WazuGuardix Internship

Prepared By: Muhammad Safwan

Date: 31st December 2025

A Practical SOC Internship 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:

The objective of this task is to enable interns to integrate MISP (Malware Information Sharing Platform) and a Honeypot with Wazuh SIEM in order to enhance threat intelligence, detect real-world attacks, and improve SOC visibility. This task focuses on understanding how external threat feeds and deception technologies strengthen detection and response capabilities in a SOC environment.

Audience:

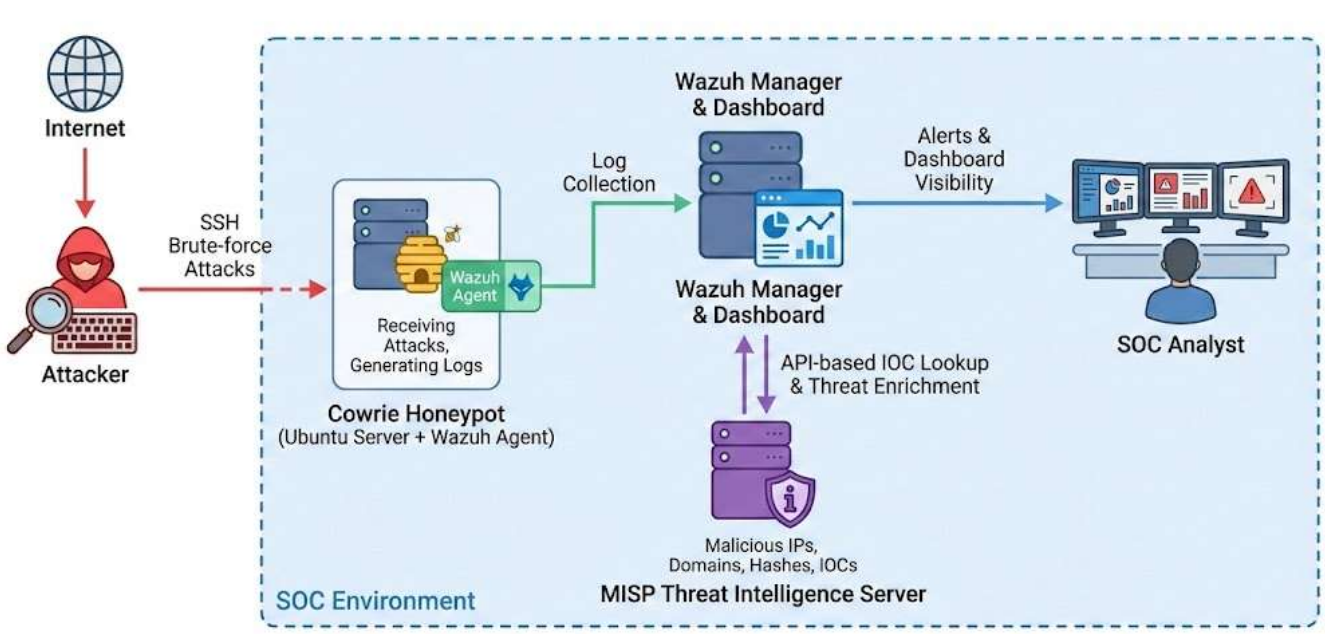
Ideal for students, SOC Students, and cybersecurity enthusiasts looking to build a strong technical foundation and gain practical skills in security monitoring and automation.

Overview of MISP Integration with Wazuh SIEM

This task focuses on implementing threat intelligence integration using MISP to enhance detection and analysis capabilities within a SOC environment. The setup is designed to utilize external intelligence feeds to identify known malicious indicators and support investigation of security events.

MISP is deployed to collect, manage, and share indicators of compromise (IoCs) from OSINT and community feeds. Through API-based integration, security events can be enriched with threat intelligence, helping analysts validate suspicious activity against known malicious signatures.

This implementation demonstrates how threat intelligence from MISP improves visibility, strengthens detection accuracy, and supports informed analysis by providing contextual information about potential threats in a centralized and structured manner.



SOC Environment: Integrated Cybersecurity Architecture with Deception, SIEM, and Threat Intelligence Enrichment

MISP Overview:

MISP, which is Malware Information Sharing Platform and Threat Sharing, is a platform developed in all its modules, and as such is an open opportunity to share, to collate threat information, analyze and distribute threat intel in all its forms. Countries, governments, and industries use this platform to share and evaluate threats, and we intend to give you the training that will ensure you jump the learning curve as fast as you can.

Installation of MISP on Ubuntu:

In this first step, I'm preparing the underlying Ubuntu operating system for the integration. I executed **sudo apt-get update -y && sudo apt-get upgrade -y** to ensure that all local package indexes are synchronized with the latest security patches and software versions.

```
safwan@safwan-VirtualBox:~$ sudo apt-get update -y && sudo apt-get upgrade -y
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Hit:2 http://pk.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://pk.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metada
ta [74.7 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 DEP-11
Metadata [212 B]
Get:6 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Me
tadata [160 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11
Metadata [940 B]
Get:8 http://pk.archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:9 http://pk.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metad
ata [276 kB]
Get:10 http://pk.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 DEP-1
1 Metadata [212 B]
Get:11 http://pk.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11
Metadata [446 kB]
Get:12 http://pk.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-1
1 Metadata [940 B]
Get:13 http://pk.archive.ubuntu.com/ubuntu focal-backports/main amd64 DEP-11 Me
tadata [7,980 B]
Get:14 http://pk.archive.ubuntu.com/ubuntu focal-backports/restricted amd64 DEP
-11 Metadata [216 B]
Get:15 http://pk.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-1
1 Metadata [30.5 kB]
Get:16 http://pk.archive.ubuntu.com/ubuntu focal-backports/multiverse amd64 DEP
-11 Metadata [212 B]
```

MySQL is being set up through **sudo apt-get install mysql-client -y**. MISP has a MariaDB/MySQL backend for storing IoCs, so having the client installed allows the local machine to verify database connections and do health checks on the backend, if needed. The system is pulling the needed binary files and resolving the dependencies as shown in the terminal.

```
safwan@safwan-VirtualBox:~$ sudo apt-get install mysql-client -y
[sudo] password for safwan:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

In this step, I retrieved MISP's official installation script, which is also on the MISP GitHub repository, by using the wget command. This script is maintained by the MISP Project. This script has all the needed logic to install MISP and its dependencies. Because I obtained the script, I proved to myself that the system had internet access and could reach the official MISP source.

