MISP Project Report

Contents

1 Intro to MISP

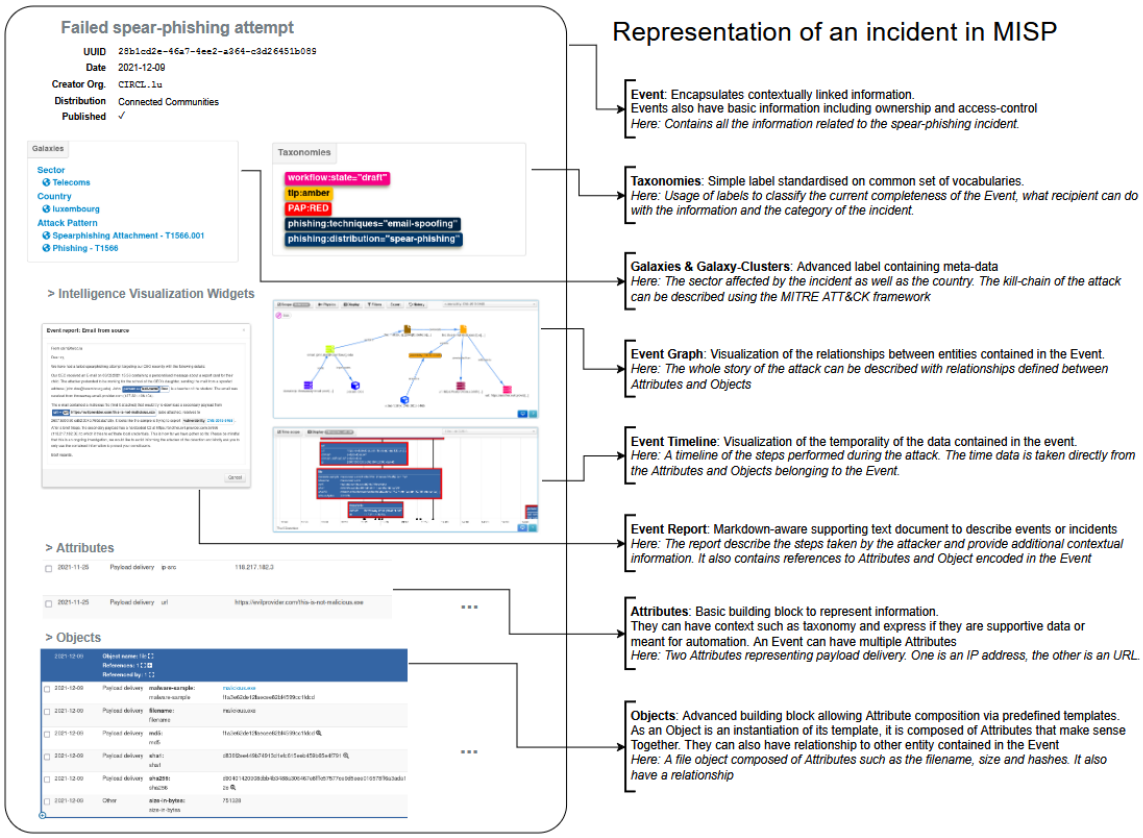
MISP (Malware Information Sharing Platform and Threat Sharing) is an open-source threat intelligence platform that allows you to share, collate, analyze, and distribute threat intelligence.

1.1 Key Features and Capabilities

- **Data Ingestion**: Import and aggregate threat intelligence from various sources
- **Data Structuring**: Organize data using Events, Attributes (IOCs), and Objects
- **Information Sharing**: Share intelligence with trusted partners and communities
- **Taxonomies**: Use standardized classification systems (CAPEC, CVE, MITRE ATT&CK)
- **Data Enrichment**: Add contextual information and threat actor profiles
- **Customization**: Adapt MISP instance to specific organizational needs
- **Integration**: Connect with various security tools and platforms
- **Analysis**: Automatic correlation of threat intelligence patterns
- **Alerting**: Notification system for specific threat events
- **API Access**: RESTful API for programmatic integration

2 Basics of MISP

The platform provides a structured and standardized framework for collecting, storing, and sharing threat intelligence data, enabling collaboration and enhanced defense against cyber threats. It has mappings with existing threat intelligence frameworks (e.g., MITRE ATTCK, CAPEC, etc.) and strong integrations with security products (e.g., CrowdStrike Falcon, Intel471, etc.). MISP is the defacto open-source threat intelligence platform mature organizations use to track threats and collaborate.



3 CHEATSHEET FOR MISP

MISP Data Model Cheat Sheet

- Context such as `Taxonomies` or `Galaxy Clusters` can be attached to the element
- Has a distribution level
- Can be synchronised to/from other instances

Event

Encapsulations for contextually linked information.

Purpose: Group datapoints and context together. Acting as an envelop, it allows setting distribution and sharing rules for itself and its children.
Usecase: Encode incidents/events/reports/...
▶ `Events` can contain other elements such as `Attributes` , `MISP Objects` and `Event Reports` .
▶ The distribution level and any context added on an `Event` (such as `Taxonomies`) are propagated to its underlying data.

Attribute

Basic building block to share information.

Purpose: Individual data point. Can be an indicator or supporting data.
Usecase: Domain, IP, link, sha1, attachment, ...
▶ `Attributes` cannot be duplicated inside the same `Event` and can have `Sightings` .
▶ The difference between an indicator or supporting data is usualy indicated by the state of the attribute's `to_ids` flag.

MISP Object

Advanced building block providing `Attribute` compositions via templates.

Purpose: Groups `Attributes` that are intrinsically linked together.
Usecase: File, person, credit-card, x509, device, ...
▶ `MISP Objects` have their attribute compositions described in their respective template. They are instanciated with `Attributes` and can `Reference` other `Attributes` or `MISP Objects` .
▶ MISP is not required to know the template to save and display the object. However, *edits* will not be possible as the template to validate against is unknown.

Object Reference

Relationships between individual building blocks.

Purpose: Allows to create relationships between entities, thus creating a graph where they are the edges and entities are the nodes.
Usecase: Represent behaviours, similarities, affiliation, ...
▶ `References` can have a textual relationship which can come from MISP or be set freely.

Sightings

Means to convey that an `Attribute` has been seen.

Purpose: Allows to add temporality to the data.
Usecase: Record activity or occurence, perform IoC expiration, ...
▶ `Sightings` are the best way to express that something has been seen. They can also be used to mark *false positives*.

Event Report

Advanced building block containing formated text.

Purpose: Supporting data point to describe events or processes.
Usecase: Encode reports, provide more information about the `Event` , ...
▶ `Event Reports` are markdown-aware and include a special syntax to reference data points or context.

Proposals

Clone of an `Attribute` containing information about modification to be done.

Purpose: Allow the correction or the creation of `Attributes` for `Events` your organisation does not own.
Usecase: Disable the IDS flag, Correct errors
▶ As `Proposals` are sync., if the creator organisation is connected to the MISP instance from where the `Proposal` has been created, it will be able to either *accept* or *discard* it.

Taxonomies

Machine and human-readable labels standardised on a common set of vocabularies.

Purpose: Enable efficent classification globally understood, easing consumption and automation.
Usecase: Provide classification such as: TLP, Confidence, Source, Workflows, Event type, ...
▶ Even though MISP allows the creation of free-text tags, it's always preferable to use those coming from `Taxonomies` , if they exists.

Galaxies

Act as a container to group together context described in `Galaxy Clusters` by their type.

Purpose: Bundle `Galaxy Clusters` by their type to avoid confusion and to ease searches.
Usecase: Bundle types: Exploit-Kit, Preventive Measures, ATT&CK, Tools, Threat-actors, ...

Galaxies Clusters

Knowledge base items used as tags with additional complex meta-data aimed for human consumption.

Purpose: Enable description of complex high-level information for classification.
Usecase: Extensively describe elements such as: threat actors, countries, technique used, ...
▶ `Galaxy Clusters` can be seen as an enhanced `Taxonomy` as they can have meta-data and relationships with other `Galaxy Clusters` .
▶ Any `Galaxy Clusters` can contain the following:

- `Cluster Elements`: Key-Value pair forming the meta-data.

Example: `Country:LU`, `Synonym:APT28`, `Currency:Dollar`, `refs:https://*`, ...

- `Cluster Relations` (): Enable the creation of relationships between one or more `Galaxy Clusters` .

Example: Threat actor `X` `is similar` to threat actor `Y` with `high-likelyhood`.

4 Setting Up MISP

MISP (Malware Information Sharing Platform) is an open-source threat intelligence platform that enables security professionals to collect, share, and correlate threat data efficiently. This section outlines the step-by-step process to install and configure MISP.

4.1 System Requirements

Before installing MISP, ensure that your system meets the following requirements:

- Ubuntu 24.04 or a compatible Linux distribution
- At least 4GB RAM (8GB recommended for production)
- Minimum 2 CPU cores
- 20GB of disk space (more for large-scale deployments)

4.2 Installation Steps

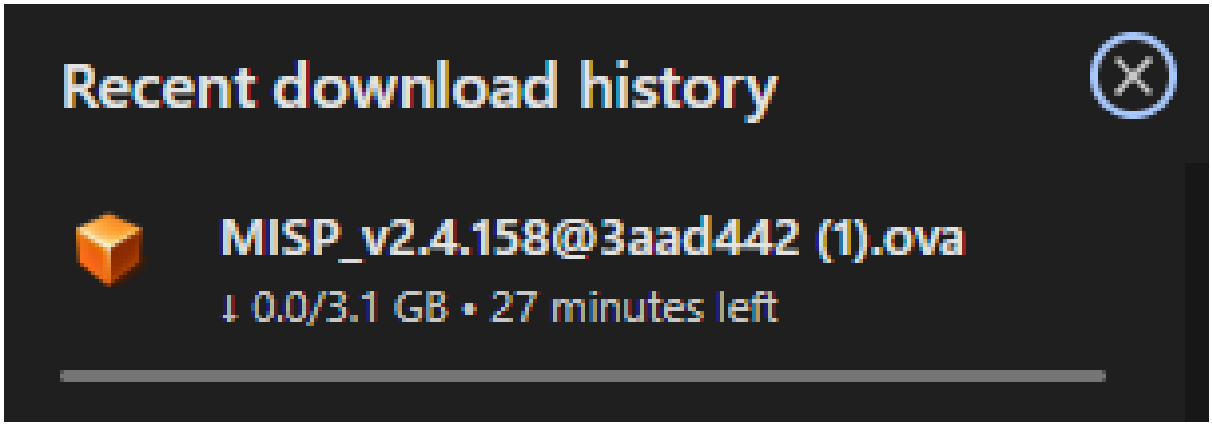
To install MISP, follow these steps:

1. **So, we will start first downloading MISP virtual machine(OVA) from this link:-**

   `https://vm.misp-project.org/latest/MISP_v2.4.158@3aad442.ova`

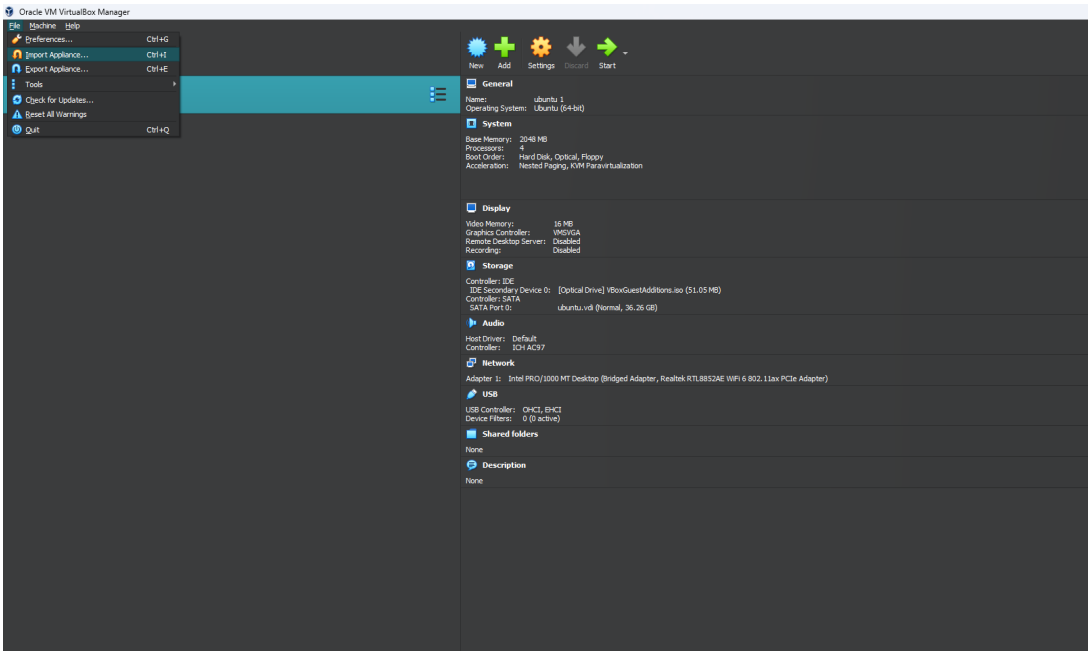
   Once you click the above link ,this will directly start the downloading of MISP VM(virtual machine)

   In your downloads sections you will see something like this

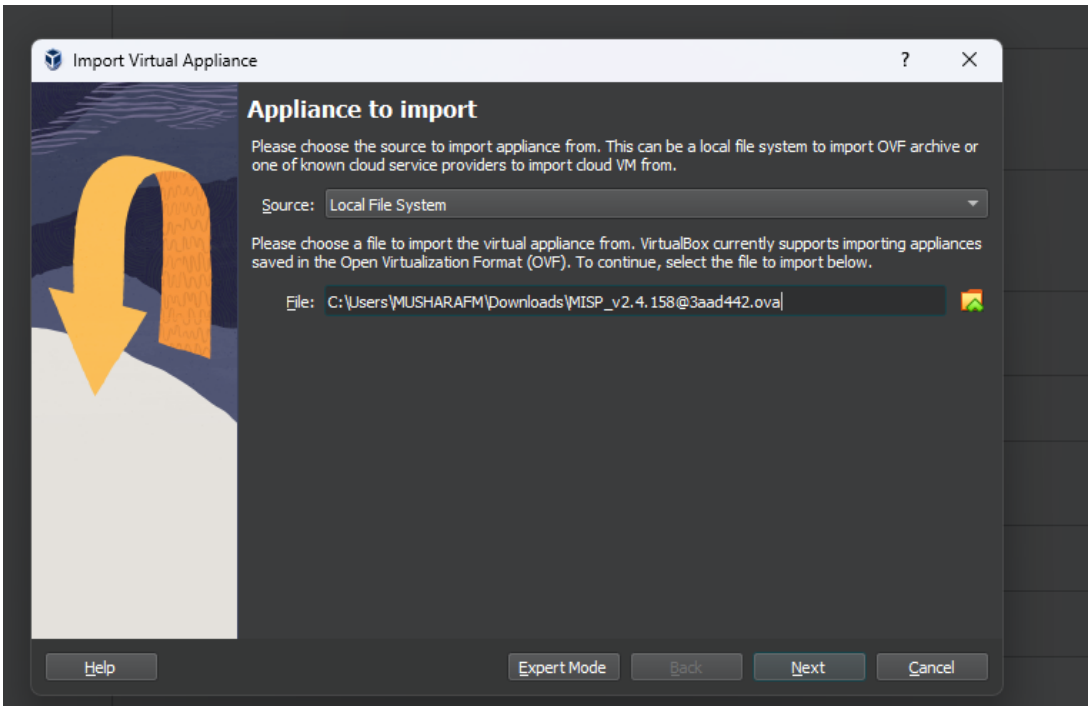
   

2. **After downloading the machine locally we head over to our virtual box. Open Oracle VM VirtualBox.**

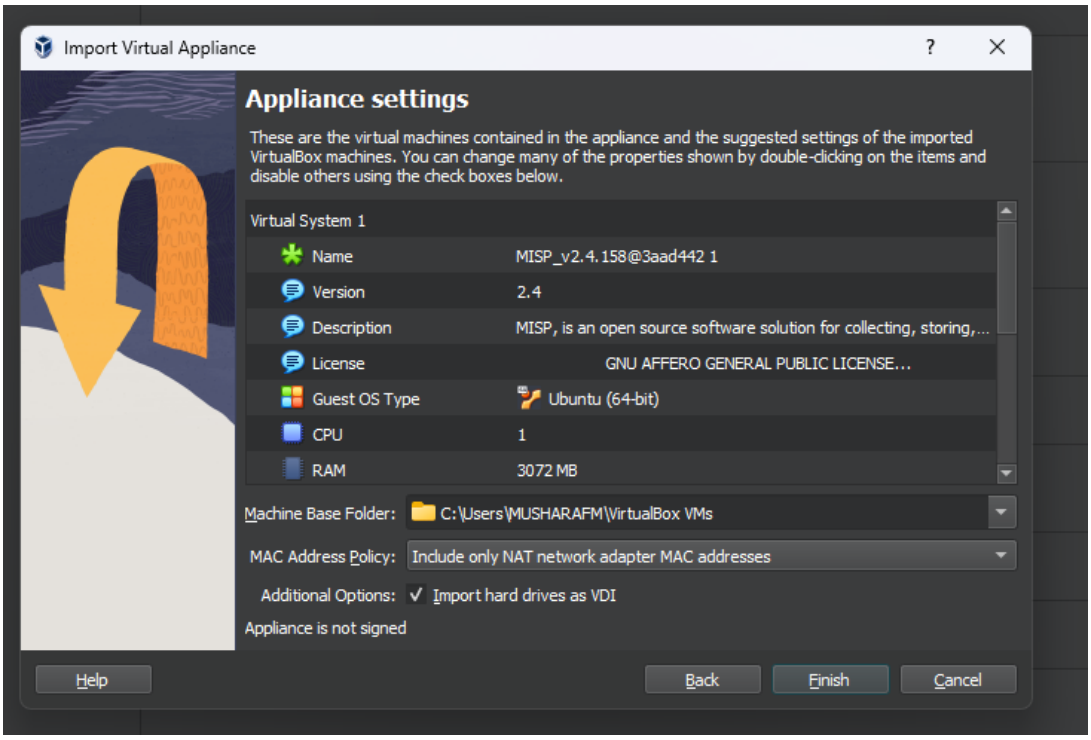
   - Go to File → Import Appliance.



