
Location: journal  
Full Log: Dec 05 19:59:24 all-workstation start.sh[13542]: PHP Fatal error: require\_once(): Failed opening required '/var/www/MISP/app/Vendor/autoload.php' (include\_path='/var/www/MISP/app/lib/cakephp/lib:/usr/share/php') in /var/www/MISP/app/Config/core.php on line 298
- PHP INTERNAL ERROR (MISSING FILE).** #3564 - A9FB24B3-62DB-4AF7-890A-329CC4235CD  
Rule ID: 31412  
Rule Level: 5  
Rule Description: PHP internal error (missing file).  
Agent ID: 000  
Agent Name: all-workstation  
MITRE IDs: N/A  
MITRE Tactics: N/A  
MITRE Techniques: N/A

## 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.

## III. Case Management System Assigning a Case to an Analyst

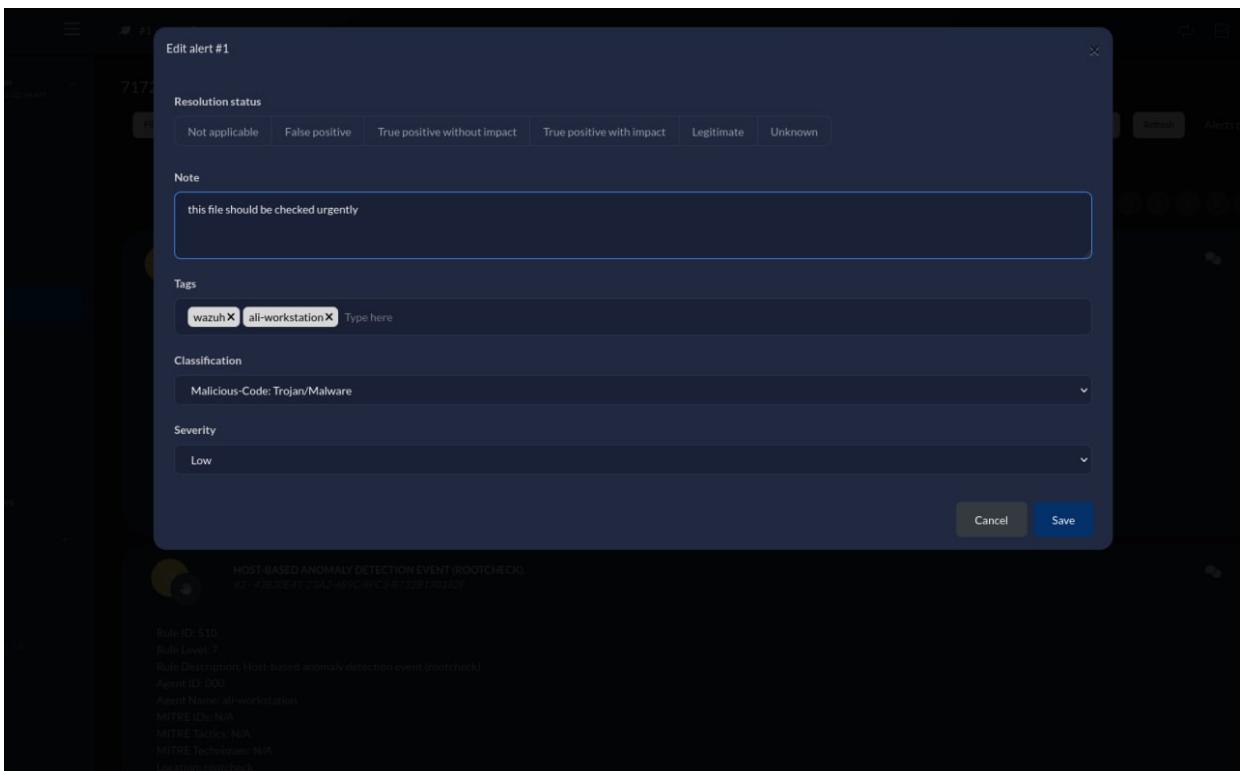
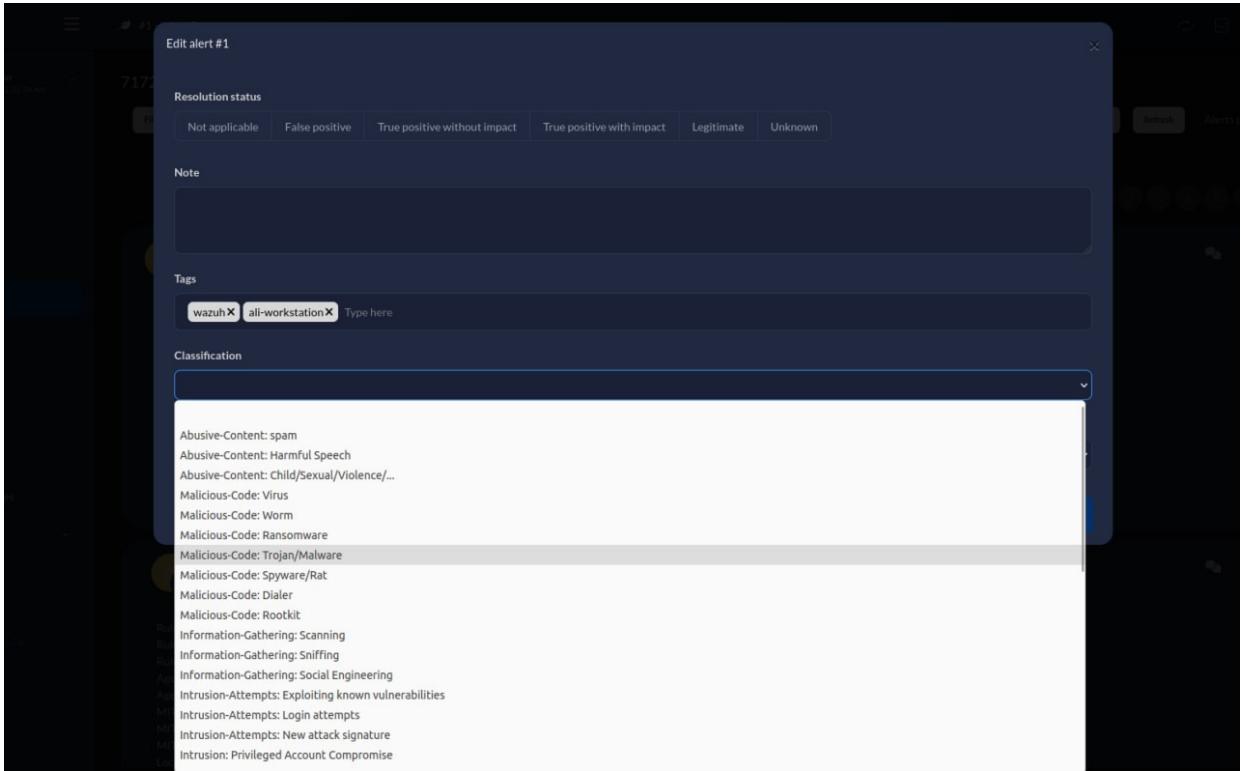
### 1. Assigning Cases in IRIS:

- Log in to the IRIS web interface.
- Navigate to the **Incident Dashboard**.
- Open an alert generated from the Wazuh integration.
- Assign the case to an analyst from the **Incident Details** section.
- Save the changes to ensure the case is linked to the chosen analyst.

### Screenshot Evidence:

The screenshot shows the IRIS web interface with the URL [https://10.91.59.139/alerts?cid=1&page=1&per\\_page=10&sort=desc](https://10.91.59.139/alerts?cid=1&page=1&per_page=10&sort=desc). The left sidebar includes options like Dashboard, Overview, INVESTIGATION (Case, Alerts, Search, Activities, DIM Tasks), MANAGEMENT (Manage cases, Advanced), and Help. The version shown is IRIS v2.4.16. The main content area displays 3565 alerts. One alert is highlighted:

**HOST-BASED ANOMALY DETECTION EVENT (ROOTCHECK).** #1 - DEC215E5-0F1E-4A17-A03E-793C94C6D1E1  
Rule ID: 510  
Rule Level: 7  
Rule Description: Host-based anomaly detection event (rootcheck).  
Agent ID: 000  
Agent Name: all-workstation  
MITRE IDs: N/A  
MITRE Tactics: N/A  
MITRE Techniques: N/A  
Location: rootcheck  
Full Log: Trojaned version of file '/bin/diff' detected. Signature used: 'bash|^/bin/sh\|file\|h\|proc\|h\|dev\|^n\|^/bin\|^sh' (Generic).