```
safwan@safwan-VirtualBox:~$ wget https://raw.githubusercontent.com/MISP/MISP/2.
4/INSTALL/INSTALL.sh
--2025-12-23 07:31:29--  https://raw.githubusercontent.com/MISP/MISP/2.4/INSTAL
L/INSTALL.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.
133, 185.199.109.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110
.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 159642 (156K) [text/plain]
Saving to: 'INSTALL.sh.1'

INSTALL.sh.1        100%[===================>] 155.90K  --.-KB/s    in 0.1s

2025-12-23 07:31:29 (1.52 MB/s) - 'INSTALL.sh.1' saved [159642/159642]
```

Once I got the installation script, I had to, first, make the script executable. This is accomplished through the command **chmod +x INSTALL.sh**. Thus, is done so the script can, in fact, be run in the terminal. The system requires certain permissions for the installation script to run on a Linux system, for security and functionality reasons.

```
safwan@safwan-VirtualBox:~$ chmod +x INSTALL.sh
```

I started the automated setup at first by running the integration script with the command: **./INSTALL.sh -A**. The script runs with a check at the beginning of each process as to whether the user is running with proper install permissions, verifying they are being sudo'ed, confirming the distribution of their OS as Ubuntu 20.04, and the check is fully supported by MISP before proceeding.

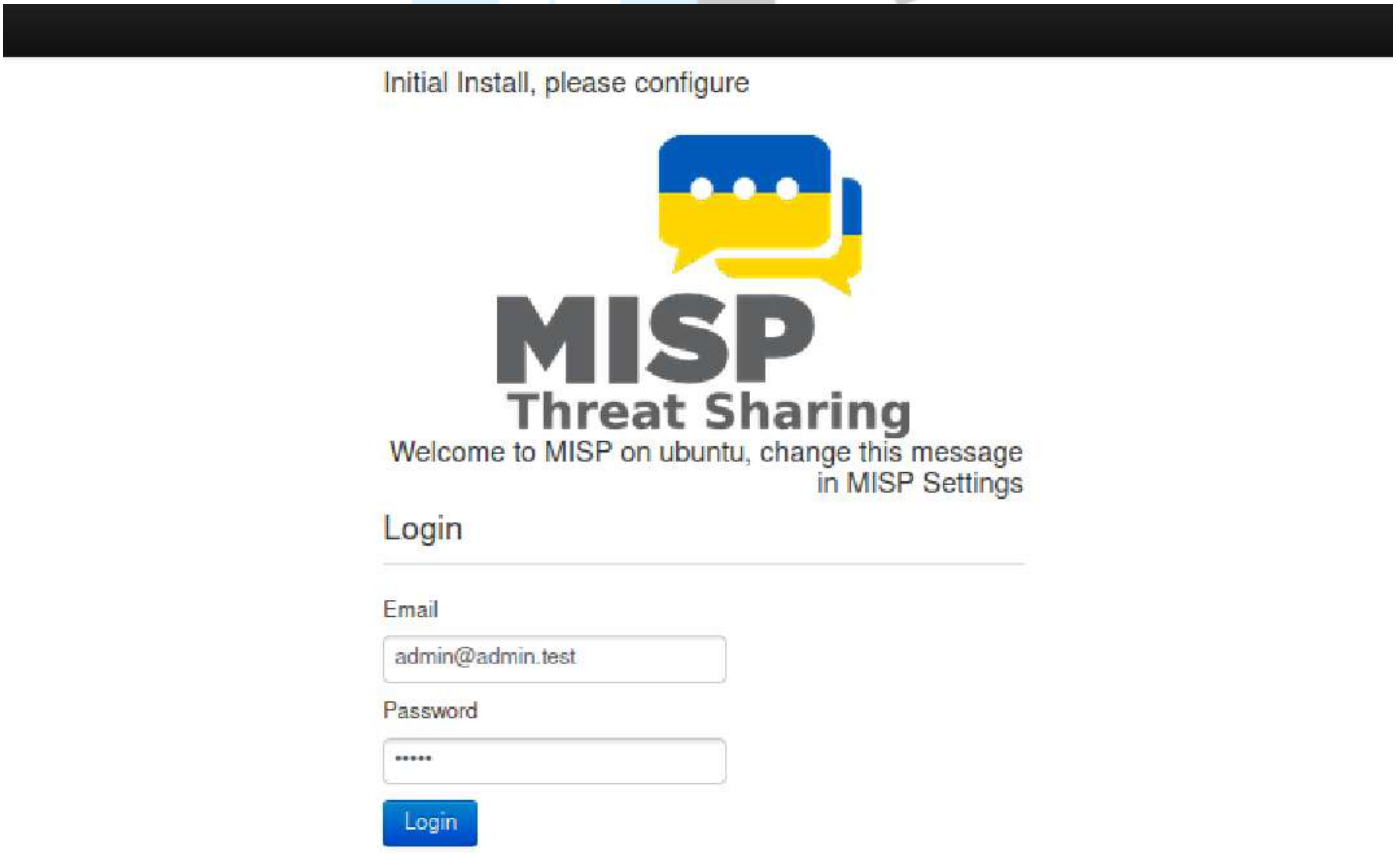
```
safwan@safwan-VirtualBox:~$ ./INSTALL.sh -A
Next step: Checking if we are run as the installer template
Next step: Checking Linux distribution and flavour...
Next step: We detected the following Linux flavour: Ubuntu 20.04
--------------------------------------------------------------------------------
Next step: Setting MISP variables
Next step: Setting generic MISP variables shared by all flavours
--------------------------------------------------------------------------------
To enable outgoing mails via postfix set a permissive SMTP server for the domai
ns you want to contact:

sudo postconf -e 'relayhost = example.com'
sudo postfix reload
--------------------------------------------------------------------------------
Enjoy using MISP. For any issues see here: https://github.com/MISP/MISP/issues
--------------------------------------------------------------------------------
safwan@safwan-VirtualBox:~$
```

I updated the firewall settings using the commands sudo ufw allow 80/tcp and sudo ufw allow 443/tcp. This opens the ports for **HTTP** and **HTTPS** traffic, ensuring that the MISP web interface is accessible and that the integration (Done Below) can communicate with the MISP API without being blocked.

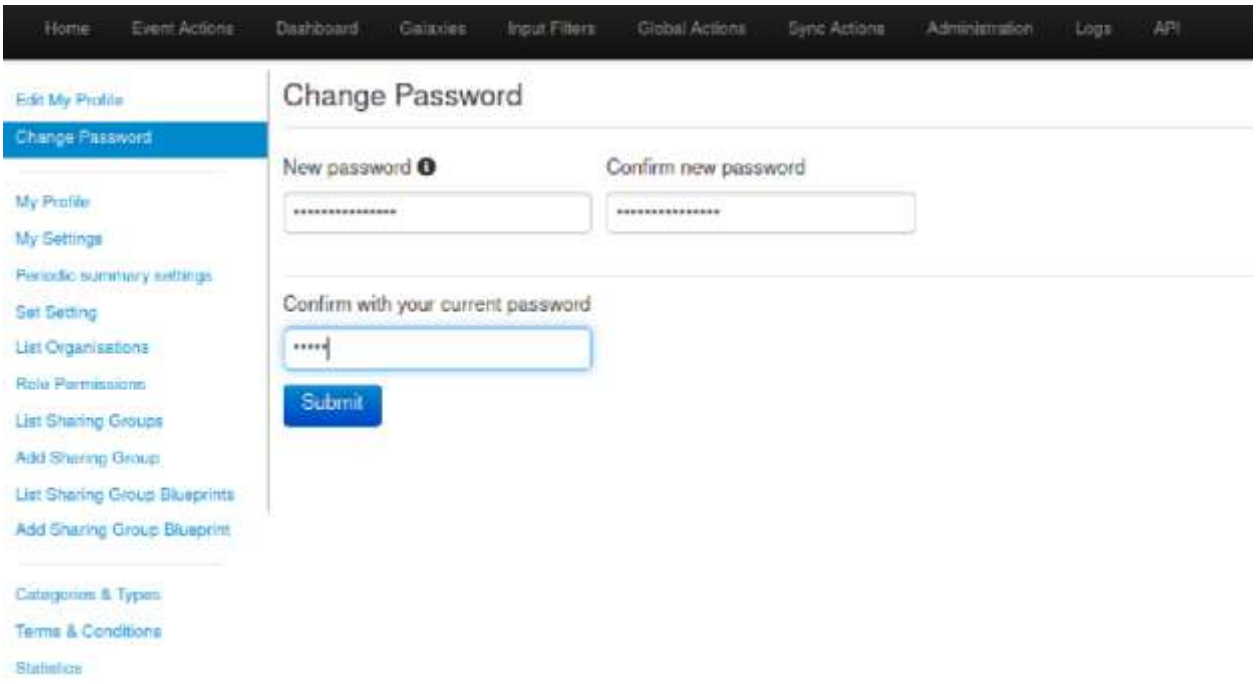
```
safwan@safwan-VirtualBox:~$ sudo ufw allow 80/tcp
Rules updated
Rules updated (v6)
```