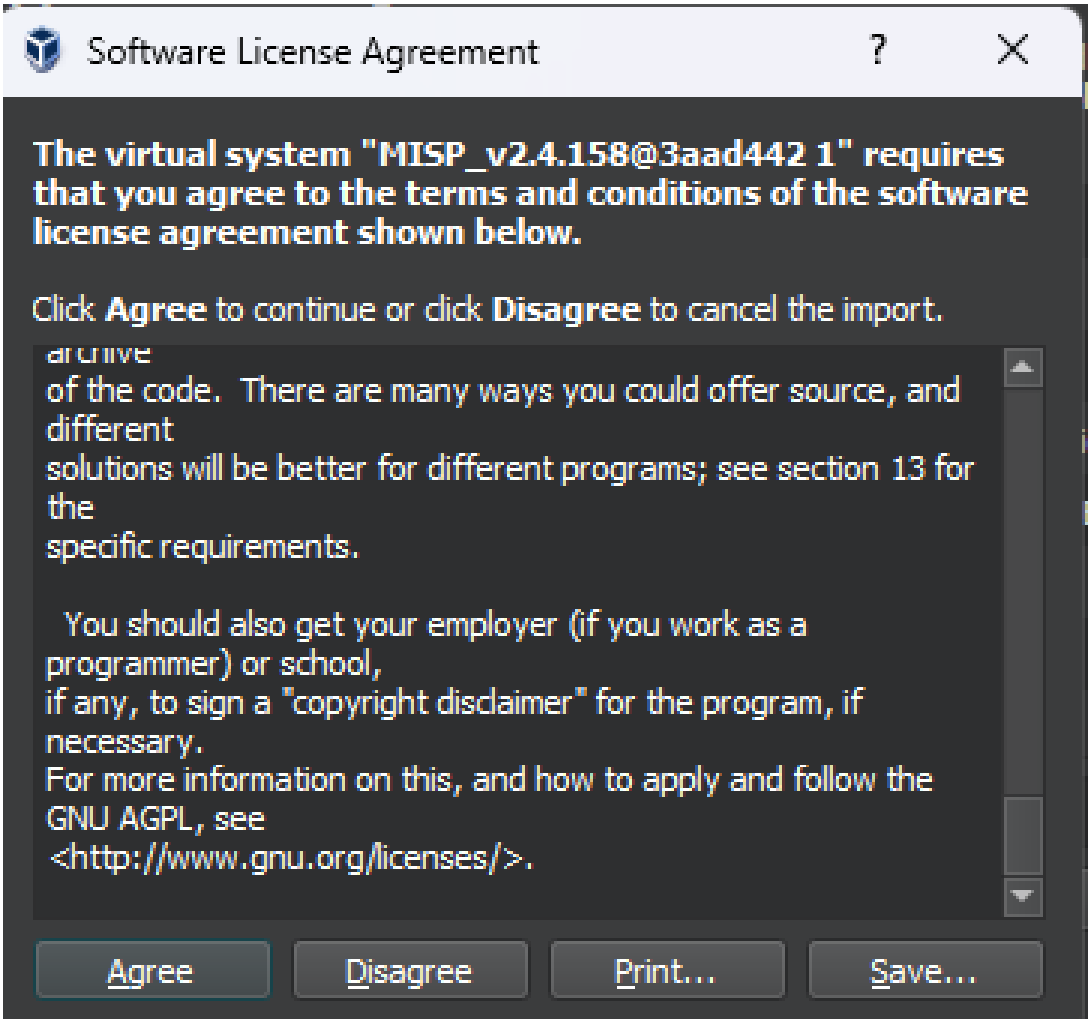
- once you click Import Aplliance, you will see this interface,here press next.



- Click Next, then review the VM settings (CPU, RAM, Disk).



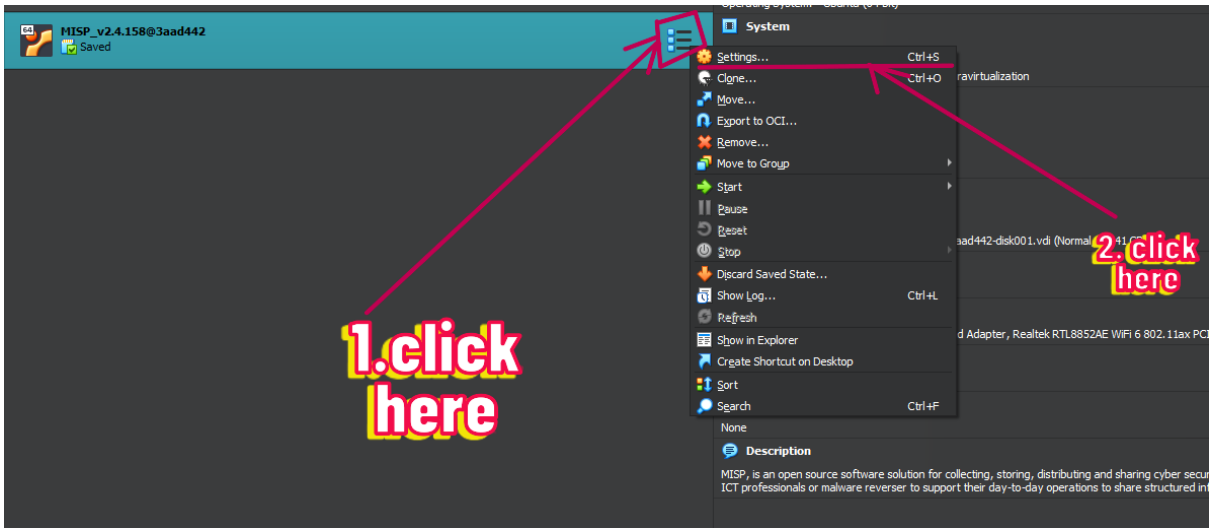
- Click finish and then click Agree, wait for the process to complete.



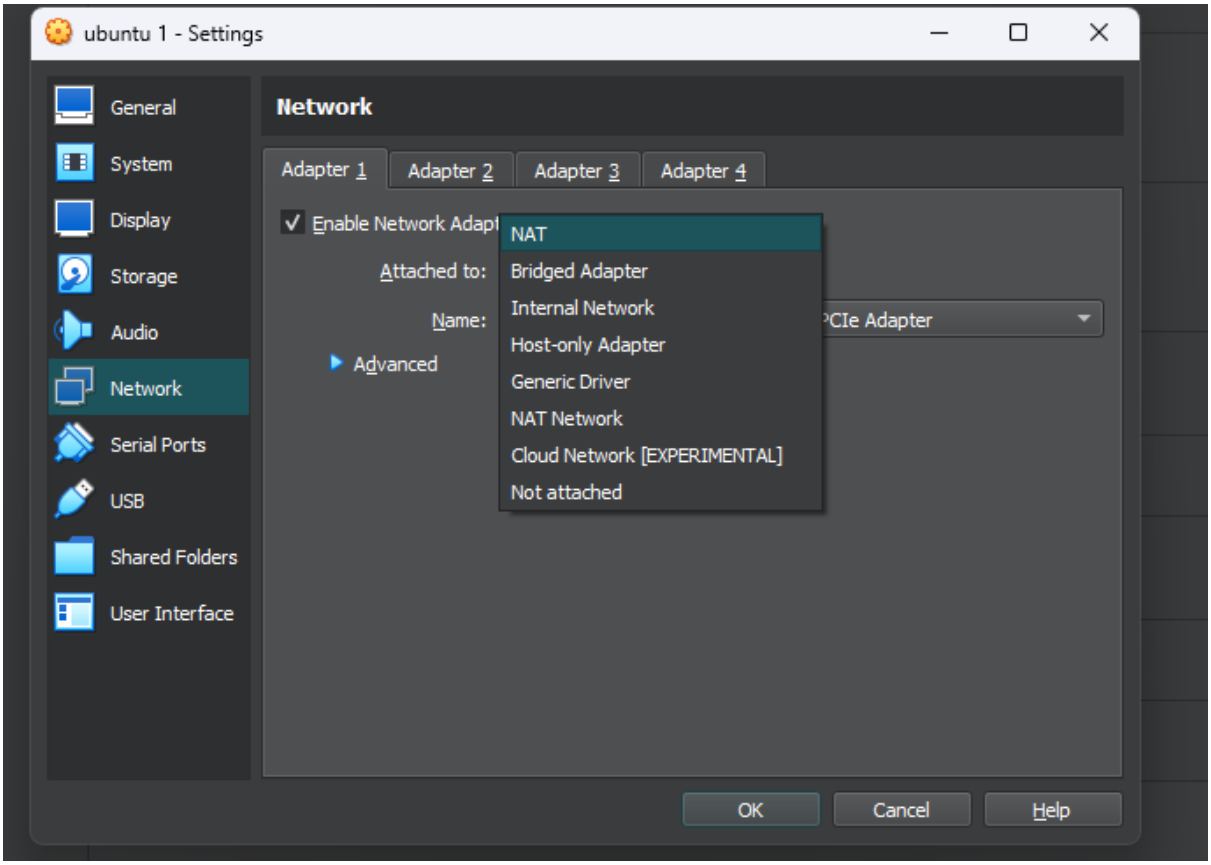
once you are done with the above steps your machine will be ready like this.

4.3 Configure Network Settings

After installation, configure MISP by setting up the following:

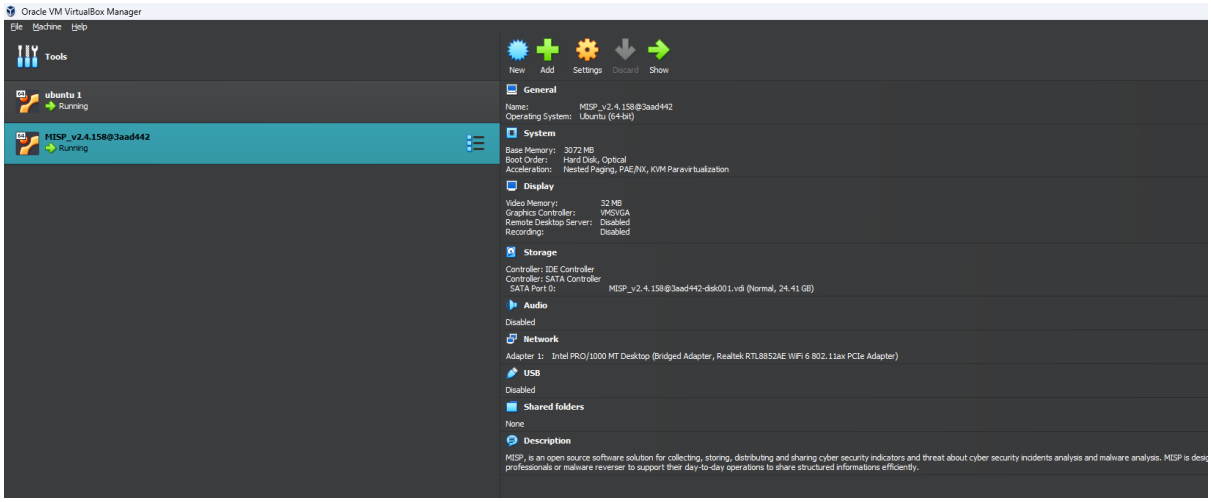


- Go to Settings → Network and choose:
- Bridged Adapter (if you want it on the same network as your host).
- NAT (if you just need internet access).



note: when using a bridged adapter you will be able to access misp locally on your already preinstalled Ubuntu and use NAT if you want to access misp dashboard on MISP VM.

Ok so now you might be confused here why I asked you to have preinstalled Ubuntu with misp VM, you will get the answer in next 2 parts. That is 2 virtual machines in the virtual box like this, one Ubuntu and one misp VM.



part 1:

(a) Now, once you have got all this setup, you will go to Settings → Network and choose Bridged Adapter for both machines.

(b) And once you start the MISP VM, You will see the interface like this.

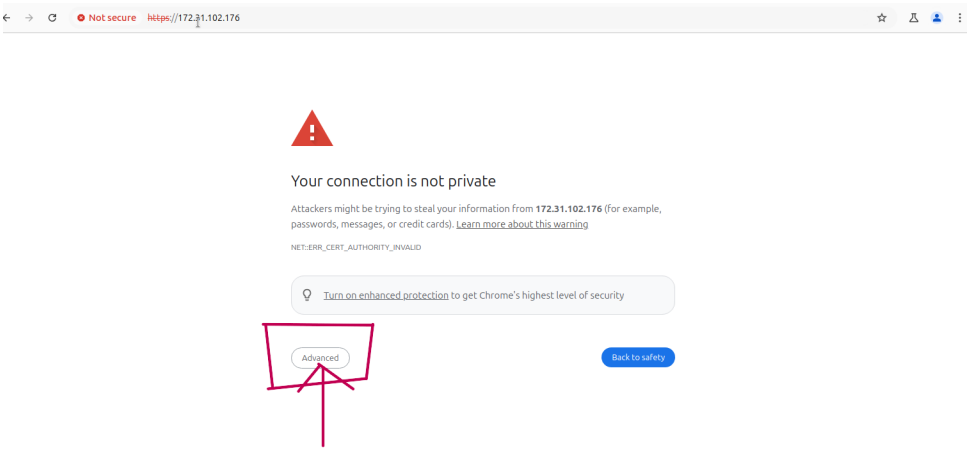
```
Welcome to the MISP Threat Sharing VM.
---

IP address: 172.31.102.176

---

MISP                http://172.31.102.176       admin@admin.test / admin
                    https://172.31.102.176
MISP-modules (API)  http://172.31.102.176:6666  (no credentials)
MISP-dashboard      http://172.31.102.176:8001  (no credentials)
Viper-web           http://172.31.102.176:8888  admin / Password1234
jupyter-notebook    http://172.31.102.176:8889

The default system credentials are: misp / Password1234

On VirtualBox port-forwarding from your host to the guest is in place.
Below are the forwards as we need to use ports >1024 for some.

MISP         -> 8080 and :8443
ssh          -> 2222
misp-modules -> 1666

If this fails, make sure the host machine is not occupying one of the forwarded ports or a firewall is active.

----
misp login: misp
Password:
Last login: Fri May 13 15:04:13 CEST 2022 from 192.168.151.1 on pts/0
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-177-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
misp@misp:~$ tiem
Traceback (most recent call last):
  File "/usr/lib/command-not-found", line 27, in <module>
    from CommandNotFound.util import crash_guard
ModuleNotFoundError: No module named 'CommandNotFound'
misp@misp:~$ time

real    0m0.000s
user    0m0.000s
sys     0m0.000s
misp@misp:~$ date
Wed Jan 29 08:18:19 CET 2025
misp@misp:~$
```

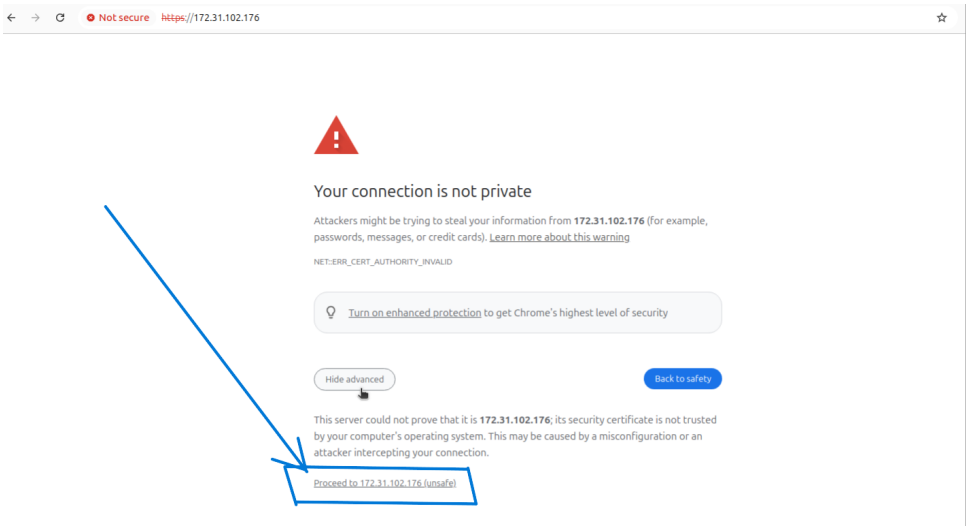
here you will have to login first with the following credentials as shown in dashboard

(c)
- MISP login →MISP
- MISP password →Password1234

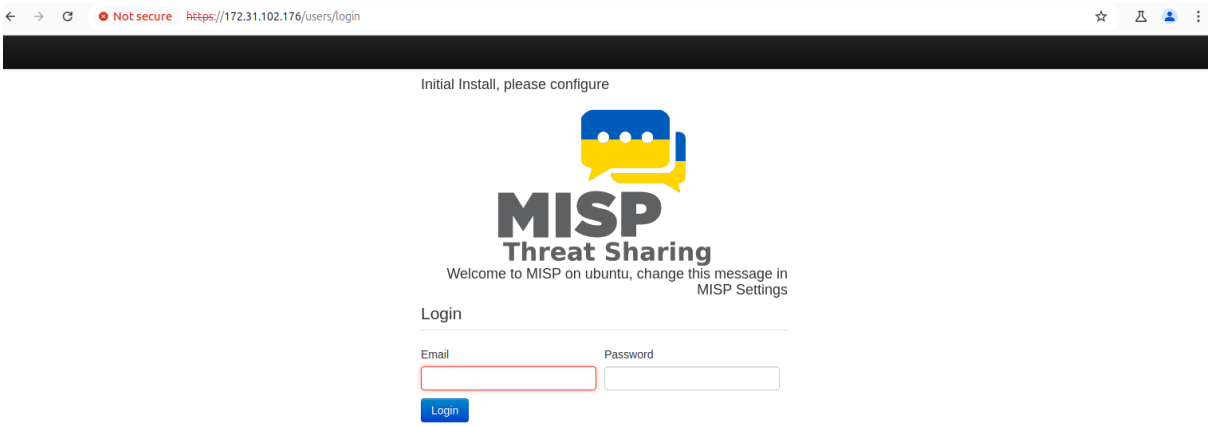
(d) once you are logged in successfully, you can take the IP address of MISP dashboard and the put it in your browser inside your UBUNtU.

(e) **After obtaining the IP address of the MISP dashboard, open Firefox or any other browser in Ubuntu and enter the MISP dashboard IP in the browser's address bar. This will allow you to access the MISP web interface.**





once you click **proceed to ip-address** you will the MISP dashboard Interface:



here you can enter misp default login credentials as **email →admin@admin.test and password →admin**

part 2:

(a) Now, the question arises why not access misp dashboard in the misp VM itself?

(b) While this is possible, the MISP VM runs on a minimal Ubuntu/Debian-based system without a graphical user interface (GUI) by default. This means it lacks a web browser like Firefox to access the MISP dashboard.

(c) **To enable direct access within the MISP VM, you would need to:**

i. Install a GUI (such as XFCE or LXDE).

ii. Install a display manager (such as LightDM).

iii. Install Firefox to browse the MISP dashboard.

(d) However, the recommended approach is to keep the MISP VM lightweight and dedicated to running MISP, while using a separate Ubuntu VM with a full desktop environment for easier access via a browser.

This setup ensures:

Better performance for MISP, as it remains a dedicated server.
Easier troubleshooting and maintenance.

A familiar browsing experience without additional setup inside the MISP VM.

(e) That's why we configure two VMs:

i. MISP VM → Runs the MISP server.

ii. Ubuntu VM → Accesses the MISP dashboard via Firefox.

OPTIONAL **If you still prefer to access MISP directly from the MISP VM, follow the steps to install a GUI and Firefox. However, for most users, using a separate Ubuntu VM is the more practical choice.**

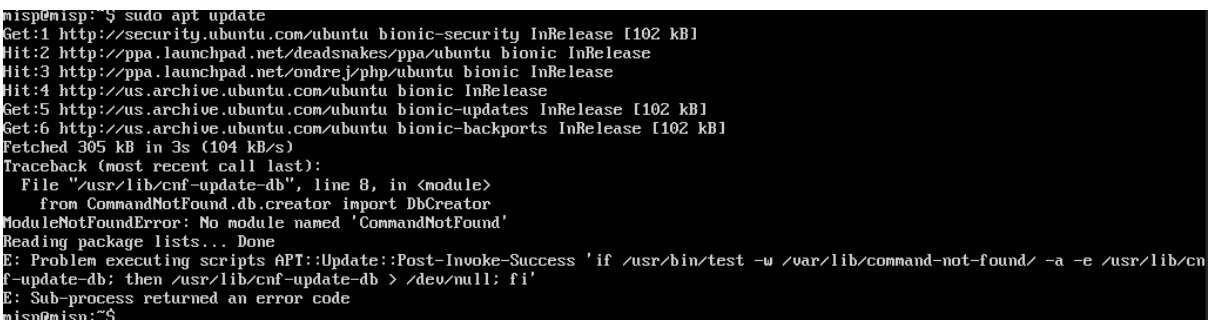
4.4 Access MISP directly from the MISP VM

4.4.1 Step 1: Install a Lightweight Desktop Environment

A lightweight desktop environment like XFCE or LXDE is ideal for running a GUI in the MISP VM without consuming excessive resources.