The screenshot shows the IRIS platform interface. On the left is a sidebar with navigation links: Dashboard, Overview, INVESTIGATION (Case selected), Alerts, Search, Activities, and DIM Tasks. The main content area is titled '#1 - Initial Demo'. It displays a summary card with the case name, open date (2024-12-05), owner (administrator), classification (Unknown), and customer (IrisInitialClient). Below this is a 'Case summary' section containing the text 'This is a demonstration.' and status indicators like 'Changes saved' and 'Last synced: 12:34:28 AM'. At the bottom right of the summary card are 'Edit' and 'Refresh' buttons.

This screenshot shows the 'Info' tab of the case details for '#1 - Initial Demo'. The left sidebar is identical to the first screenshot. The main content area shows the general info of the case, including its name, description ('This is a demonstration.'), and various metadata fields like classification, state, severity, open date, and owner. At the bottom are 'Delete case' and 'Close case' buttons.

This screenshot shows a confirmation dialog box over the case details page. The dialog contains a large exclamation mark icon and the question 'Are you sure?'. Below it, a message states 'Case ID 1 will be closed and will not appear in contexts anymore'. At the bottom are 'Cancel' and 'OK' buttons. The background shows the same general info and tabs as the previous screenshots.

We generated a [report](#) and added it to Github.

A screenshot of the IRIS application interface. On the left, there's a sidebar with navigation links like Dashboard, Overview, INVESTIGATION (Case, Alerts, Search, Activities, DIM Tasks), and MANAGE (Manage Users, Advanced, Help). In the center, there's a case summary for '#1 - Initial Demo' with details: Opened on 2024-12-05 by administrator, Owned by administrator, Closed on 2024-12-13. A modal window titled 'Select report template' is open over the case summary. The modal contains a message about report generation supporting images and the option to use 'Safe Mode'. It shows a dropdown menu set to 'report\_temp (English) only for testing' and two buttons: 'Generate in Safe Mode' and 'Generate'. At the bottom right of the modal, there are buttons for 'Close', 'Cancel', and 'Generate'. The background of the main interface shows a 'Case summary' section with a note 'This is a demonstration.' and some status indicators at the bottom right.

## 2. Verification:

- The assigned analyst can view the case in their dashboard.
- Notifications are sent to the analyst based on the IRIS configuration.

# IV. Simulation of Selected Security Incidents and Automated Response by the SIEM

## 1. Incident Simulation:

- Simulate a **brute force attack** using a tool like Hydra or a Python script.
- Configure Wazuh to detect failed login attempts by monitoring authentication logs.

Example command for brute force simulation:

```
hydra -l admin -P /path/to/password/list 10.91.56.198 ssh
```

## 2. Active Responses Configured:

- **host-deny**: Blocks the attacking IP by adding it to the `/etc/hosts.deny` file.
- **firewall-drop**: Blocks the attacking IP using an `iptables` rule.
- **disable-account**: Disables the account being used for brute force attempts.

## 3. Active Response Configuration Details:

### A. Host Deny

#### • Configuration:

Add the following `<active-response>` block to the Wazuh configuration file (`/var/ossec/etc/ossec.conf`):

A large empty rectangular box intended for pasting the Wazuh configuration code.

```
<active-response>
  <command>host-deny</command>
  <location>local</location>
  <level>7</level>
  <timeout>30</timeout>
</active-response>
```

## B. Firewall Drop

- **Configuration:**

Add the following `<active-response>` block to the Wazuh configuration file (`/var/ossec/etc/ossec.conf`):

```
<active-response>
  <command>firewall-drop</command>
  <location>local</location>
  <level>7</level>
  <timeout>30</timeout>
</active-response>
```