```
safwan@safwan-VirtualBox:~$ sudo ufw allow 443/tcp
Rules updated
Rules updated (v6)
safwan@safwan-VirtualBox:~$
```

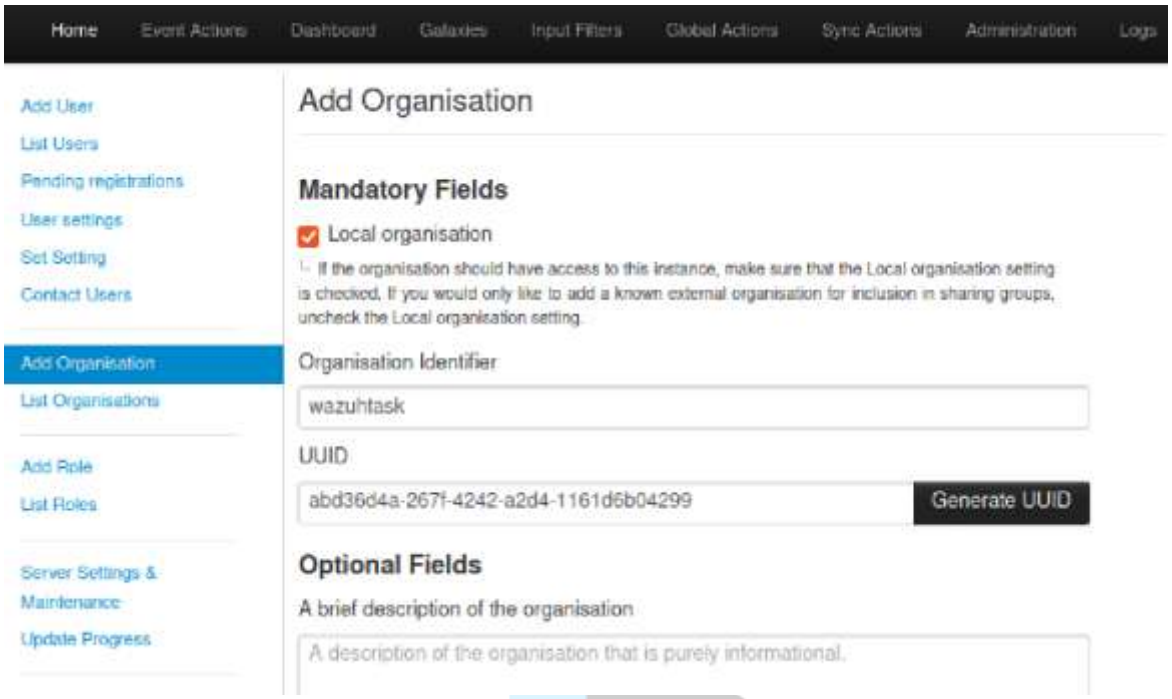
Once the process had finished running and the configured variables were properly in place, I connected to the system by inserting the Ubuntu's IP address in the URL of the web browser to ensure properly deployed MISP. I accessed the application and validated the platform's full operational capacity. Then with the help of default credentials I logged into the MISP System.



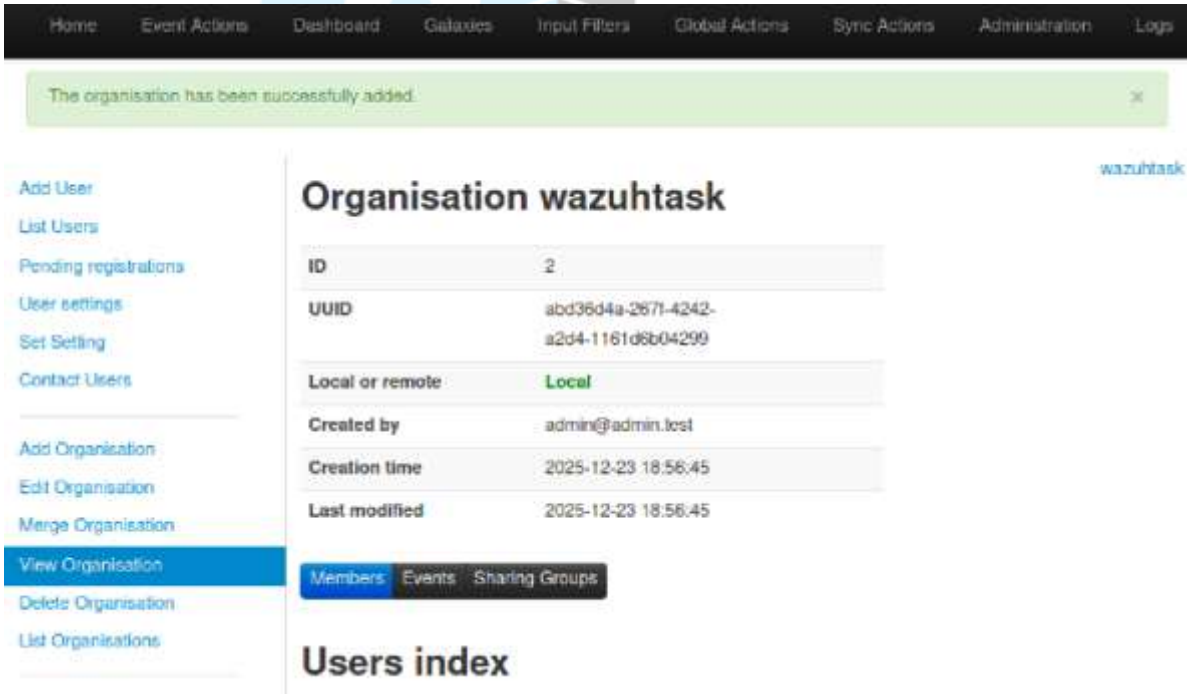
After that, I changed the default password of the MISP instance with my own to enhance security.



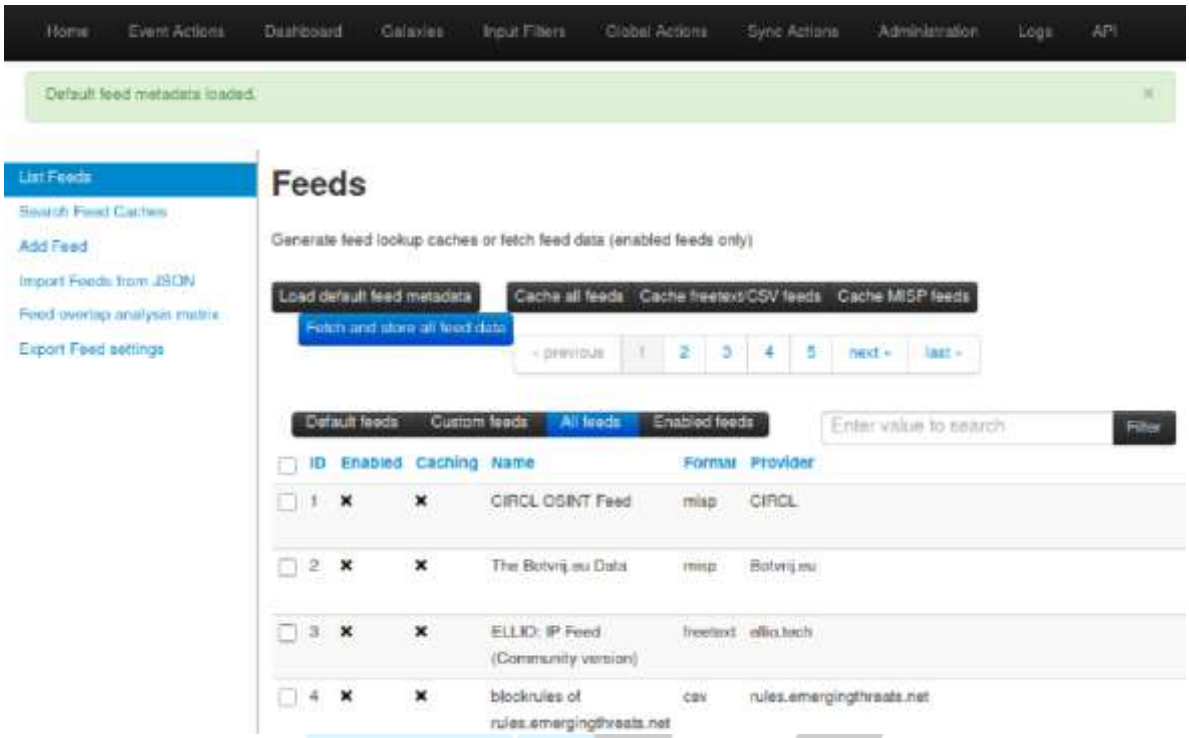
Then, I clicked the "Add Organisation" and added new organization named "wazuhtask". MISP proceeded to create a new local organisation with the user as the local manager, where a new unique local UUID was generated to signify ownership and trust as the primary internal custodian of the derived threat intelligence. The account system we set allows maintenance, and the observance of the controls policies of the federated account system to be observed over the correlated Wazuh alerts from the indicators.



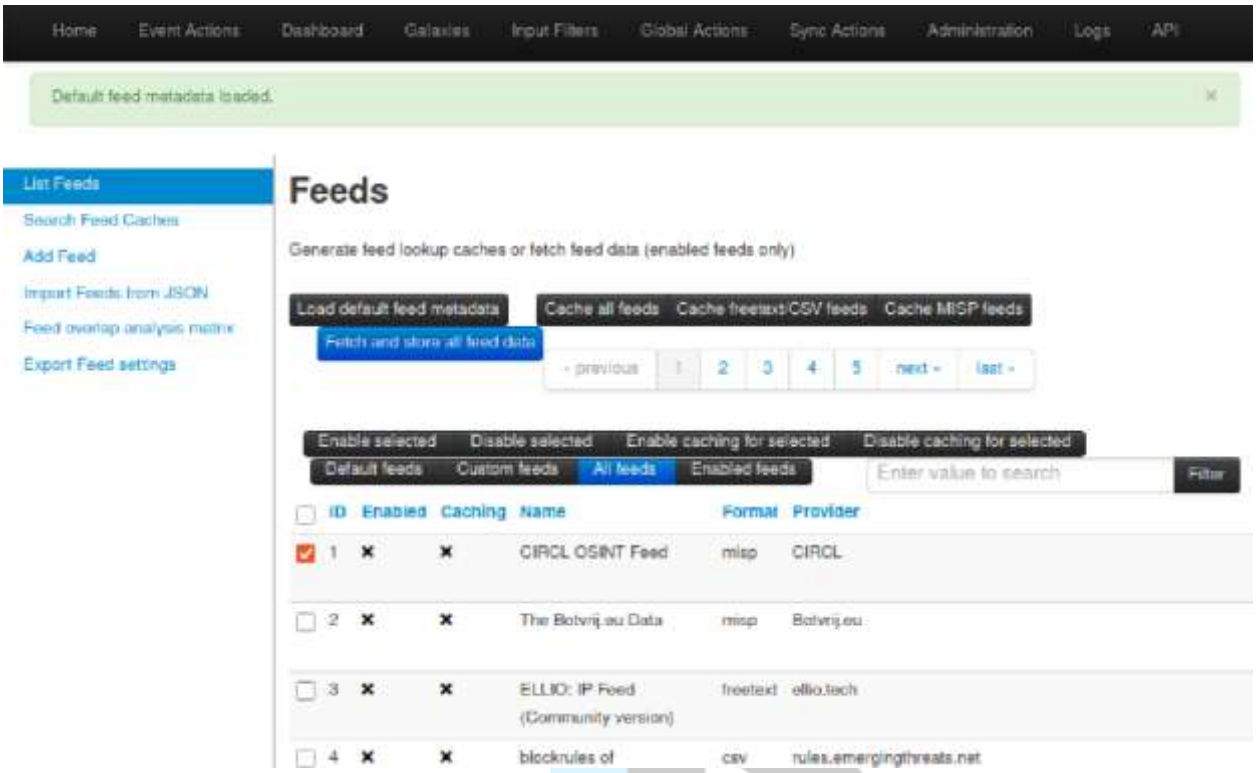
This below screenshot shows the successful addition of new organization as described in the previous screenshot.

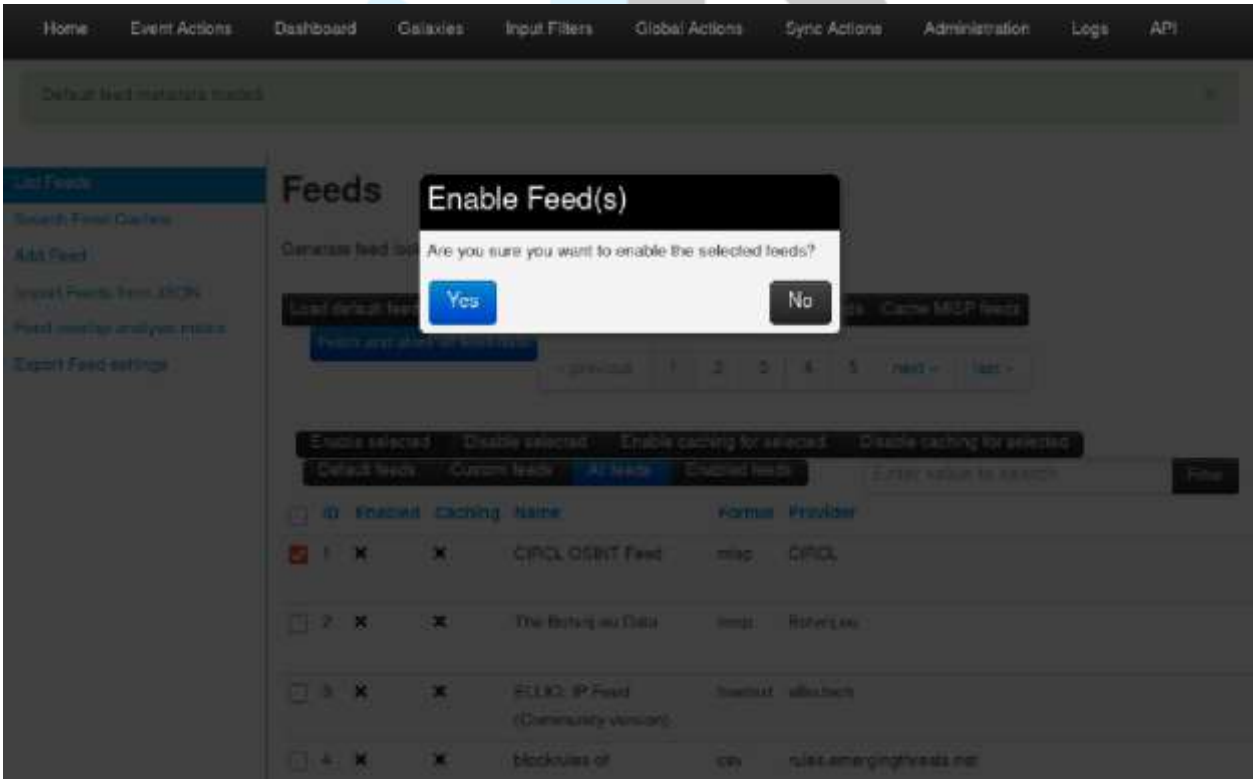


To begin building external threat intelligence sources in this phase, I selected Load default feed metadata from the MISP Feeds screen. This filled the platform with community-level open-source intelligence (OSINT) providers, including the CIRCL OSINT Feed, Botvrij.eu, and Emerging Threats rules. It is necessary to begin these feeds in an SOC environment and allows the platform to ingest and correlate worldwide threats with local security data for more effctive incident response.