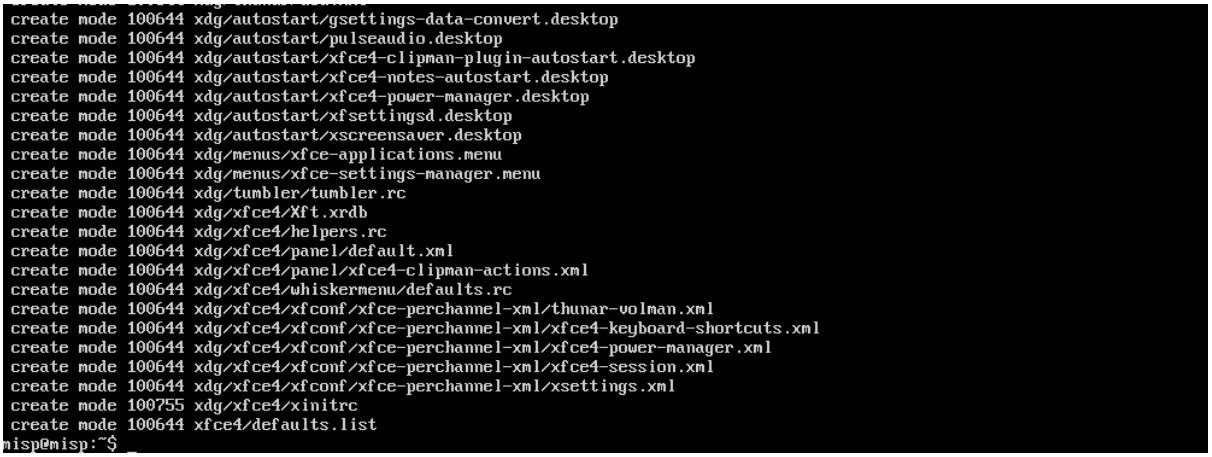
(a) **Run the following command to install XFCE:**

```
sudo apt update
```



```
sudo apt install xfce4 xfce4-goodies -y
```

you should see something like this upon successful execution of the above command:

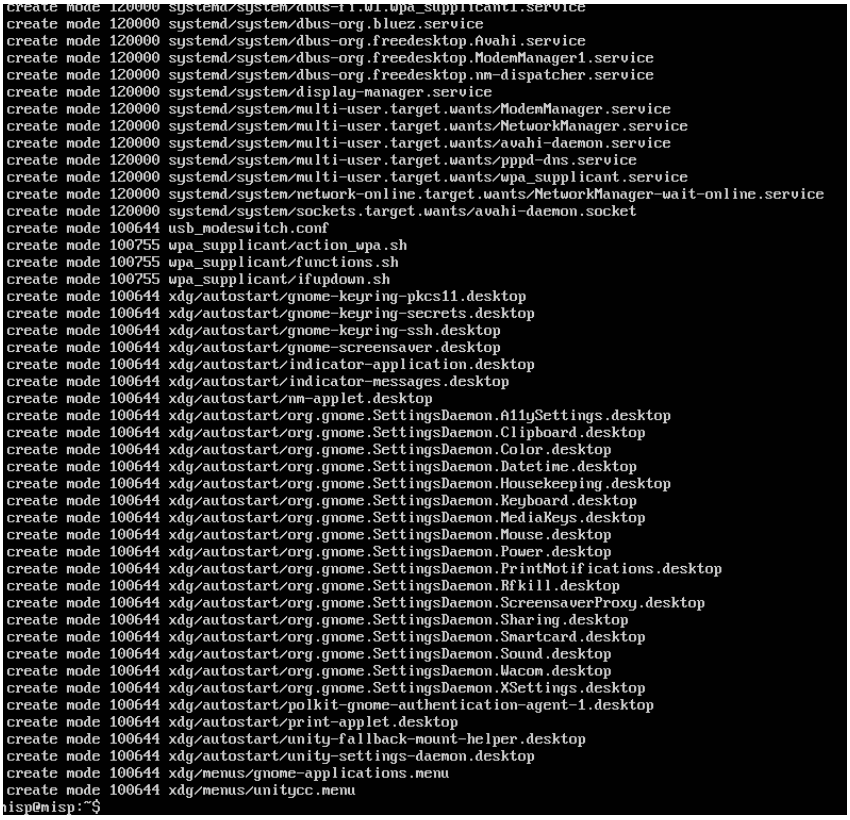


4.4.2 Step 2: Install a Display Manager

(b) **To manage graphical sessions, install LightDM as the display manager:**

```
sudo apt install lightdm -y
```

you should see something like this upon successful execution of the above command:

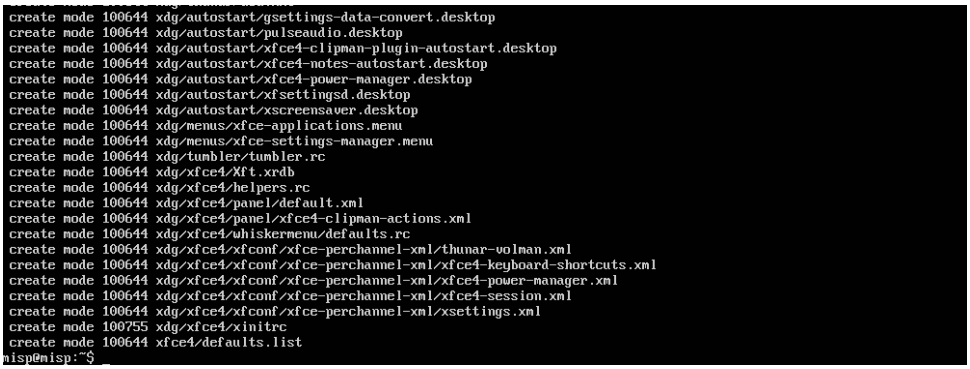


4.4.3 Step 3: Install Firefox

(c) **If Firefox is not already installed on your MISP VM, install it using:**

```
sudo apt install firefox -y
```

you should see something like this upon successful execution of the above command:

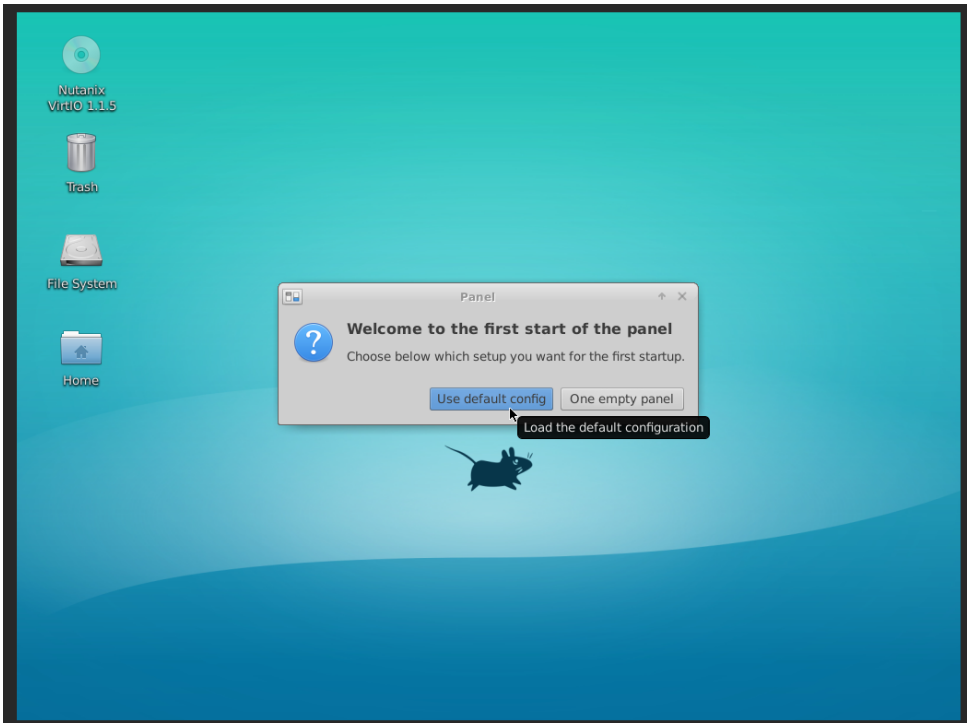


4.4.4 Step 4: Start the GUI

(d) **After installing the desktop environment, start the GUI manually by running:**

```
startx
```

you should see something like this upon successful execution of the above command:



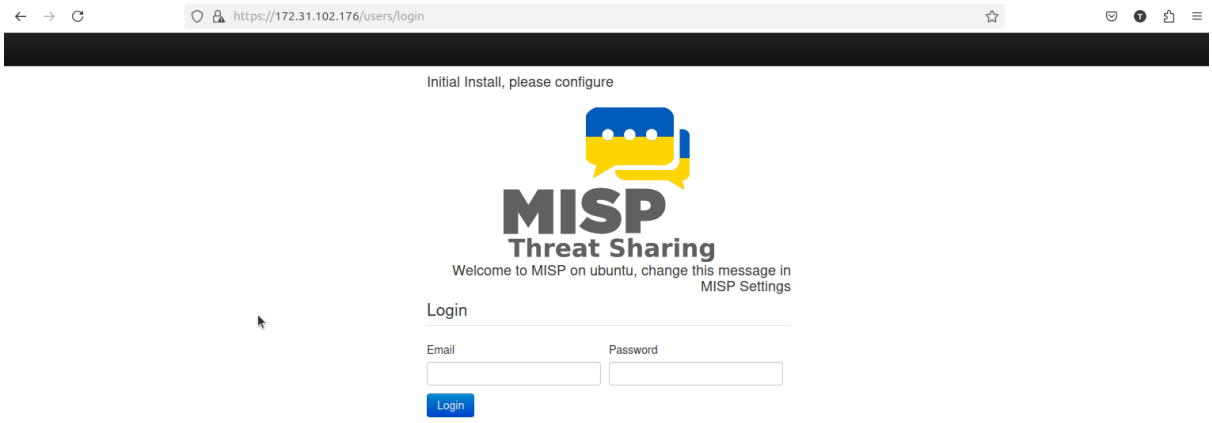
here click **Use default confg**

(e) **Once the GUI starts, you should see the XFCE desktop environment. To open Firefox:**

(f) **Open the terminal in the GUI and run:**

```
firefox
```

So, after following these two methods you will be able to access misp dashboard locally using its IP address.



here, you will use default log in credentials:-

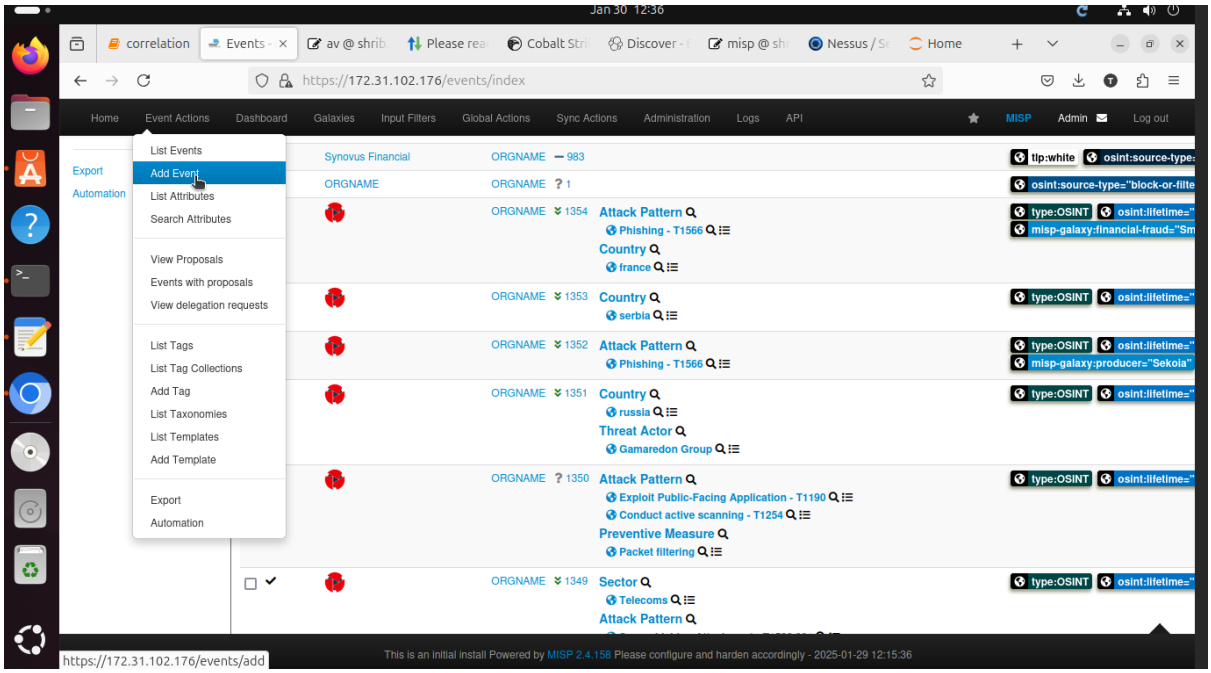
- Email:- admin@admin.test
- password:-admin
- This completes our misp setup

5 EVENT CREATION

5.1 What is an Event

MISP events are "encapsulations for contextually related information represented as attribute and object." They can be threat intelligence articles, malware analysis reports, threat research, or any other way you can think of representing threat intelligence. Events are the individual containers that group your atomic pieces of threat intelligence with contextual information so analysts can actually use it. They are the main way of interacting with data in MISP.

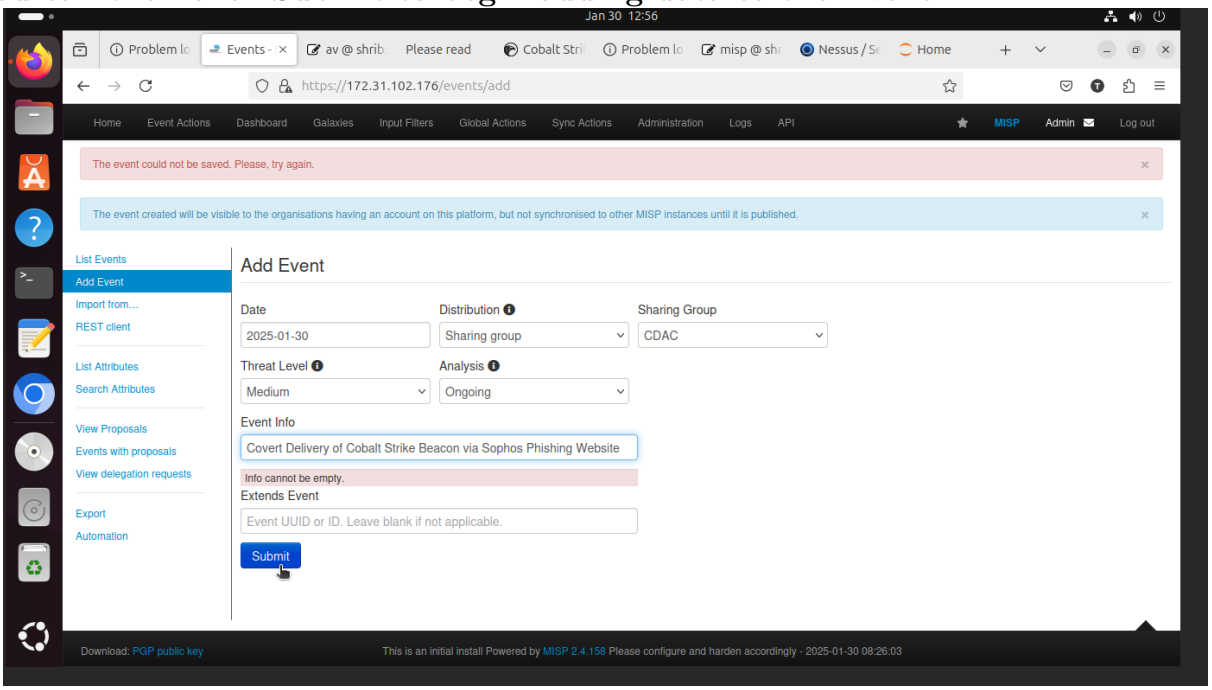
5.2 First, to add an Event in MISP, go to Event Actions + Add Event:



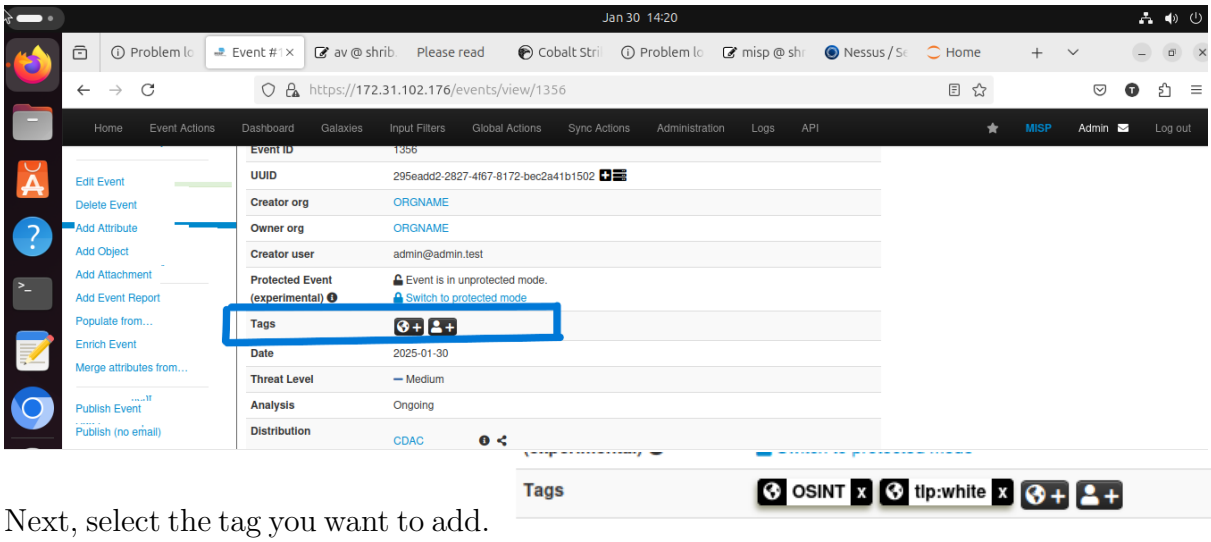
5.3 Next, fill out the metadata about the Event:

- **Distribution:** How you want your event to be shared across MISP instances (your organization only, this community only, connected communities, or all communities).
- **Threat Level:** The sophistication or danger the threat poses (high, medium, low, or undefined).
- **Analysis:** Which stage of analysis this event is at currently (initial, ongoing, or complete).
- **Event Info:** A title, brief description, or identifier to track the event.
- **Extends Event** If the Event is related to a previous Event, you can link it by the UUID or ID of that previous Event.

You can then click Submit to begin adding data to the Event.