## C. Disable Account

- **Configuration:**

Add the following `<active-response>` block to the Wazuh configuration file (`/var/ossec/etc/ossec.conf`):

```
<active-response>
  <command>disable-account</command>
  <location>local</location>
  <level>7</level>
  <timeout>30</timeout>
</active-response>
```

```
243 <active-response>
244   <command>host-deny</command>
245   <location>local</location>
246   <level>7</level>
247   <timeout>30</timeout>
248 </active-response>
249
250
251   <active-response>
252     <command>firewall-drop</command>
253     <location>local</location>
254     <level>7</level>
255     <timeout>30</timeout>
256   </active-response>
257     <active-response>
258       <command>disable-account</command>
259       <location>local</location>
260       <level>7</level>
261       <timeout>30</timeout>
262 </active-response>
```

After adding every active response to the file `/var/ossec/etc/ossec.conf`, then restart the wazuh manager we can test the actions after that :

```
[root@ali-workstation ~]# sudo systemctl restart wazuh-manager.service
[sudo] password for ali:
```

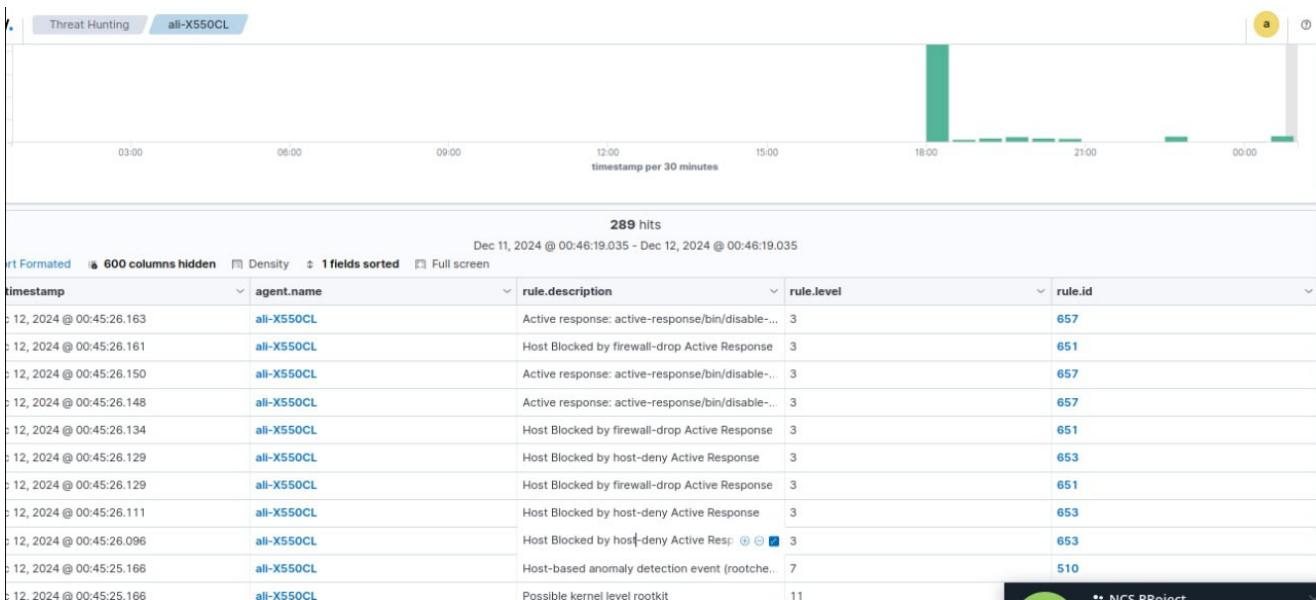
#### 4. Testing the Active Responses:

we tried the ssh connection with wrong password which will run all of above responses:

```
Last login: Thu Dec 12 00:07:45 on console
alihamdan@Alis-MacBook-Air-2 ~ % ssh -V

OpenSSH_9.8p1, LibreSSL 3.3.6
[alihamdan@Alis-MacBook-Air-2 ~ % ssh ali@10.91.59.102
The authenticity of host '10.91.59.102 (10.91.59.102)' can't be established.
ED25519 key fingerprint is SHA256:2sOExVPHYw+57EMn15cLYOUfc4LfHkI90G5Z7Yfbelk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.91.59.102' (ED25519) to the list of known hosts.
[ali@10.91.59.102's password:
Permission denied, please try again.
[ali@10.91.59.102's password:
Permission denied, please try again.
[ali@10.91.59.102's password:
ali@10.91.59.102: Permission denied (publickey,password).
alihamdan@Alis-MacBook-Air-2 ~ %
```

```
alihamdan@Alis-MacBook-Air-2 ~ % ssh -V  
OpenSSH_9.8p1, LibreSSL 3.3.6  
[alihamdan@Alis-MacBook-Air-2 ~ % ssh ali@10.91.59.102  
The authenticity of host '10.91.59.102 (10.91.59.102)' can't be established.  
ED25519 key fingerprint is SHA256:2sOExVPHYw+57EMn15cLYOUfc4LfHkI90G5Z7Yfbelk.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '10.91.59.102' (ED25519) to the list of known hosts.  
[ali@10.91.59.102's password:  
Permission denied, please try again.  
[ali@10.91.59.102's password:  
Permission denied, please try again.  
[ali@10.91.59.102's password:  
ali@10.91.59.102: Permission denied (publickey,password).  
alihamdan@Alis-MacBook-Air-2 ~ % ]
```



## 4. Difficulties Faced and New Skills Acquired

### Difficulties Faced

- Challenges in tool integration and resolving API incompatibilities.
- Troubleshooting network configurations and performance issues.
- the main problem it was with MISP and integration with wazuh
  1. first problem there is poor documentaion that compatable with the updated versions
  2. we tried to install the MISP locally but alawys it damage the opearating system distrubution
  3. we find that we can do it with docker but still we needed to update a lot on the docker-compose.yaml file and on configuration with docker DNS to build and up MISP
  4. there are only documentaion to how to automat requesting from sysmon only (windows ) and our agent is linux how ever we tried with windows agent and did not work

### New Skills Acquired

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

## 5. Conclusion, Contemplations, and Judgement

### Conclusion

- **Effectiveness of the SOC Setup:**

The implemented SOC demonstrated its ability to detect and respond to various security incidents, such as brute force attacks and malware infections. By integrating Wazuh with AbuselPDB, the system leveraged enriched threat intelligence, while IRIS streamlined case management and escalation processes. The automation of responses enhanced operational efficiency and reduced reaction times.

- **Strengths of the Tools Used:**

The combination of Wazuh, AbuseIPDB, and IRIS showcased the potential of open-source tools to deliver robust and scalable solutions for security operations. Their flexibility, coupled with accessible documentation, allowed for seamless integration and configuration. Additionally, the cost-effectiveness of open-source software made it a practical choice for building a functional SOC.

- **Areas for Improvement:**

Despite its successes, the project revealed some limitations, including challenges with API compatibility, insufficient documentation for advanced integrations (e.g., MISP), and scalability concerns under heavy traffic loads. Addressing these issues would further enhance the system's effectiveness and reliability.

## Recommendations

- **Enhance Existing Workflows:**

Fine-tune alerting and response workflows to improve detection accuracy and minimize false positives. Adjust Wazuh rules and thresholds to better align with real-world scenarios and emerging threats.

- **Expand the Toolset:**

Integrate additional tools such as MISP for collaborative threat intelligence sharing or Elastic Stack for enhanced data visualization and analytics. These integrations would enrich the overall SOC ecosystem and provide deeper insights into security events.

- **Prepare for Larger Deployments:**

Evaluate the system's performance in a high-traffic environment and make necessary adjustments to support scalability. This includes implementing redundancy for critical components and ensuring the infrastructure can handle increased demands.

- **Develop Advanced Automation:**

Design and test automated responses for more complex threats, such as lateral movement or data exfiltration attempts. Incorporating advanced automation would strengthen the SOC's ability to respond rapidly to sophisticated attacks.

- **Focus on Training and Documentation:**

Create detailed training resources for SOC analysts, including step-by-step playbooks for managing common incident types. Strengthen internal documentation with troubleshooting guides and instructions for advanced configurations, ensuring easier adoption and maintenance of the system.

---

## 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.