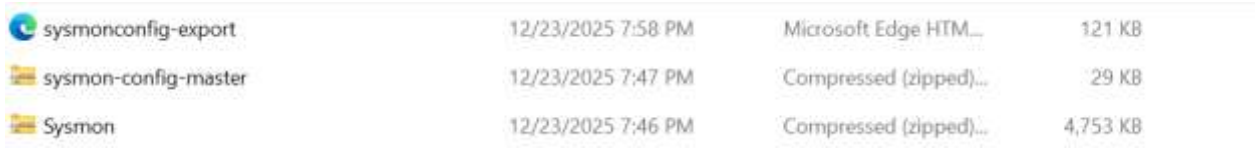
In this step, I prepared to activate an individual intelligence source by selecting the CIRCL OSINT Feed entry. Caching a feed will download all the feed's IOCs as attributes onto your MISP instance's Redis server. The feed allows the administrative menu to open and lets the use of Enable selected and Enable caching for selected functions. This using this granular selection process is an important aspect of threat data management.


Home
Event Actions
Dashboard
Galaxies
Input Filters
Global Actions
Sync Actions
Administration
Logs
API
Default feed metadata loaded.
List Feeds
Search Feed Caches
Add Feed
Import Feeds from JSON
Feed overlap analysis matrix
Export Feed settings
Feeds
Generate feed lookup caches or fetch feed data (enabled feeds only)
Load default feed metadata
Cache all feeds
Cache freetext/CSV feeds
Cache MISP feeds
Fetch and store all feed data
Enable selected
Disable selected
Enable caching for selected
Disable caching for selected
Default feeds
Custom feeds
All feeds
Enabled feeds
Enter value to search
Filter
ID
Enabled
Caching
Name
Format
Provider
1
CIRCL OSINT Feed
misp
CIRCL
2
The Botvrij.eu Data
misp
Botvrij.eu
3
ELLIO: IP Feed
(Community version)
freetext
ellio.tech
4
blocklist.de
csv
rules.emergingthreats.net


Home
Event Actions
Dashboard
Galaxies
Input Filters
Global Actions
Sync Actions
Administration
Logs
API
Feeds
Enable Feed(s)
Are you sure you want to enable the selected feeds?
Yes
No

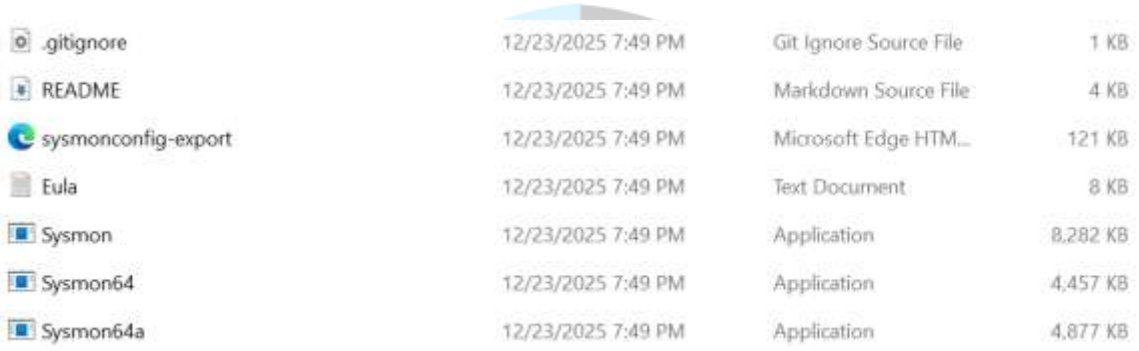
Setting up Sysmon on wazuh Agent:

First, I downloaded Sysmon and the configuration file sysmon-config-master along with sysmonconfig-export. The first step in configuring better security monitoring on the computer is having these files.



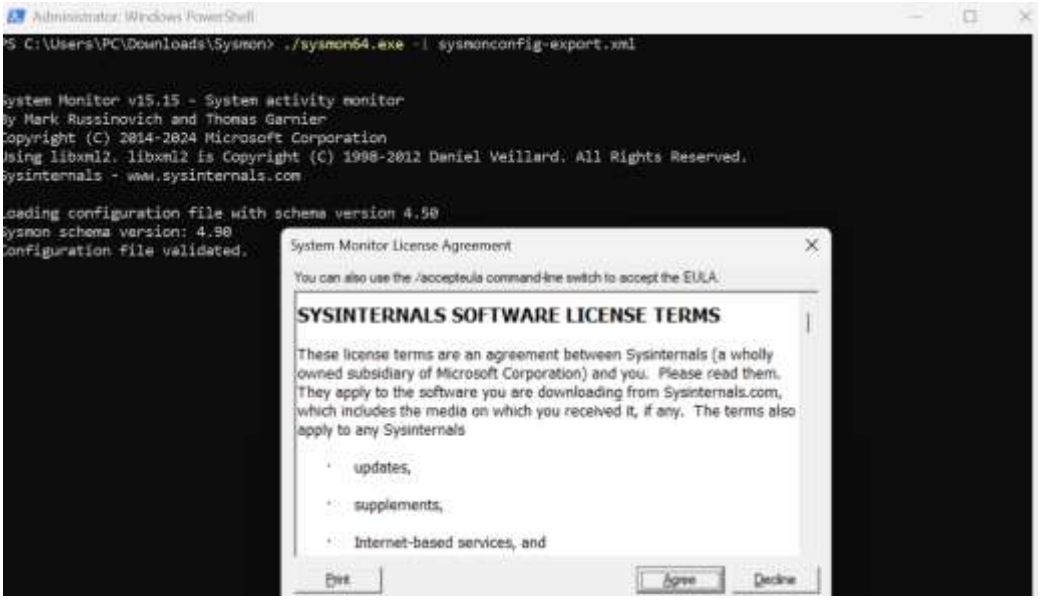
sysmonconfig-export	12/23/2025 7:58 PM	Microsoft Edge HTM...	121 KB
sysmon-config-master	12/23/2025 7:47 PM	Compressed (zipped)...	29 KB
Sysmon	12/23/2025 7:46 PM	Compressed (zipped)...	4,753 KB

After that I unzipped the files and moved them to a single directory. I keep the Sysmon64 program and the sysmonconfig-export file in the same directory. This allows me to perform the next step more conveniently since I will be installing the program and simultaneously applying the security rules with one command.



.gitignore	12/23/2025 7:49 PM	Git Ignore Source File	1 KB
README	12/23/2025 7:49 PM	Markdown Source File	4 KB
sysmonconfig-export	12/23/2025 7:49 PM	Microsoft Edge HTM...	121 KB
Eula	12/23/2025 7:49 PM	Text Document	8 KB
Sysmon	12/23/2025 7:49 PM	Application	8,282 KB
Sysmon64	12/23/2025 7:49 PM	Application	4,457 KB
Sysmon64a	12/23/2025 7:49 PM	Application	4,877 KB

Afterward, I executed a command to install Sysmon with the the Windows PowerShell terminal in Administrator mode. The command I used was **./sysmon64.exe -i sysmonconfig-export.xml** to ensure that Sysmon installs with the specific security rules that we prepared in advance. Installing Sysmon also requires an Accept License Agreement screen to pop up, which is a step to finalizing the setup and turning on the service.



In this screenshot below, the installation and running of Sysmon have been verified in this step. Sysmon64 and its associated driver SysmonDrv have been installed and initiated as indicated in the output of the command. This indicates the system is now monitoring and will capture security events in preparation for threat detection.