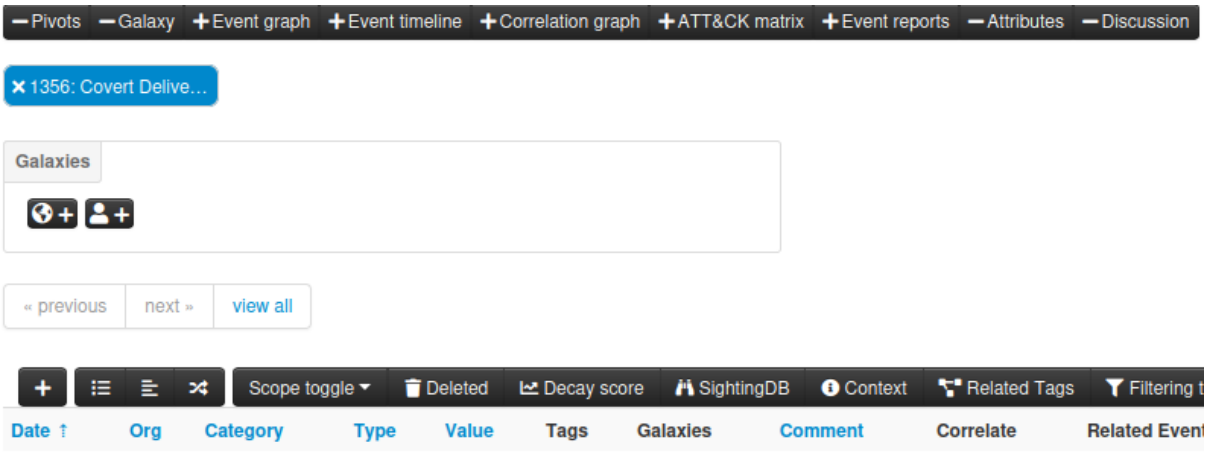


5.4 Adding Tags

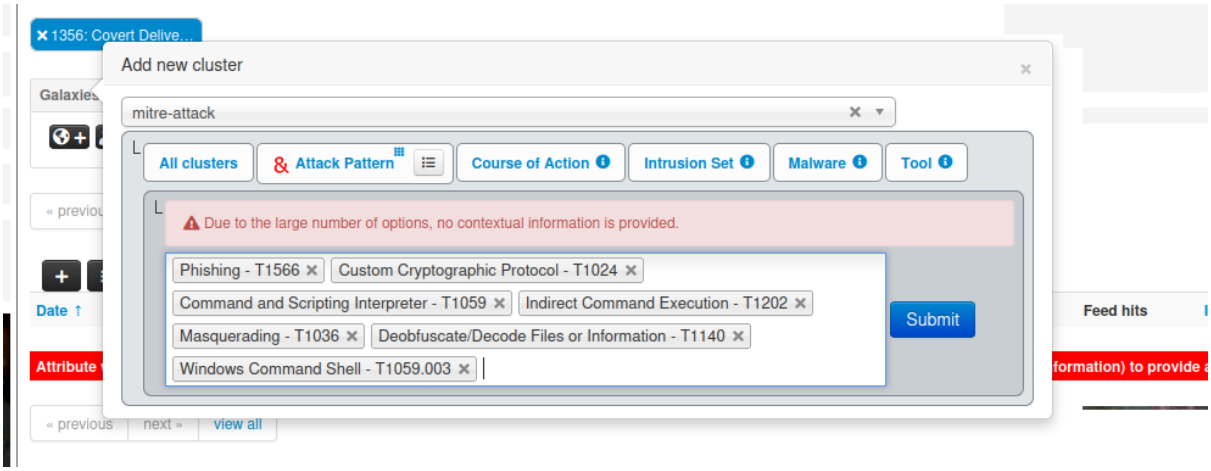


Next, select the tag you want to add.

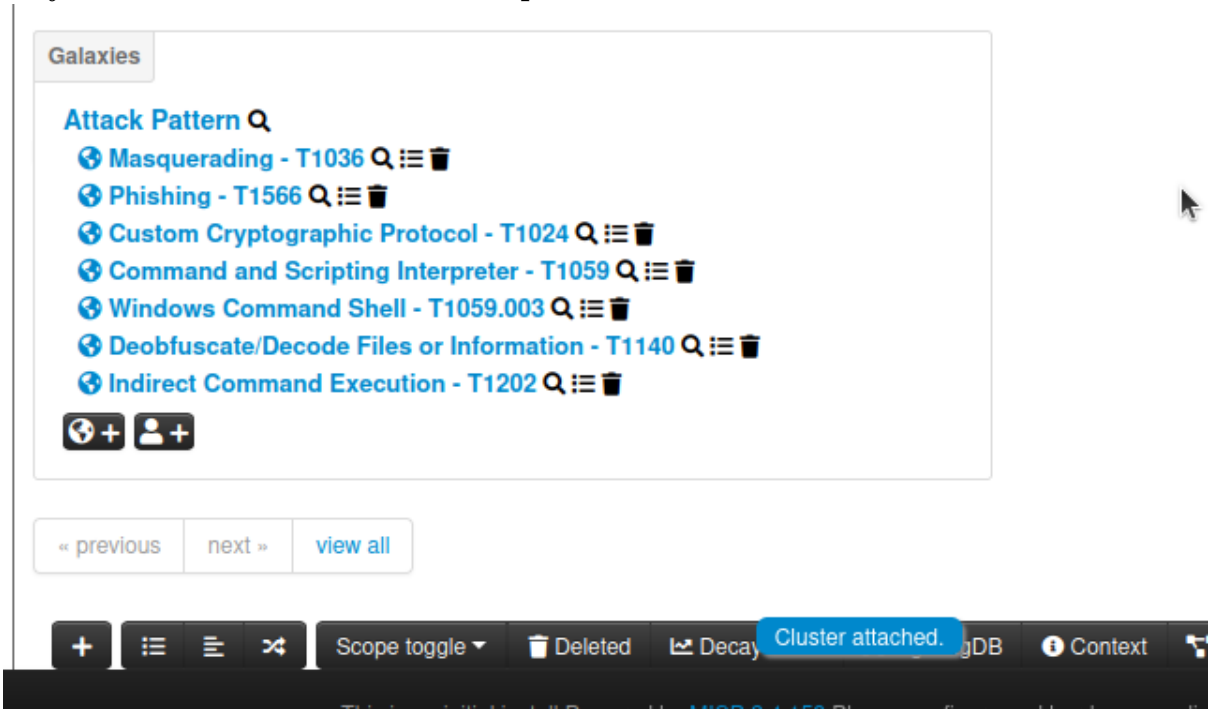
5.5 Adding Galaxy Clusters



You can attach a MISP galaxy as a cluster to add more context to an Event. Galaxies allow you to describe an Event using shared threat intelligence frameworks or standardized lingo. For instance, you can add the name of malware seen in an Event, map the Event to MITRE ATTCK technique, or add a threat actor.



Once you have added all the techniques and clicked Submit



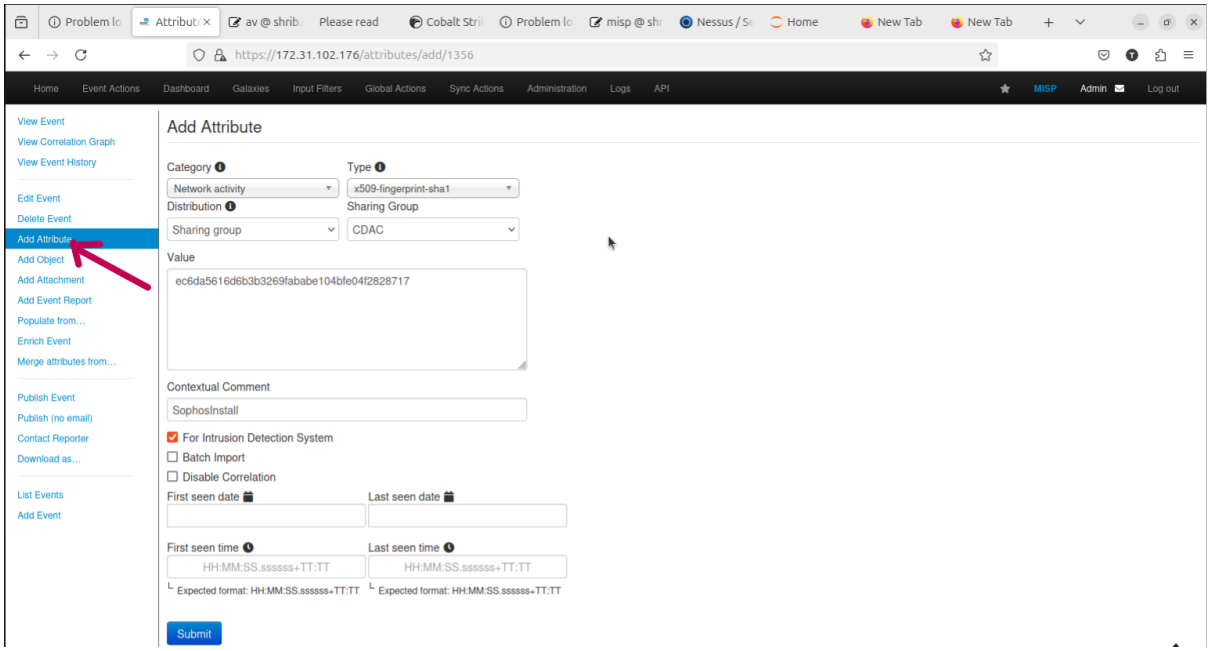
you will see them populated in the Event view.

6 Populating an Event

MISP attributes are atomic pieces of intelligence, such as network indicators (e.g., IP addresses, domains, URLs), system indicators (a string in memory, a file hash, etc.), or even a Bitcoin wallet. An attribute will have a type that describes it and an overarching category that puts it in context (making it intelligence rather than just data). You can add the Intrusion Detection System (IDS) flag to an attribute to determine if an attribute can be automated (such as being exported as an IDS ruleset or used for detection).

6.1 Adding MISP Attributes

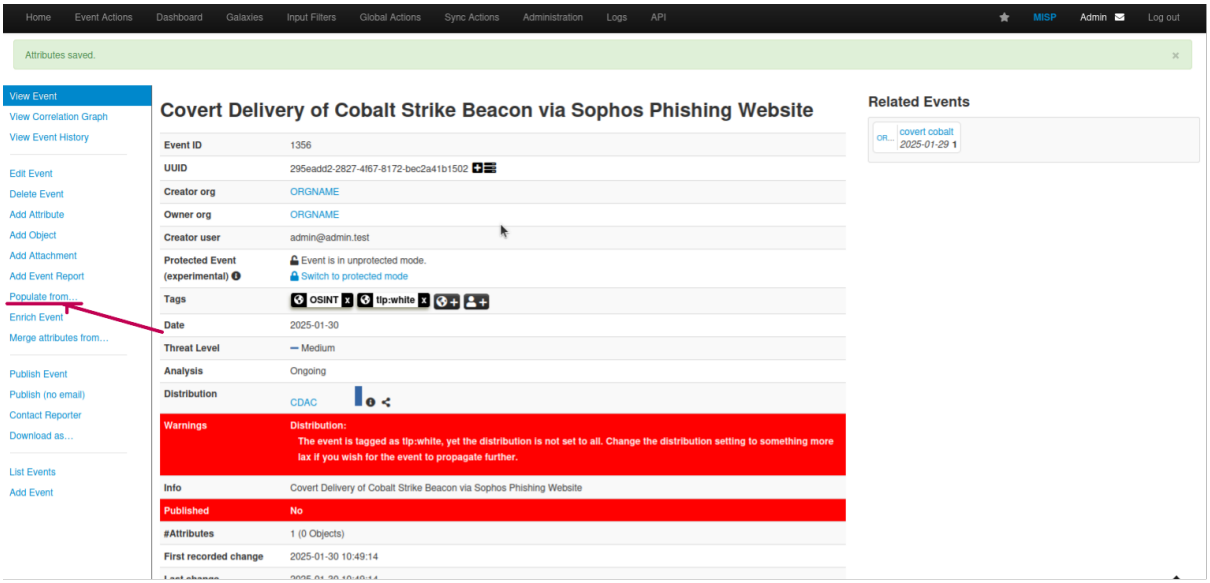
- To add an attribute to an Event, select the Add Attribute button from the left-hand menu.
- You can then fill out the Add Attribute web form to provide context to the attribute you add. First, choose what Category the attribute falls under, then choose the Type of attribute and how you want to share this attribute (Distribution). You can then copy and paste the Value across.



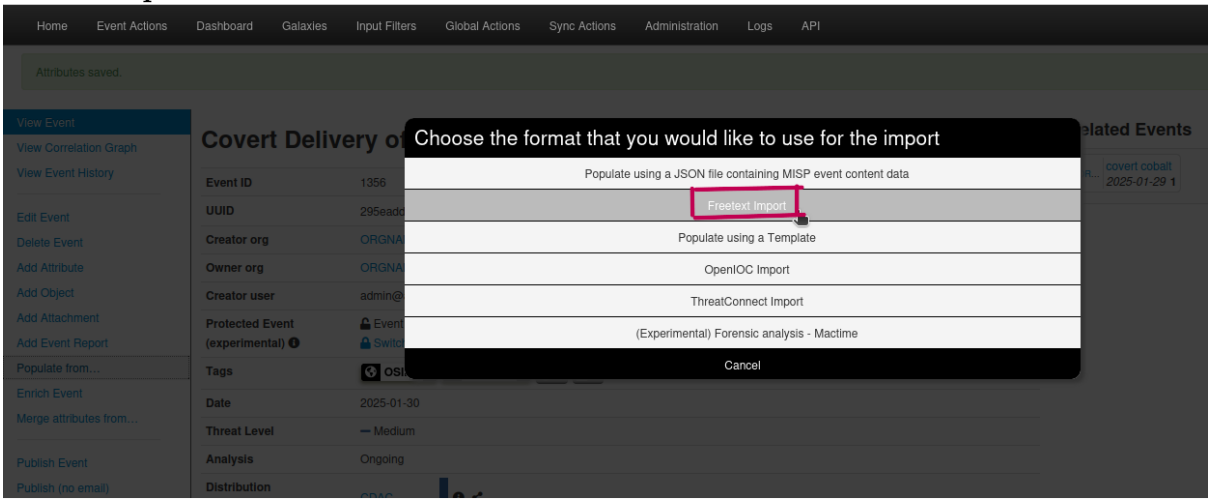
When you are done filling out the attribute details, click the Submit button. This will add the attribute(s) you created to the MISP Event. Scrolling down the View Event page, you can see these attributes listed. You can add attributes directly from this view by clicking the Add attribute button (+ icon) here.

6.2 Automated Data Upload

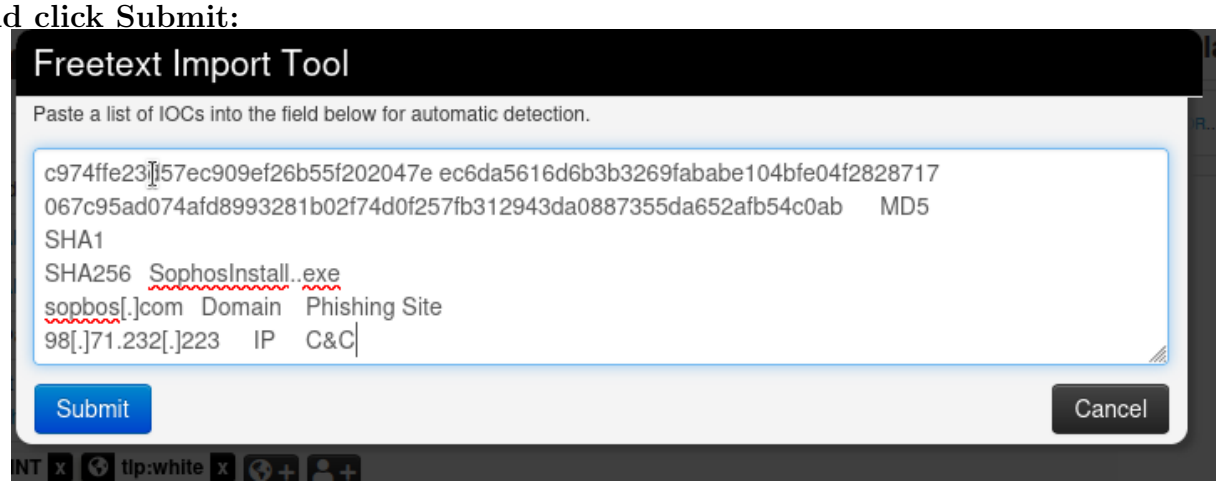
- Although you can create objects and attributes manually, MISP offers easier ways to add intelligence to an Event.
- Select Populate from in the left-side menu brings up a popup window where you can choose to populate the MISP Event using automated methods



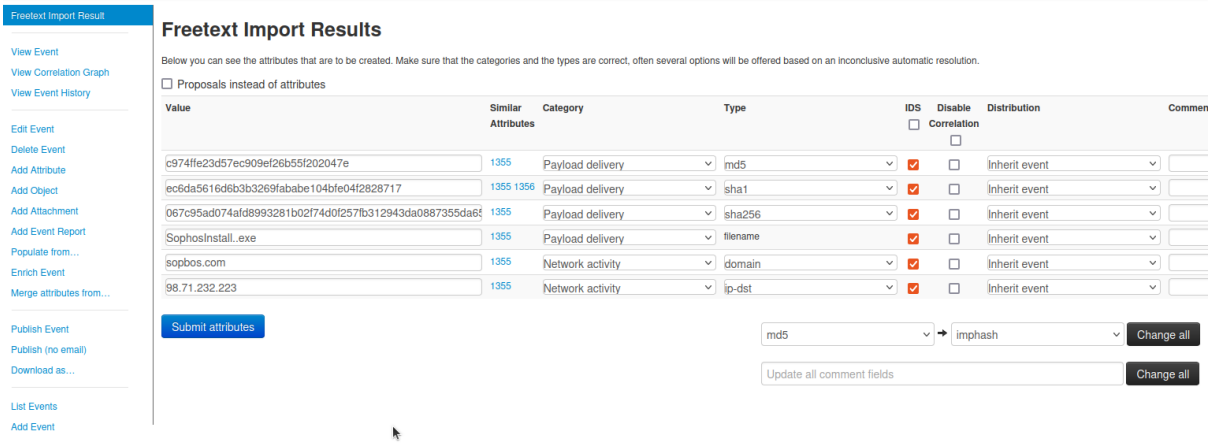
The two most common import methods you are likely to use are Popuate using a Template which lets you use a custom template to make it easier to upload attributes to MISP, and the Freetext Import. Let's take a look at the Freetext Import tool.



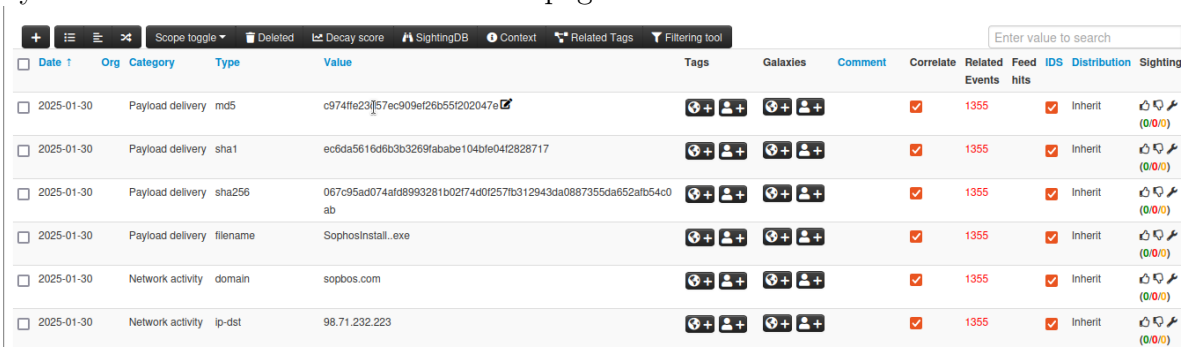
Copy and paste the IOCs into the Freetext Import tool from your source and click Submit:



The Freetext Import tool will parse these IOCs into legitimate MISP attributes. However, it might not get it right every time. As such, the tool lets you change your attributes before adding them to your Event in a review screen.



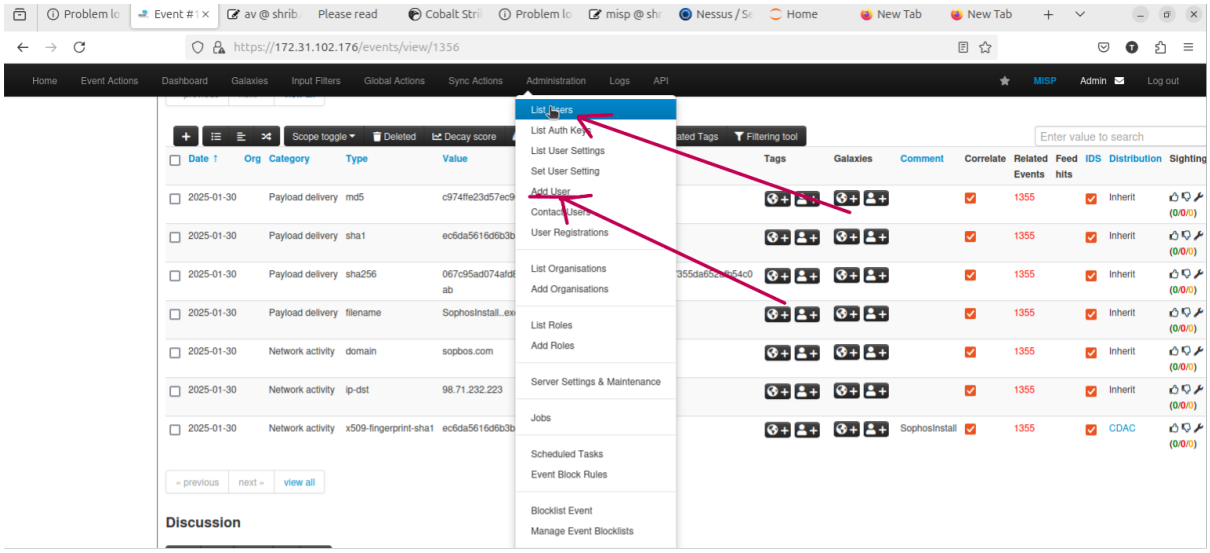
Clicking the Submit attributes button will add the attributes to the MISP event, which you can then view on the Event View page.



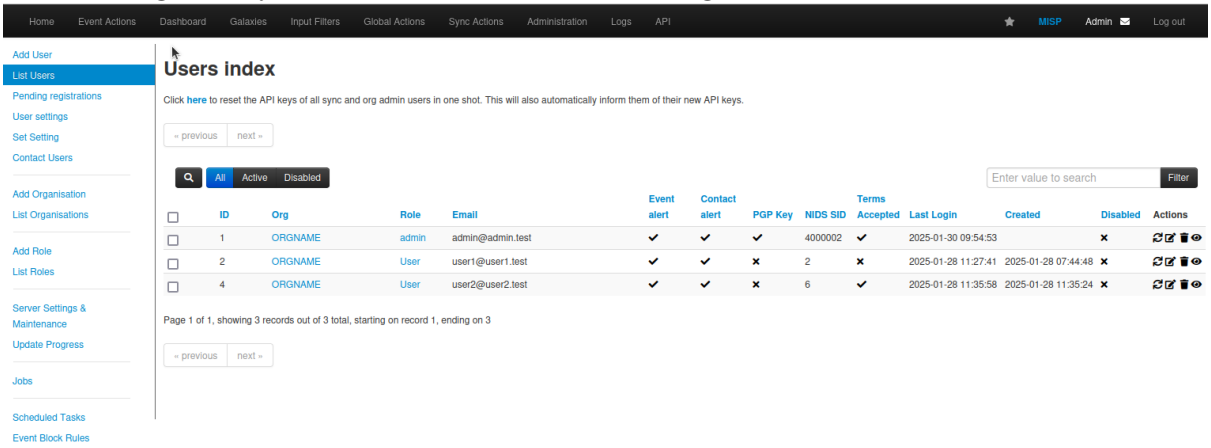
MISP's Freetext Import option is a powerful tool that makes it very easy to upload a large number of IOCs in one go, rather than having to manually upload individual attributes or objects. The tool also lets you customize how attributes are uploaded. For instance, you could have created a file object out of those file hashes to link them to the same filename using the Create object button. I highly recommend using this tool, or creating a custom template, when uploading data to MISP.

7 Creating Multiple Users and allowing them to communicate on a shared malware

1. Log in to your MISP instance as an administrator.
2. Navigate to **Administration** →**List Users**.
3. Click on **Add User** in the top-right corner.
4. Fill in the user details:
   - **Email:** Enter the user's email address.(user@user.test)
   - **Organisation:** Select the appropriate organization.(shared grouop - CDAC)
   - **Role:** Choose the role (User), as admin is laready registered
   - **Password:** Set a strong password.
5. Click **Submit** to create the user.
6. Repeat the above steps for each additional user.



after adding users you can now see them on clicking list user



8 Decaying Model Tools

Decaying models help prioritize threat intelligence over time by reducing the relevance of older data. Several tools can be used for implementing decay models in MISP.

8.1 Step 1: Understanding Decaying Models

Decaying models assign a confidence score to attributes based on their age and impact. This helps analysts focus on the most relevant data.

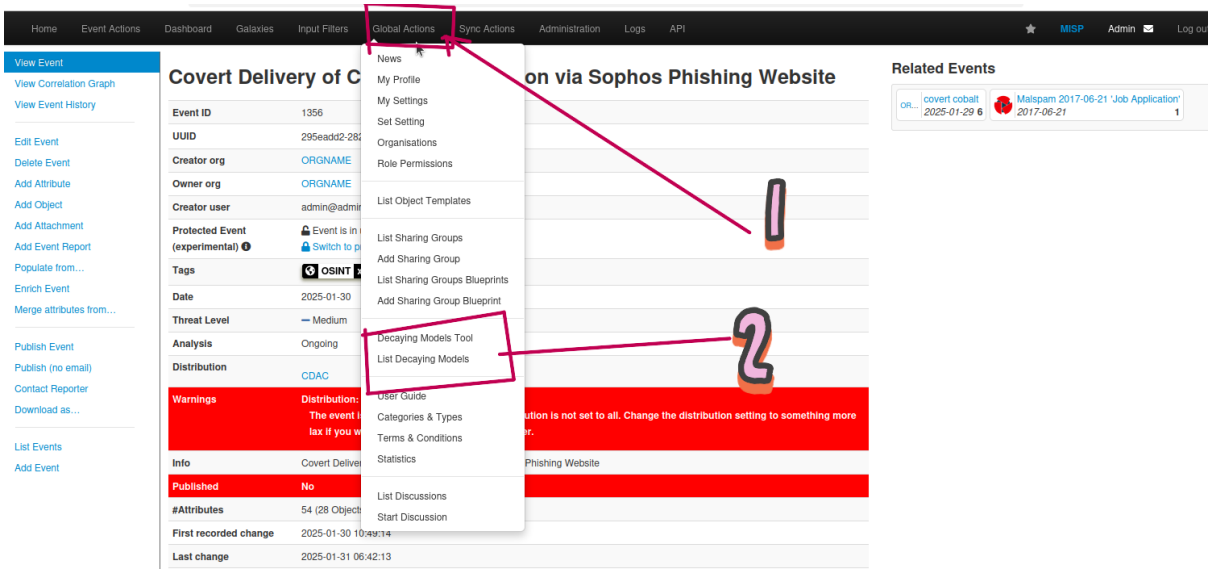
8.2 Step 2: Available Decaying Tools

- **MISP Decaying Model:** Built-in MISP feature that allows attribute decay configuration based on event types.
- **Custom Decay Scripts:** Python scripts can be used to create custom decay logic tailored to organizational needs.
- **Machine Learning Models:** Advanced models can be trained to predict decay patterns based on historical data.

8.3 Step 3: Configuring Decay in MISP

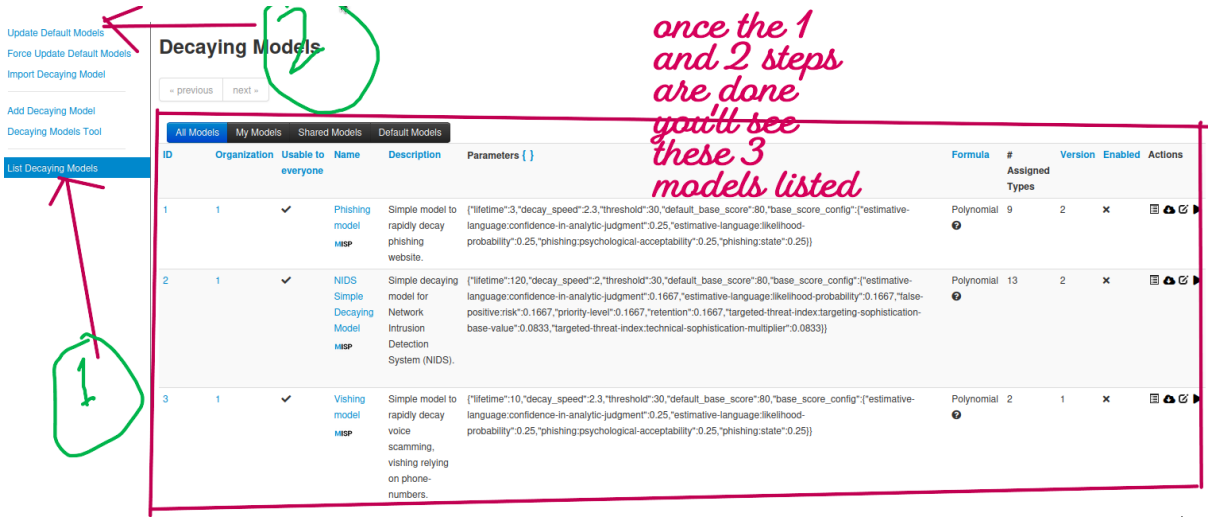
To configure decay models in MISP:

1. Navigate to **Global Actions →decaying models list**.

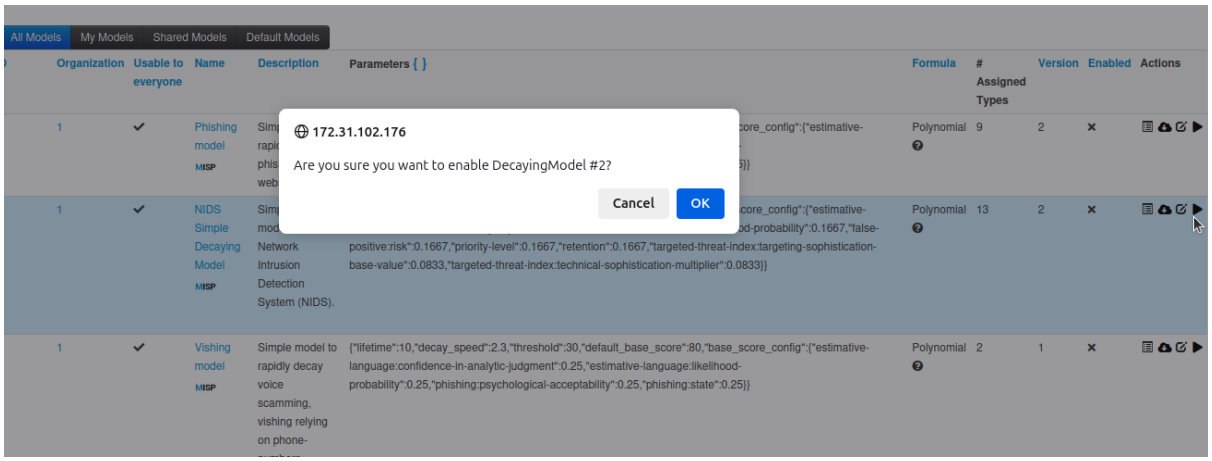


2. Locate the **Decaying Model** settings.

3. Click the Update Default Models options from the left-side menu to load the default decaying models that come with MISP.

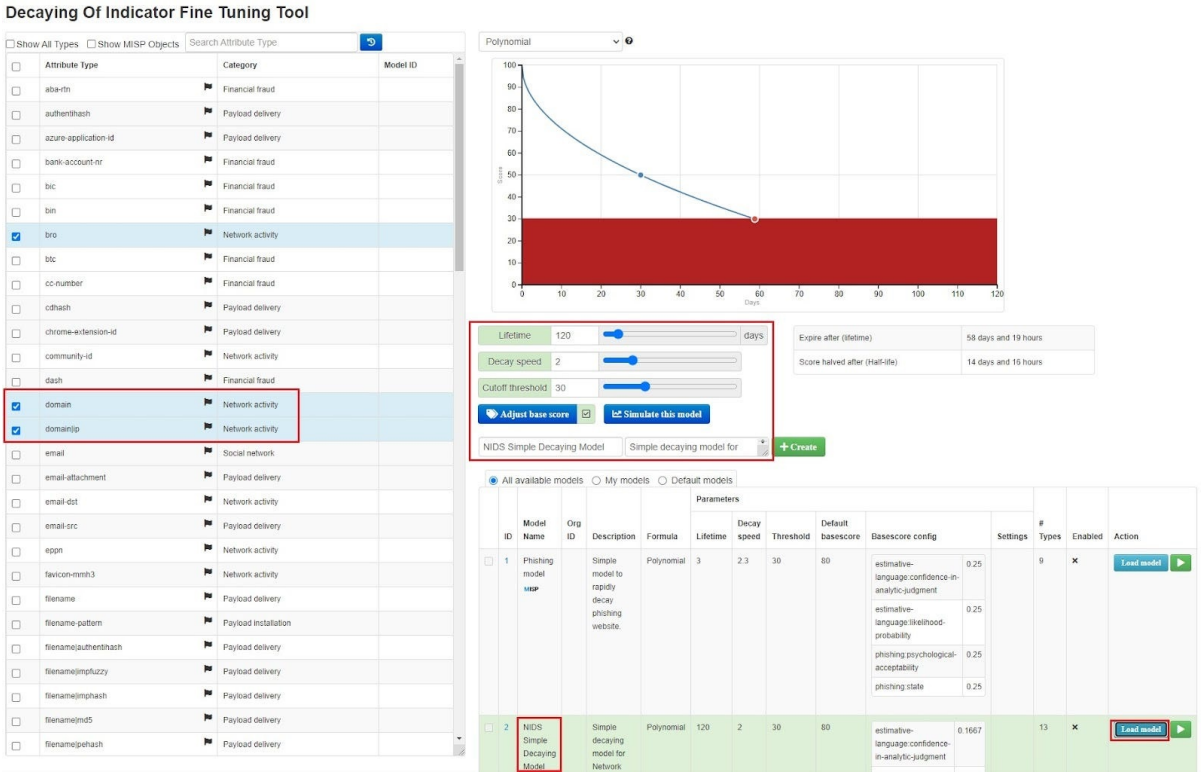


4. This will populate the Decaying Models screen with a list of the default models that come with every MISP install. You can use the buttons under the Actions column to view, download, edit, and enable the models.

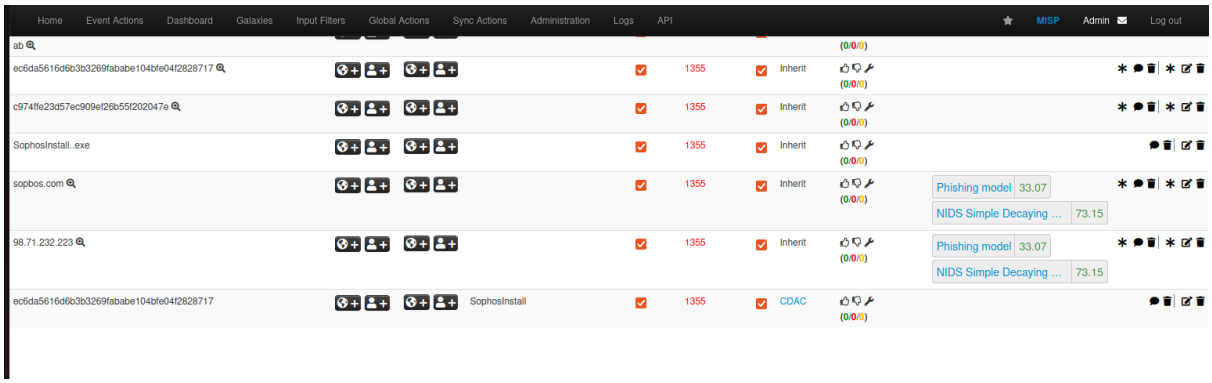


5. You can add your own decaying model or import one using the options on the left-side menu. However, for now, we will just apply a default one to our MISP attributes. To do this, first select the Decaying Models Tool option in the left-side menu.

6. You can choose to select multiple attributes, assign these factors to them, and then create your own decaying model using the Create button. However, using one of the default decaying models MISP provides is the easiest way. To do this, scroll down to the model you want to use and select Load model. This will load the model into the fine tuning tool, where you can see which attributes it affects and how it affects them. Here, I have selected the NIDS Simple Decaying Model, which affects network indicators.



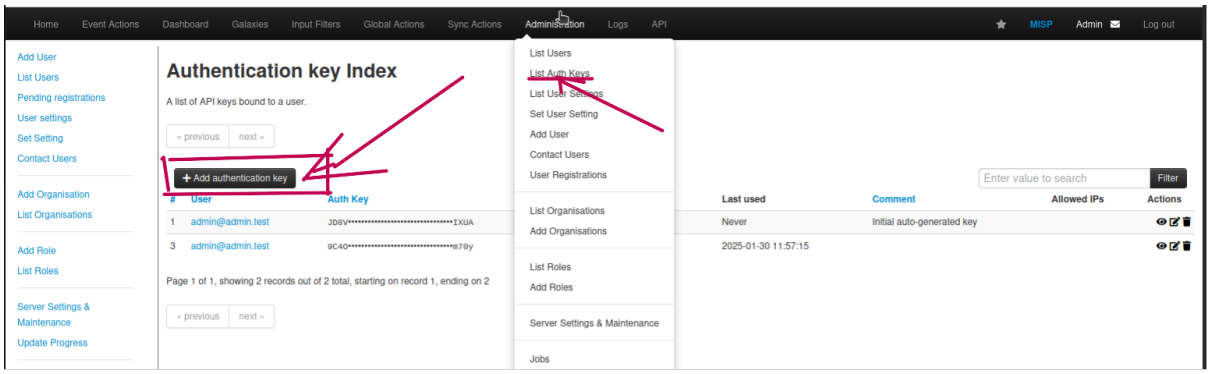
7. Once enabled, you will now see a Score field next to the MISP attributes that relate to the decaying model you enabled. This reflects the current value an attribute has, and it will decrease over time. Navigate to an Event and click the Decay score button to see this.



this concludes the basics of misp

9 Steps to Generate API Key for Automation and Integration

1. Log in to your MISP instance as an administrator.
2. Navigate to **Administration** →**List Auth keys**.
3. Find the user for whom you want to generate an API key.
4. Click on the **Edit** button next to the user's name.
5. Scroll down to the API key section.
6. Click on **Generate new authentication key**.
7. Copy and securely store the generated API key.