```
Loading configuration file with schema version 4.50
Sysmon schema version: 4.90
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon64..
Sysmon64 started.
PS C:\Users\PC\Downloads\Sysmon>
```

In this step, I incorporated a new configuration fragment into the setup by the Wazuh agent. This integration instructs Wazuh to monitor the Operational logs of Windows Defender. This addition means the Wazuh manager is now able to receive and process security alerts from Windows Defender. This will enhance the scope of the insight we have into any malware and other threats the computer may have.

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

The new configuration changes were implemented in this step by using the command Restart-Service Wazuh to restart the Wazuh service. I then used Get-Service Wazuh to verify that the service is in a running state after the restart. The output returned a status of Running, indicating that the agent is now actively and successfully tracking system activity and new monitoring rules have been implemented.

```
PS C:\Users\PC\Downloads\Sysmon> Restart-Service Wazuh
PS C:\Users\PC\Downloads\Sysmon> Get-Service Wazuh

Status   Name               DisplayName
------   ----               -----------
Running  WazuhSvc           Wazuh
```

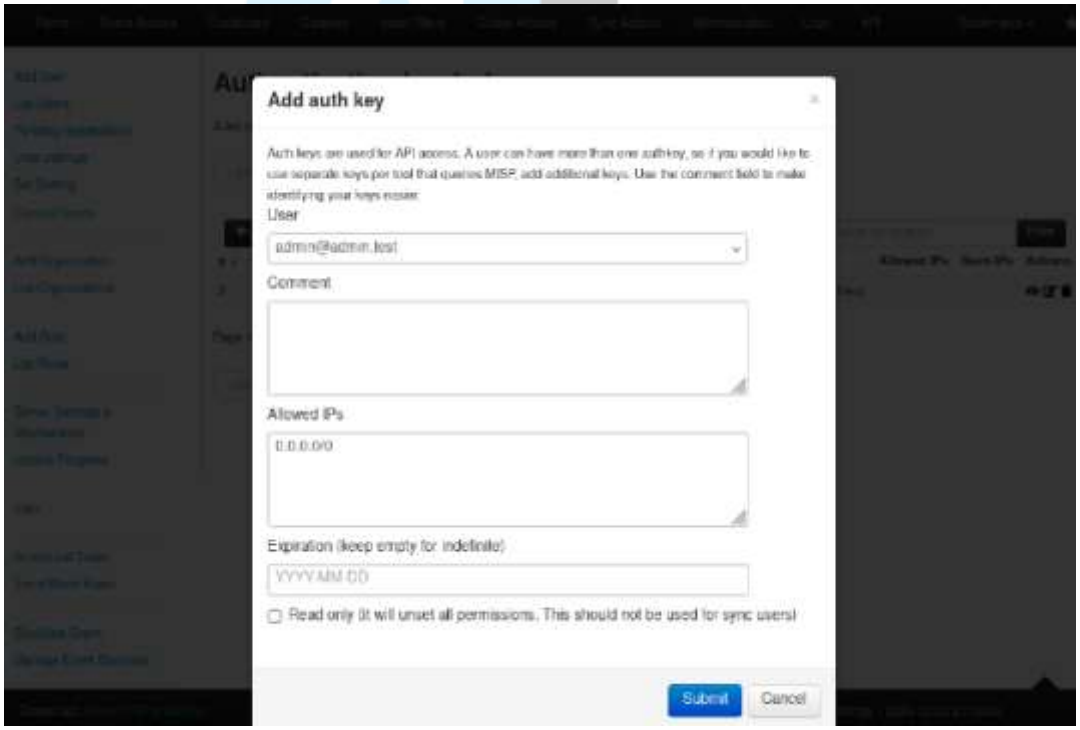
In this step, I added a new security rule to the Wazuh manager in order for DNS queries The rule (ID 61650) measures at "Level 8" and it is essential to looks for specific Sysmon events as well. By adding this rule, the system will now create an alert whenever a computer tries to look up a website or server address, which is a great way to spot suspicious network activity.

```
<group name="sysmon,windows,dns">
  <rule id="61650" level="8" overwrite="yes">
    <if_sid>61600</if_sid>
    <field name="win.system.eventID">^22$</field>
    <description>Sysmon - Event ID 22: DNSEvent (DNS query)</description>
    <options>no_full_log</options>
    <group>sysmon_event_22,</group>
  </rule>
</group>
```

In this step, I restarted the Wazuh manager on the Linux server using the command systemctl restart wazuh-manager. This is a necessary step to make sure the manager loads the new DNS rules I just added. Restarting ensures that the entire security platform is up to date and ready to start processing the new types of logs.