10 Automation for Advanced Threat Detection and Intelligence Module

10.1 Fetch Threat Feeds from OSINT Sources

Purpose: Retrieve actionable intelligence such as malicious IPs, domains, and hashes from public and private threat intelligence platforms.

1. MISP API: Retrieve and process threat intelligence events.
2. AlienVault OTX API: Fetch reputation details for IPs/domains.
3. VirusTotal API: Check file hashes, URLs, or IPs for known threats.

11 Steps to Integrate AlienVault OTX with MISP

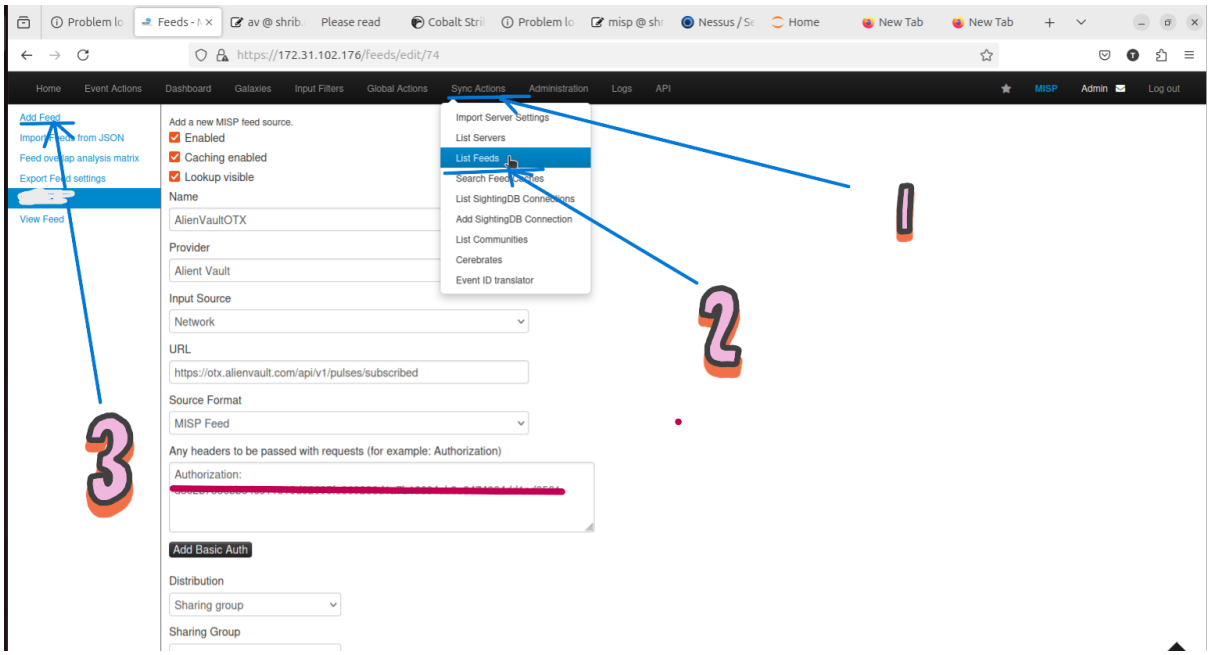
11.1 Prerequisites

- A running instance of MISP.
- An active AlienVault OTX account.
- API key from AlienVault OTX.

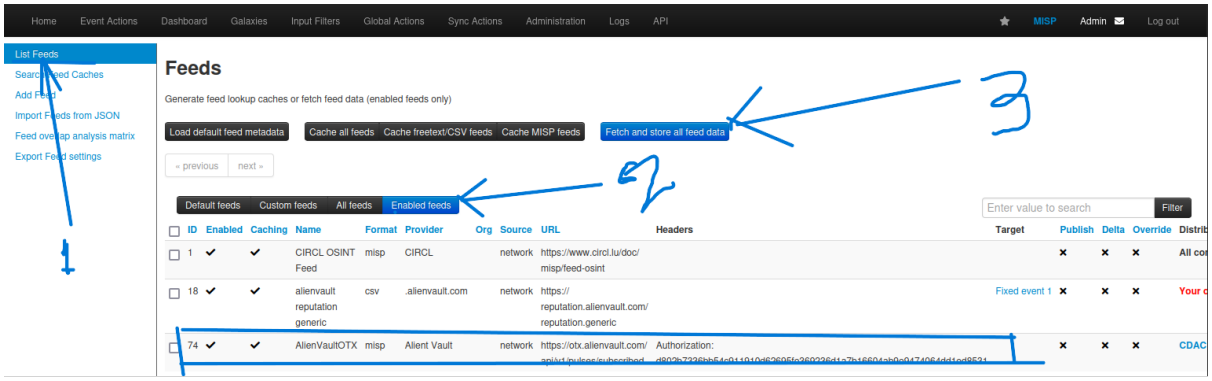
11.2 Configuring MISP to Fetch Data from OTX

1. Log in to your MISP instance as an administrator.
2. Navigate to **Sync Actions →List Feeds**.
3. Click on **Add Feed** and configure the following:
    - **Name:** AlienVault OTX
    - **URL:** Provide the appropriate OTX feed URL.
    - **Enabled:** Yes
    - **Source Format:** MISP Feed

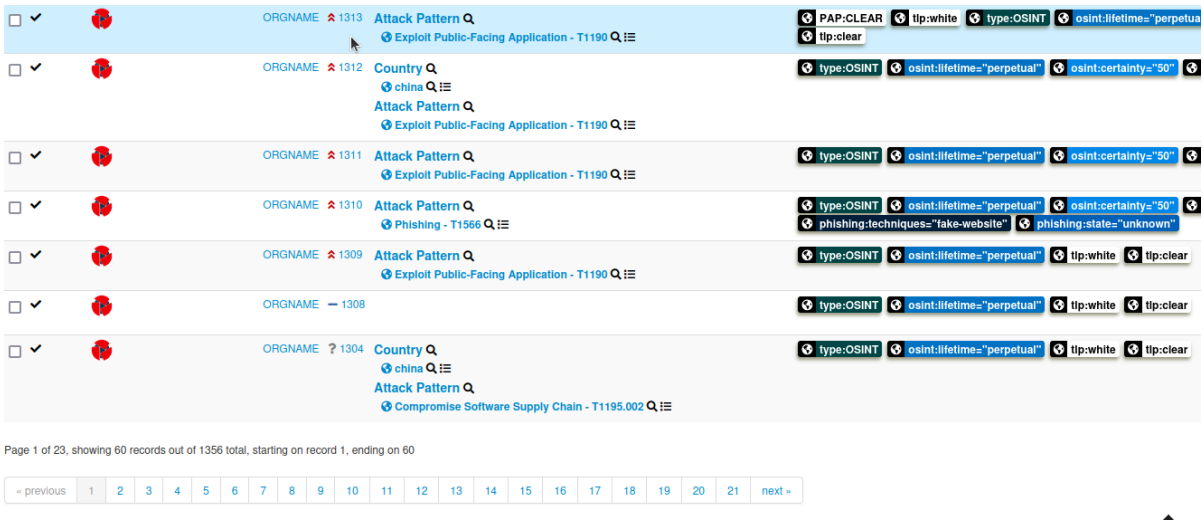
4. Click **Submit** to save the configuration.



5. Navigate to **Sync Actions** →**Fetch Feeds** and click **Fetch and Store** to pull data from OTX.



6. Verify that the fetched data is available in the **Events** section.



Now, to add virusTotal integration in misp , we need to install misp modules in our misp vm

Once the virtual environment is loaded, use the following command to install MISP modules:

12 Installing MISP Modules

MISP modules extend the functionality of MISP by providing additional enrichment and expansion features. The following steps outline the installation and configuration of MISP modules.

12.1 Step 1: Install MISP Modules

Once the virtual environment is loaded, use the following command to install MISP modules:

```
pip install misp-modules
```

12.2 Step 2: Install System Dependencies

On Ubuntu, install the required system dependencies:

```
sudo apt install libpoppler-cpp-dev libzbar0 tesseract-ocr
```

For an updated list of dependencies, check the GitHub workflow inside `.github/workflows`.

12.3 Step 3: Install Additional Python Packages

Some dependencies are not installed by default due to PyPI limitations. Install them manually using:

```
pip install \
    git+https://github.com/cartertemm/ODTReader.git \
    git+https://github.com/abenassi/Google-Search-API \
    git+https://github.com/SteveClement/trustar-python.git \
    git+https://github.com/sebdraven/pydnstrails.git \
    git+https://github.com/sebdraven/pyonyphe.git
```

12.4 Step 4: Run MISP Modules

After installation, you can start MISP modules with the following command:

misp−modules

If necessary, reload the virtual environment to update the search path for executables.

12.5 Step 4: Run MISP Modules

After installation, you can start MISP modules with the following command:

misp−modules

If necessary, reload the virtual environment to update the search path for executables.

13 Enriching Events with VirusTotal

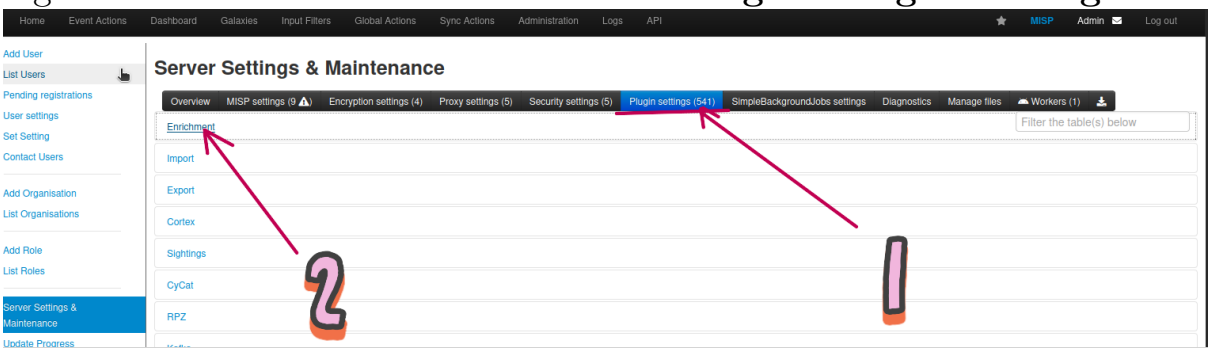
Once MISP modules are installed, you can use the VirusTotal enrichment module to gather additional threat intelligence. Follow these steps to configure and use it:

13.1 Step 1: Enable the VirusTotal Module

eginenumerate

Log in to your MISP instance as an administrator.

Navigate to **Administration →Server Settings →Plugin Settings**.



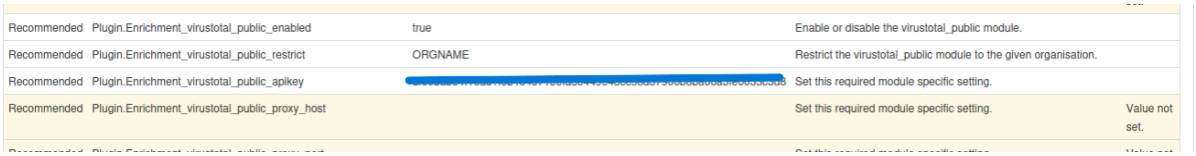
Find the section related to Enrichment and ensure that the VirusTotal module is enabled with the following configurations.

Save the settings.

13.2 Step 2: Configure API Key

VirusTotal requires an API key for querying its database. Obtain an API key from VirusTotal and configure it in MISP:

1. Go to `https://www.virustotal.com/gui/home/upload` and sign in.
2. Retrieve your API key from your account settings.
3. In MISP, navigate to **Administration →Plugin Settings**.
4. Locate the VirusTotal module settings and enter your API key.



5. Save the configuration.

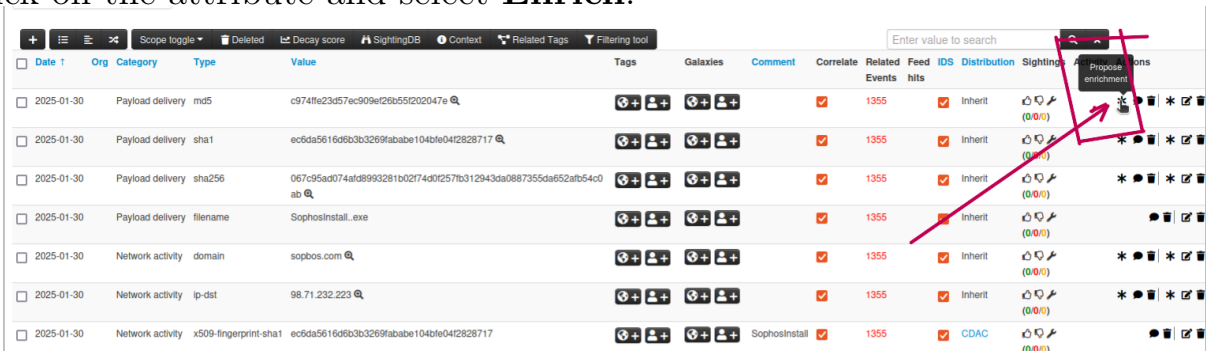
13.3 Step 3: Enrich an Event Using VirusTotal

eginenumerate

Navigate to an existing MISP event or create a new one.

Add an attribute such as a file hash, domain, or IP address.

Click on the attribute and select **Enrich**.

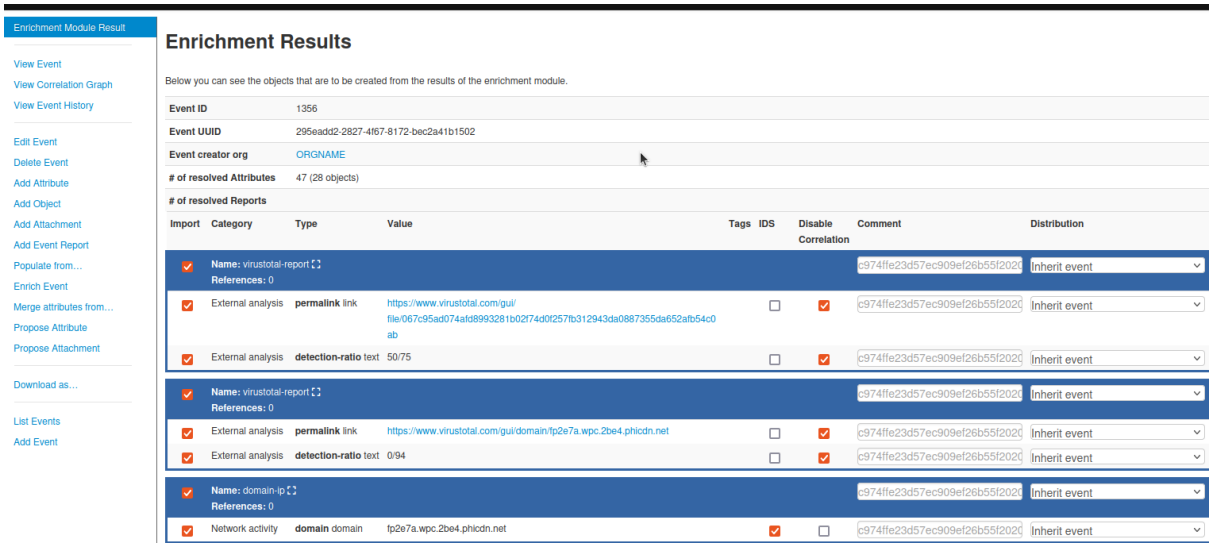


Choose **VirusTotal** from the list of enrichment modules.

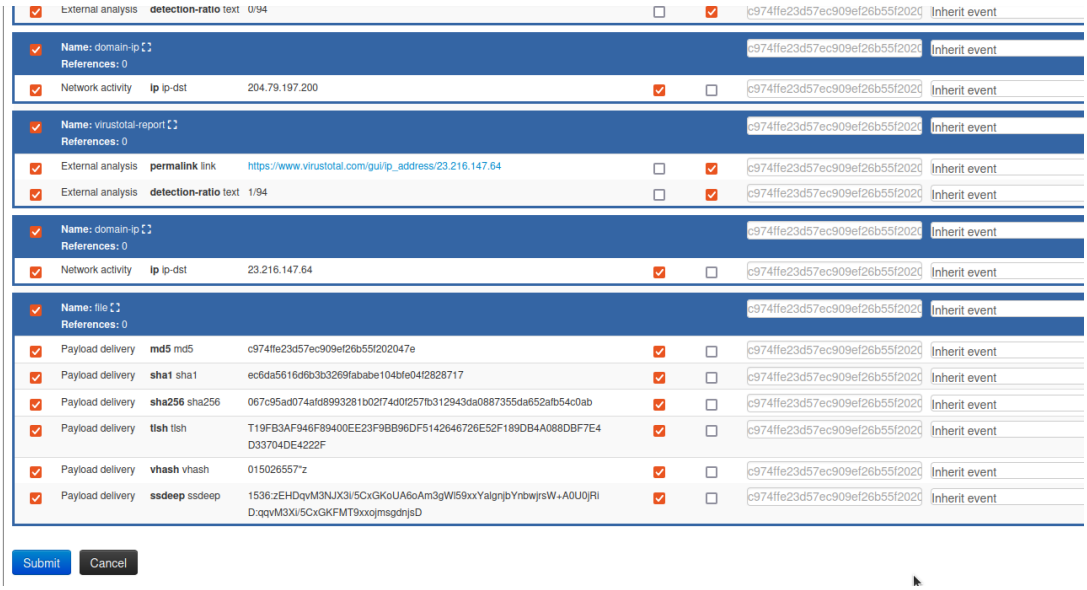
Wait for the enrichment process to complete and review the additional data fetched from VirusTotal.

13.4 Step 4: Review and Utilize the Enriched Data

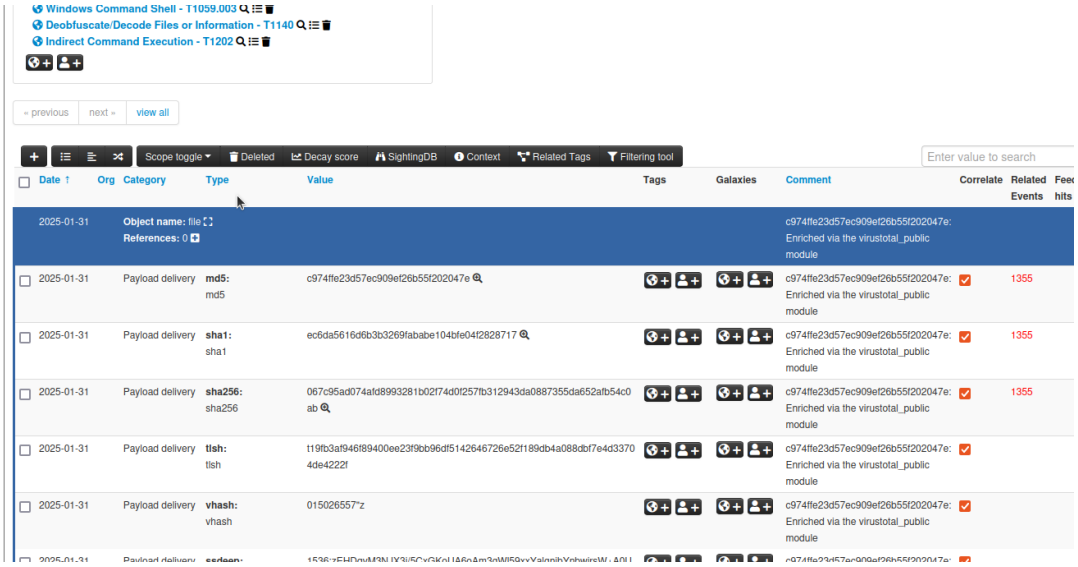
1. After enrichment, MISP will display additional threat intelligence from VirusTotal.



2. Review the information to assess threats and take appropriate action, scroll down and click submit.



3. Now, you can check the enriched data in attributes section of Events in Misp.



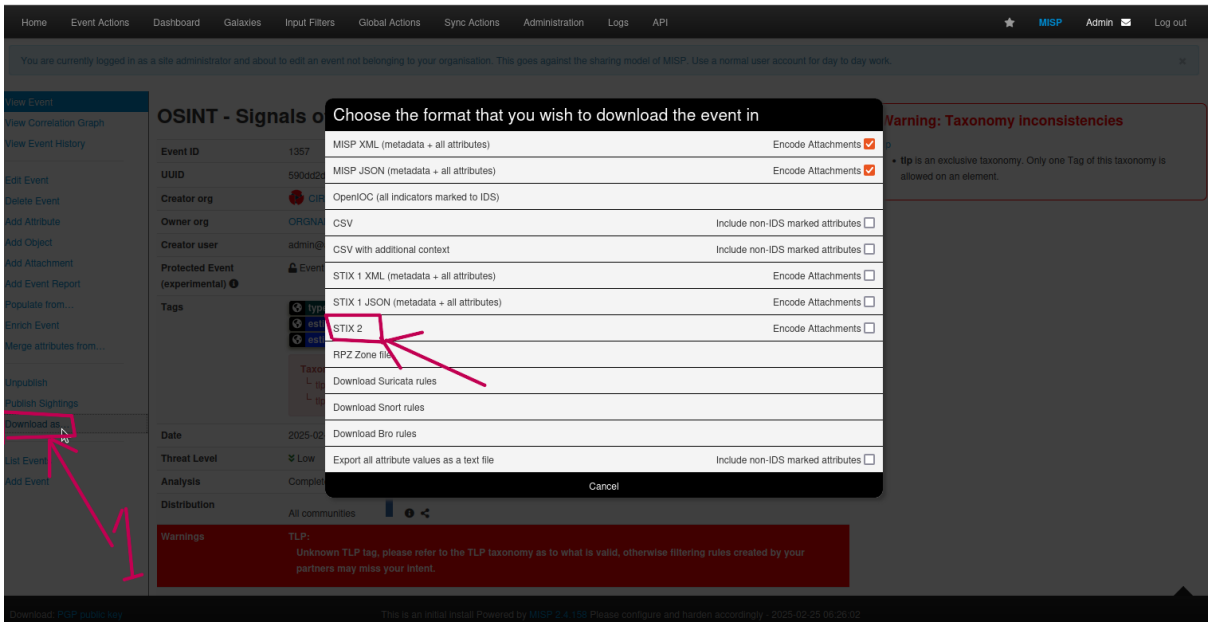
you can scroll down to check all the enriched data.

4. If needed, correlate the new data with other MISP events for deeper analysis.

14 Correlate Threat Feeds with Vulnerabilities

14.1 Prerequisites for performing Correlation

1. Misp Events should be available in stix-2 format for downloading



if you are unable to download then use the following commands in your misp VM to be to download the required MISP event stix-2 format

```
sudo apt update
pip install stix2
sudo apt install jq curl
```

2. once you are able to download the misp event in stix2 format now need to get data from NVD or you can also perform a nessus scan in your system

```
curl -s "https://services.nvd.nist.gov/rest/
  json/cves/2.0?pubStartDate=2024-01-01T00
  :00:00.000&pubEndDate=2024-01-30T23
  :59:59.999&resultsPerPage=500" | jq -c '.
  vulnerabilities[] | { index: { _index: "nvd-
  cves" } }, .' > nvd_cves.json
```

This fetches CVEs published between 01-01-2024 and 30-01-2024 and formats them for Elasticsearch bulk upload.

3. you can also perform nessus system scan to fetch all the necessary IOCS for correlation

4. once you get the file of CVEs/IOCS from nessus or NVD (using above command) then you need to format the NVD-CVE file using this script.

```
python3 - <<EOF
import json

# Load the original JSON file
input_file = "/home/musharaf-misp-admin/
  musharaf/misp scan_f20myj.nessus"
output_file = "/home/musharaf-misp-admin/
  nessus_cve_bulk.json"

with open(input_file, "r", encoding="utf-8") as
    file:
     data = json.load(file)

# Extract CVE data
cve_list = data.get("vulnerabilities", [])

# Convert to Elasticsearch bulk format
bulk_data = ""
for item in cve_list:
    cve = item.get("cve", {})
    cve_id = cve.get("id", "UNKNOWN")
    bulk_data += json.dumps({"index": {"_index"
      : "nvd_cve_index", "_id": cve_id}}) + "\n
      "
    bulk_data += json.dumps(cve) + "\n"

# Save to bulk file
with open(output_file, "w", encoding="utf-8")
  as bulk_file:
    bulk_file.write(bulk_data)

print(f"Bulk data saved to {output_file}")
EOF
```

```
Bulk data saved to /home/musharaf-misp-admin/
  nvd_cve_bulk.json
```

5. since you have two files now misp-event in stix2 format and NVD-cves file in desired format for correlation, next we will continue with ELK installation that is **elasticsearch logstash kibana** which is a SIEM tool.

15 ELK Stack for Threat Intelligence Correlation

The ELK Stack (Elasticsearch, Logstash, and Kibana) is widely used for real-time data analysis and visualization. In the context of threat intelligence, ELK enables efficient correlation of **Indicators of Compromise (IOCs) with vulnerabilities (CVEs)**.

- **Elasticsearch** serves as a distributed search engine to store and query threat intelligence data efficiently.
- **Logstash** helps process and ingest structured/unstructured security logs.
- **Kibana** provides an interactive UI to visualize the correlated data.

By using ELK, we can:

- Store threat intelligence data from MISP, VirusTotal, and Nessus in Elasticsearch.
- Correlate extracted CVEs with threat indicators (IPs, domains, hashes).
- Visualize trends, attack patterns, and vulnerability exposures using Kibana.

15.1 Installing ELK Stack

To set up the ELK stack, follow these steps:

15.1.1 Step 1: Install Elasticsearch

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-
  elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/8.
  x/apt stable main" | sudo tee -a /etc/apt/
  sources.list.d/elastic-8.x.list
sudo apt update
sudo apt install elasticsearch -y
```

Enable and start the Elasticsearch service:

```
sudo systemctl enable elasticsearch
sudo systemctl start elasticsearch
```

15.1.2 Step 2: Install Logstash

```
sudo apt install logstash -y
```

15.1.3 Step 3: Install Kibana

```
sudo apt install kibana -y
sudo systemctl enable kibana
sudo systemctl start kibana
```

Once installed, Kibana will be accessible at:

```
http://localhost:5601
```

once you are done with all the installation process, now we will send our nvd-cves file to elastic search.for that you may have to get the authorization credentials.

```
    curl -u elastic:"wlt8i+2TdnvX0dn-EBES" -X POST
       "https://localhost:9200/nvd_cve_index/_bulk
      " -H "Content-Type: application/json" --data
      -binary @/home/musharaf-misp-admin/
      nvd_cve_bulk.json -k
```

note : replace this ”wlt8i+2TdnvX0dn-EBES” with your actual password of elastic

upon successful completion you should see something like this


{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-29578","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1961,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-29570","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1962,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30368","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1963,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30369","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1964,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-24819","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1965,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-29479","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1966,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-29480","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1967,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-29849","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1968,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30370","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1969,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30371","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1970,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30372","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1971,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30373","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1972,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30375","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1973,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30376","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1974,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30378","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1975,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-26060","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1976,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-26061","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1977,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30544","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1978,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-30613","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1979,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-26059","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1980,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-26097","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1981,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-26865","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1982,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-27848","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1983,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-27849","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1984,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-29566","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1985,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-0276","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1986,"_primary_term":1,"status":201}},{"index":{"_index":"nvd_cve_index","_id":"CVE-2023-0277","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1987,"_primary_term":

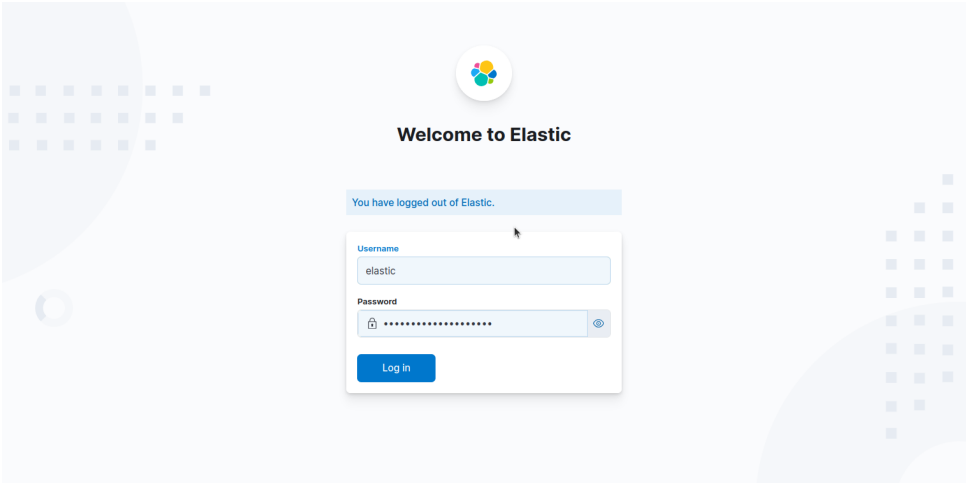
so with the help of above command we are able to send data into elastic search in required bulk format.

similarly follow the exact steps for misp cves file that is in stix2 format using the same command but only replacing the file name.

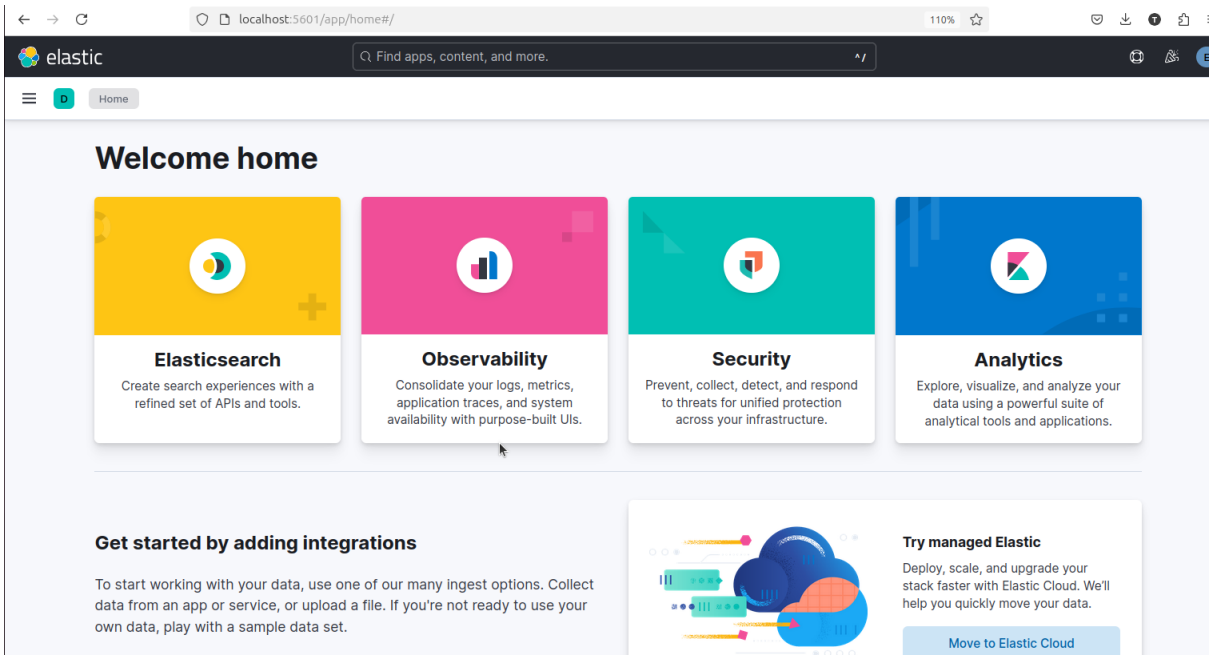
NOTE: you can also use log stash for sending files to elastic search since we are dealing with smaller files we will use curl in our terminal

15.2 how ELASTICSEARCH works for correlation

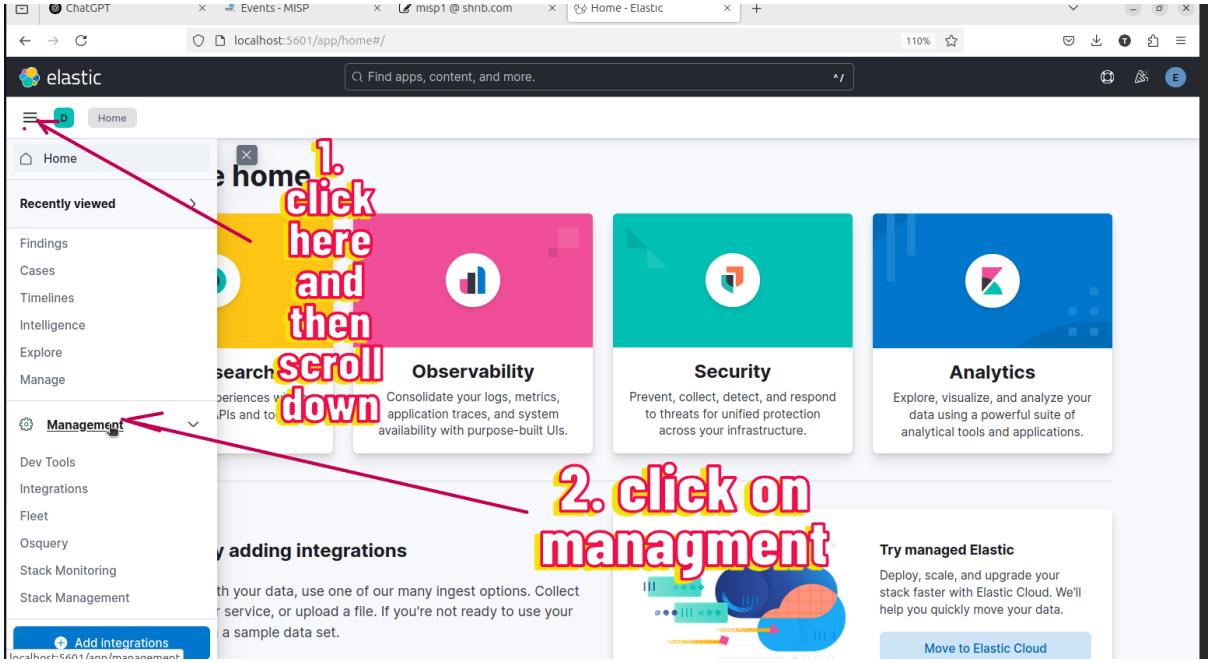
- step1 : once you Open Kibana (http://localhost:5601).(you have to sign up for your authorization credentials)



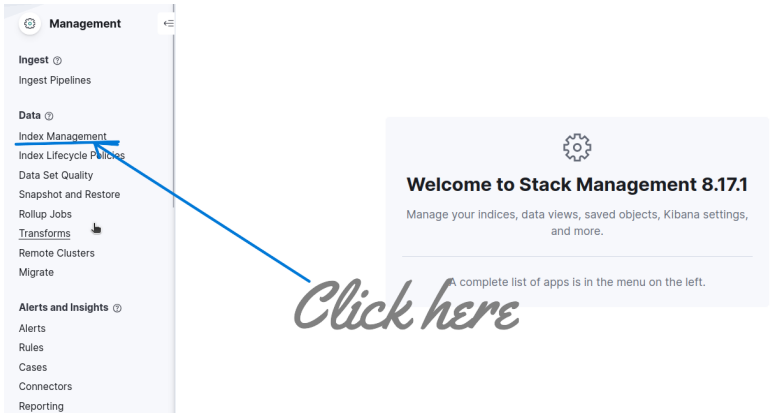
- once you sign up you should see something like this as your dashboard



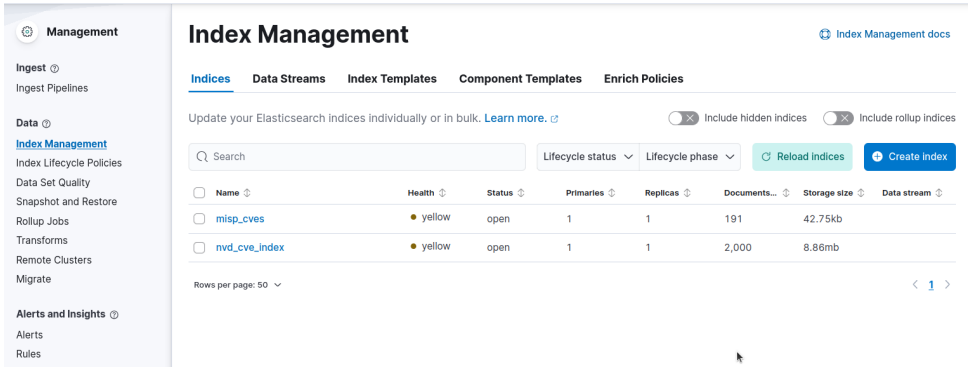
- next step



- next



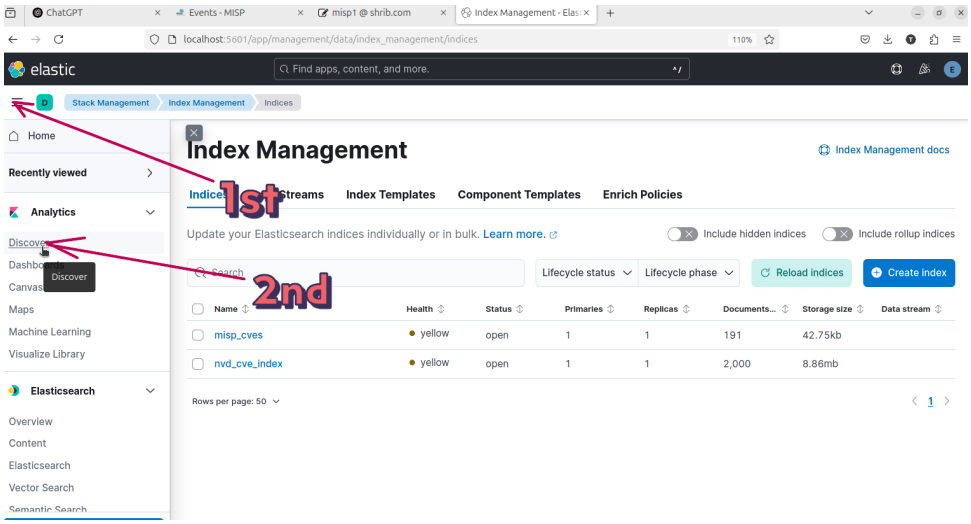
- next



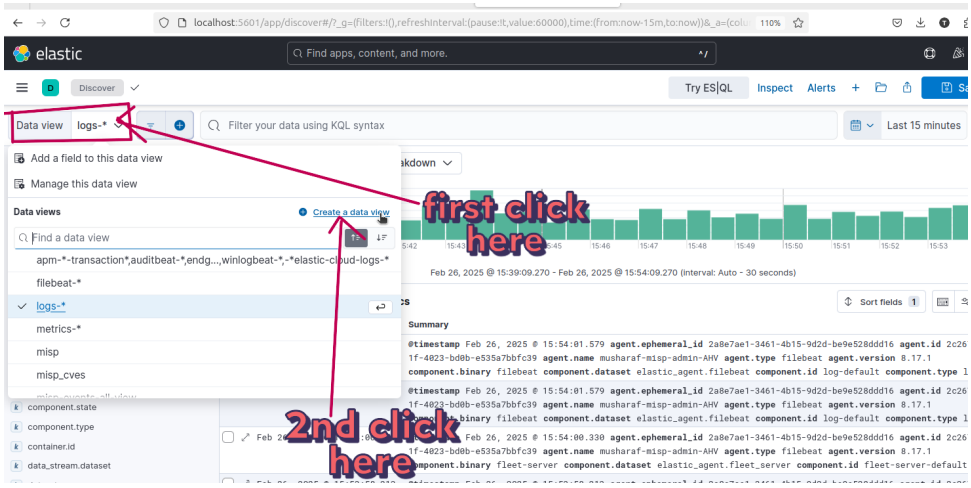
Here you can see our two files, one containing misp cves and another containing NVD cves

15.3 Creating data views

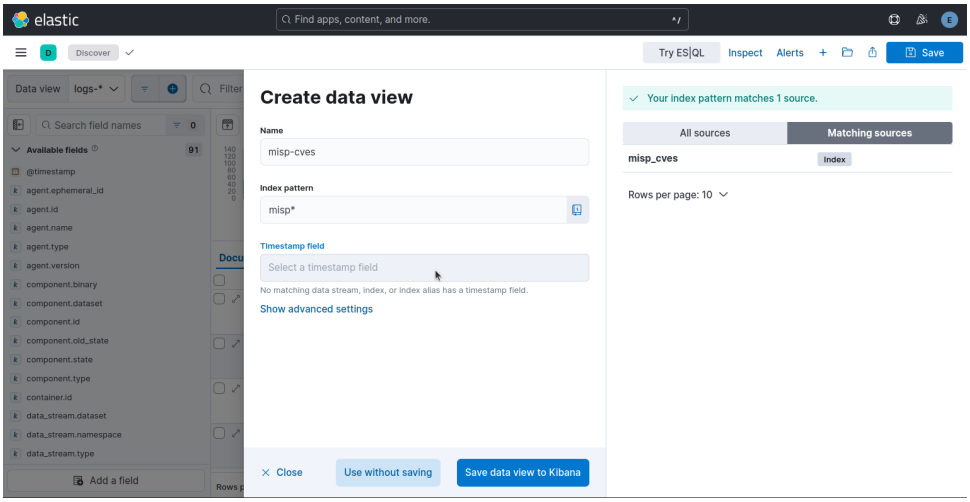
1. first



2. second



3. Now comes the important step



(a) **Open Data View Management**

- Click on the **"Create data view"** button.

(b) **Enter Data View Details**

- **Name**: Set a name for the data view, e.g., `misp-cves`.
- **Index Pattern**: Use `misp*` to match all indices that start with `misp`.
- Ensure the matching index appears on the right under **"Matching sources"**.

(c) **Select Timestamp Field (If Available)**

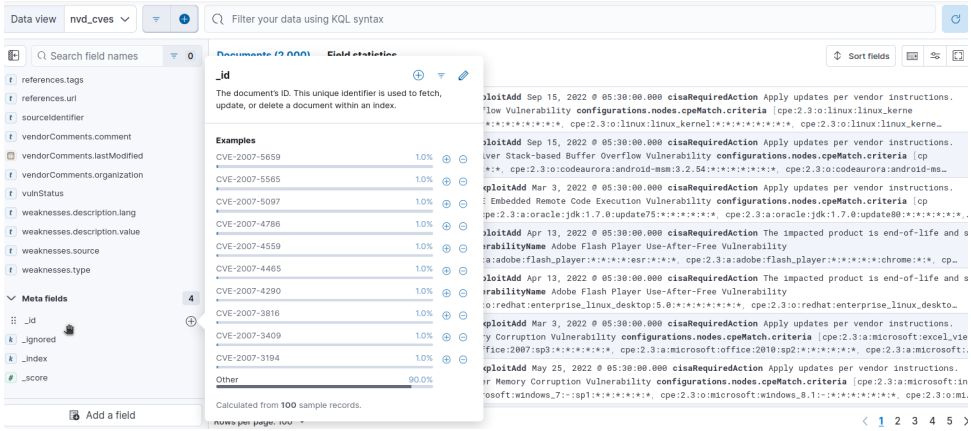
- If the index has a timestamp field, select it.
- If no timestamp field is found (as seen in Figure **??**), you can proceed without one.

(d) **Save the Data View**

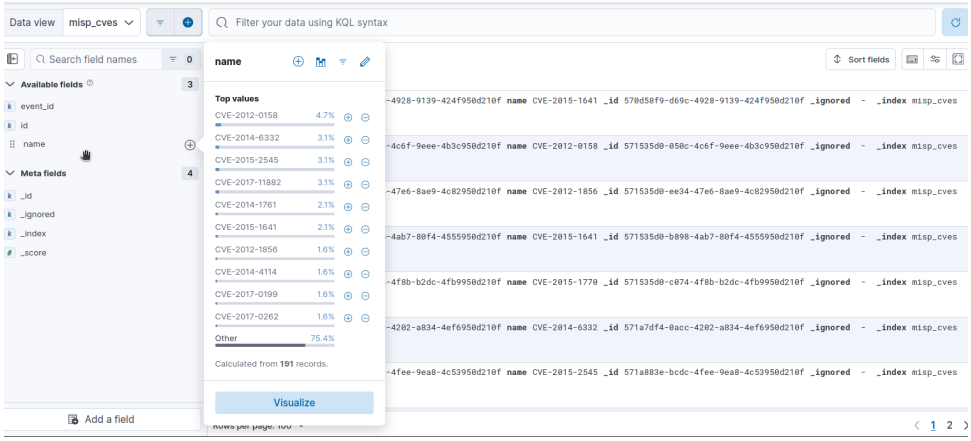
(e) **Repeat same steps for NVD-cves**

- Click **"Save data view to Kibana"** to finalize the setup.

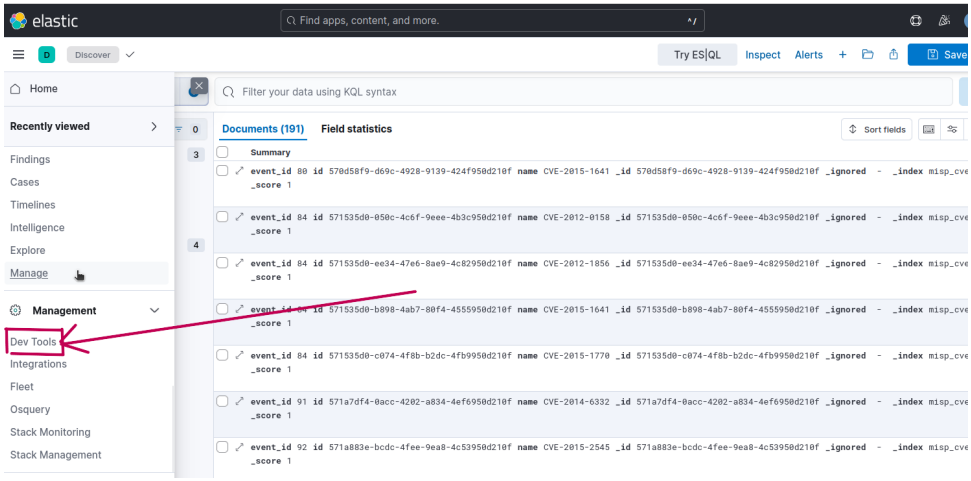
4. once you are done you will be able to find the cves in any of the folder below for nvd cves in data view they are in id folder like this.



5. and for misp cves, CVEs are in name folder



15.4 Performing correlation between cves of misp and nvd



Navigating to Dev Tools in Kibana: To access the console in Kibana, go to the **Management** section in the left sidebar and click on **Dev Tools**, as indicated by the arrow.

16 Correlation Queries in Kibana Console

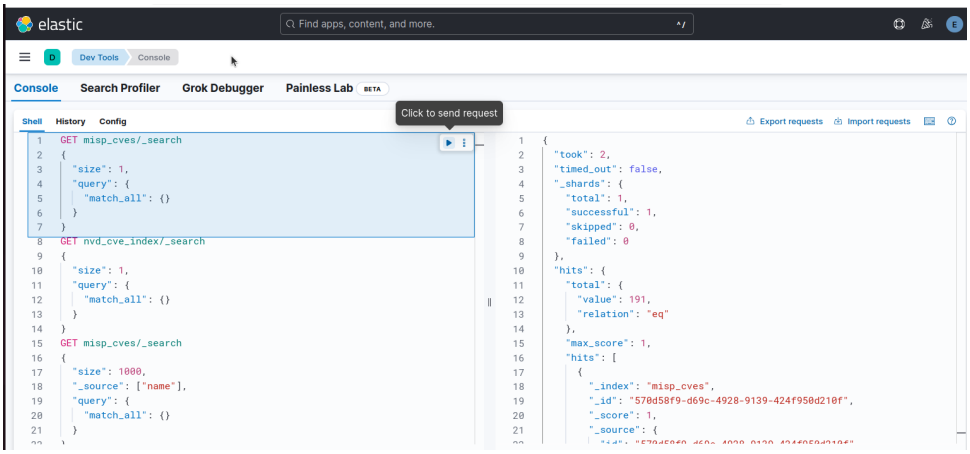
To perform correlation between threat intelligence data from MISP and vulnerability data from NVD, several Elasticsearch queries were executed using Kibana's Dev Tools. These queries help retrieve relevant data for analysis.

16.1 Retrieving Data from MISP and NVD Indices

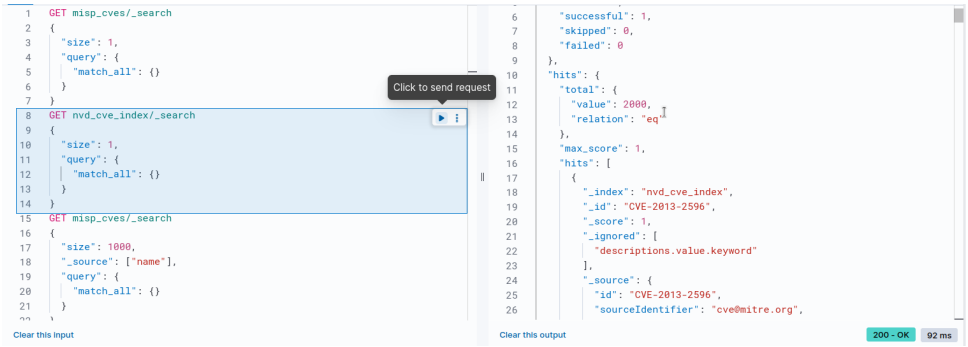
The following queries fetch data from the `misp_cves` and `nvd_cve_index` indices.

Listing 1: Retrieve sample data from MISP and NVD indices