```
root@safwan-VirtualBox:/home/safwan# systemctl restart wazuh-manager

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

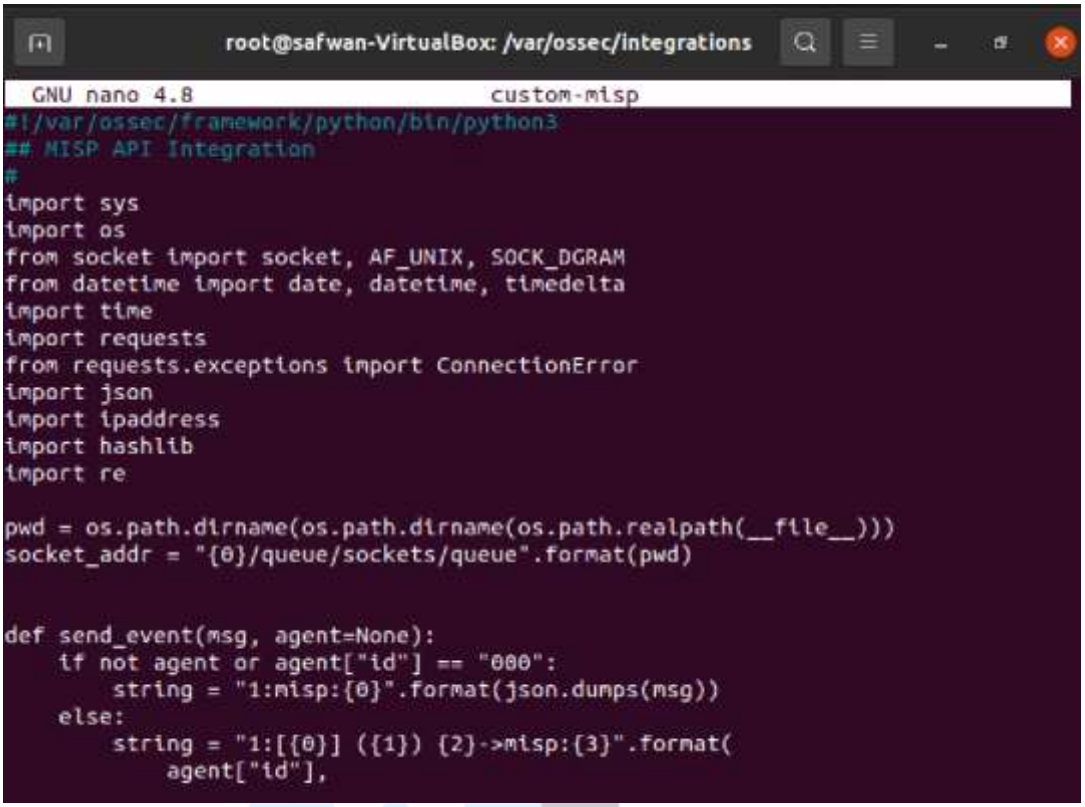
Here I launched the step from auth keys, this generated a unique key API to access MISP. Once I had confirmed the user information just now, I clicked the Submit button to generate key.



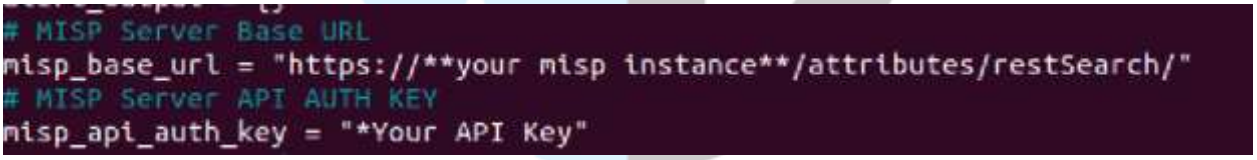
Integrating MISP Api with Wazuh:

In this step, I downloaded a specialized integration script from GitHub and opened it in the nano editor. I placed this script in the /var/ossec/integrations directory and named it custom-misp. As required for Wazuh integrations, I saved the file **without the .py extension**. This script is the "bridge" that allows the Wazuh manager to talk to the MISP platform, enabling them to

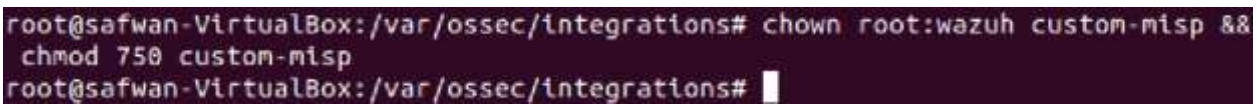
automatically cross-check security events against known threat intelligence.



In this step, I edited the Python script to include placeholders for the MISP Base URL and the API Auth Key. Following security best practices, I did not enter the live keys at this stage to keep the credentials safe.



I modified the ownership of the file to the root user and the wazuh group utilizing the chown command. I then proceeded to execute command chmod 750 to modify the file access permissions. This way, the Wazuh manager can access and execute the script, but it is maintained without modification by other non-privileged users.



In this step, I inserted integration blocks into the Wazuh configuration file (ossec.conf). I provided the integration name, custom-misp, and identified the event groups—namely, the network connections and changes to processes—that Wazuh should forward to MISP. This integration is the mechanism that enables Wazuh to retrieve the information it has and compare it to the threat intelligence information available in MISP.

```
<integration>
<name>custom-misp</name>
<group>sysmon_event1,sysmon_event3,sysmon_event6,sysmon_event7,sysmon_event_15
<alert_format>json</alert_format>
</integration>
```

In this step, I created a new rules file called misprule.xml. I added rules that tell Wazuh how to handle information coming back from MISP.

Rule 100620 (Level 10): General MISP Events, this is the mainmost rule. It tells Wazuh to watch for any MISP Integration data. While custom script sends data to Wazuh, this rule detects it, and the system knows how to complete that step.

Rule 100621 (Level 5): Connection Errors, this rule is looking for errors specifically. It will raise an alert if the Wazuh script attempts to contact the MISP server and cannot for any reason (e.g. incorrect API key, server is down). This lets you know if the connection is dead.

Rule 100622 (Level 12): Threat Detected (IoC Found), this is the most critical rule. It is triggered if MISP detects a correlation between your machine's activity and any known Indicators of Compromise (IoCs) like a malicious IP address or malware file. This rule is important because it is Level 12 and generates a critical alert, so analysts are aware it has found a possible threat.

```
 GNU nano 4.8                          misprule.xml
<group name="misp,">
<rule id="100620" level="10">
<field name="integration">misp</field>
<match>misp</match>
<description>MISP Events</description>
<options>no_full_log</options>
</rule>
<rule id="100621" level="5">
<if_sid>100620</if_sid>
<field name="misp.error">\.+</field>
<description>MISP - Error connecting to API</description>
<options>no_full_log</options>
<group>misp_error,</group>
</rule>
<rule id="100622" level="12">
<field name="misp.category">\.+</field>
<description>MISP - IoC found in Threat Intel - Category: $(misp.category), At>
<options>no_full_log</options>
<group>misp_alert,</group>
</rule>
</group>
```

I executed the command systemctl restart wazuh-manager to finish all settings configuration for the integration and the newly added alert rules. After the manager finishes the restart process, the entire system for automated threat intelligence is fully functional and operational.

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