```
GET misp_cves/_search
{
  "size": 10,
  "query": {
    "match_all": {}
  }
}
```



```
GET nvd_cve_index/_search
{
  "size": 1,
  "query": {
    "match_all": {}
  }
}
```



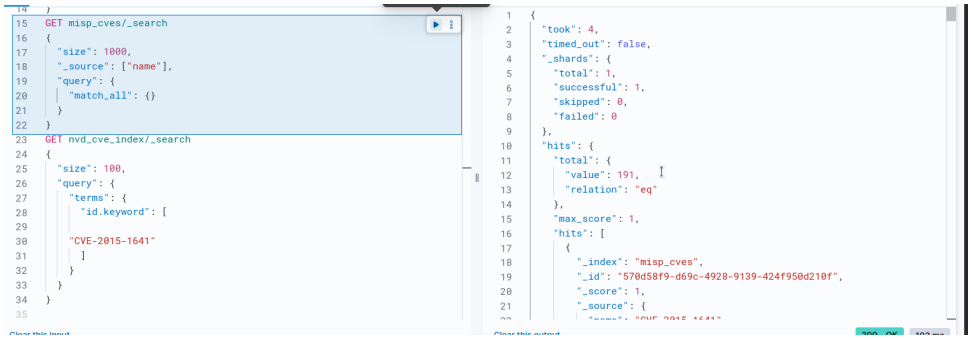
These queries retrieve a limited number of documents from both indices, ensuring that the data structure and fields are correctly understood before performing correlation.

16.2 Extracting CVE Names from MISP

To extract CVE names from the `misp_cves` index, the following query was executed:

Listing 2: Retrieve CVEs from MISP index

```
GET misp_cves/_search
{
  "size": 1000,
  "_source": ["name"],
  "query": {
    "match_all": {}
  }
}
```



This query retrieves up to 1000 CVE from the MISP index while excluding unnecessary fields.

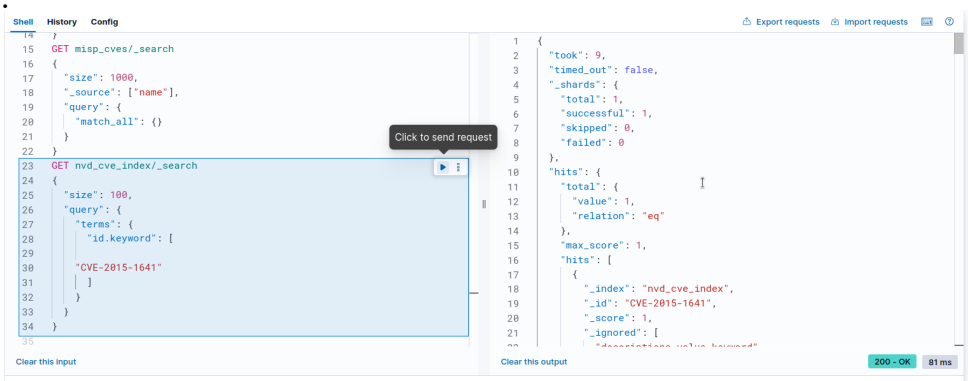
16.3 Filtering NVD Data for a Specific CVE

To check for the presence of a specific CVE (e.g., `CVE-2015-1641`) in the NVD dataset, the following query was used:

Listing 3: Search for a specific CVE in NVD

```
GET nvd_cve_index/_search
{
  "size": 100,
  "query": {
    "terms": {
      "id.keyword": [
        "CVE-2015-1641"
      ]
    }
  }
}
```

This query searches the NVD index for entries with the specified CVE ID.



Analysis of Response:

- `"took": 9` - The query execution time was 9 milliseconds.

- `"timed_out": false` - The query did not exceed the allowed time limit.
- `"_shards"` - The response shows that 1 shard processed the request successfully without failures.
- `"hits.total.value": 1` - This indicates that exactly one document matched the search criteria.
- `"hits.hits"` - Contains the retrieved document details.
- `"_index": "nvd_cve_index"` - The document belongs to the `nvd_cve_index` dataset.
- `"_id": "CVE-2015-1641"` - Confirms that the retrieved document corresponds to `CVE-2015-1641`.
- `"_score": 1` - Indicates the relevance score of the document to the search query.

16.3.1 Conclusion

The query successfully identified that `CVE-2015-1641` exists in the NVD index. This confirms that there is a known vulnerability associated with this identifier, and further details can be extracted for security analysis.

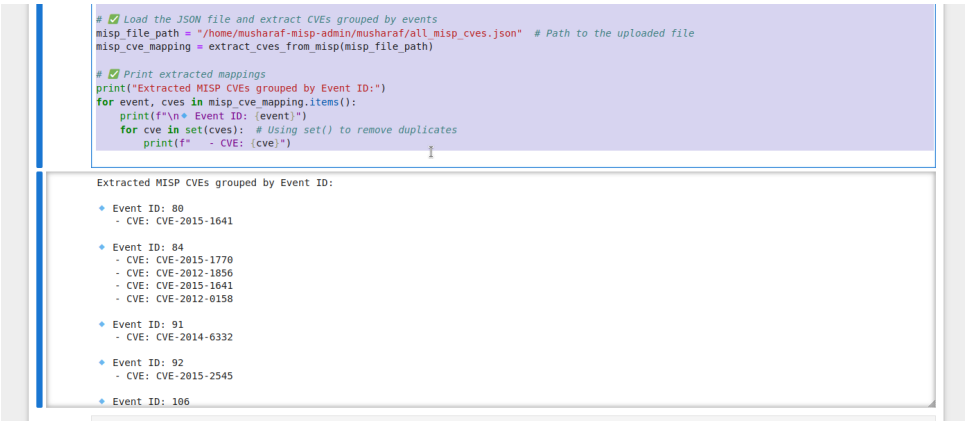
17 correlation without ELK

1. First we use python code in jupyter to fetch the neccassry CVEs for correlation

```
        import json

def extract_cves_from_misp(file_path):
    """Extracts CVEs and groups them by their
       event IDs from the MISP JSON file"""
    with open(file_path, "r") as f:
        misp_data = json.load(f)

    event_cve_mapping = {}

    for entry in misp_data:
        event_id = entry.get("event_id", "
          Unknown Event")
        cve_id = entry.get("name", "Unknown CVE
          ")

        if event_id not in event_cve_mapping:
            event_cve_mapping[event_id] = []

        event_cve_mapping[event_id].append(
          cve_id)

    return event_cve_mapping

#     Load the JSON file and extract CVEs
  grouped by events
misp_file_path = "/home/musharaf-misp-admin/
  musharaf/all_misp_cves.json"  # Path to the
  uploaded file
misp_cve_mapping = extract_cves_from_misp(
  misp_file_path)

#     Print extracted mappings
print("Extracted MISP CVEs grouped by Event ID:
  ")
```

```
for event, cves in misp_cve_mapping.items():
    print(f"\n Event ID: {event}")
    for cve in set(cves):  # Using set() to
      remove duplicates
        print(f"   - CVE: {cve}")
```

note: replace the file path your actual file path



2. now I will use nessus scan of my system her for correlation, here will fetch cves from nessus scan file

```
          import xml.etree.ElementTree as ET

def extract_cves_cpes_from_nessus(file_path):
    """Extracts CVEs and CPEs from a Nessus
      scan report and groups by host"""
    tree = ET.parse(file_path)
    root = tree.getroot()

    nessus_cve_cpe = {}

    for report_host in root.findall(".//
      ReportHost"):
        host_name = report_host.get("name", "
          Unknown Host")

        for report_item in report_host.findall(
          ".//ReportItem"):
            # Extract CVEs
            cve_list = report_item.find("cve")
            cve_ids = cve_list.text.split(",")
```

```
                if cve_list is not None else []

            # Extract CPE
            cpe_element = report_item.find("cpe
                ")
            cpe = cpe_element.text if
                cpe_element is not None else "
                Unknown CPE"

            for cve_id in cve_ids:
                if host_name not in
                    nessus_cve_cpe:
                    nessus_cve_cpe[host_name] =
                        {}
                if cve_id not in nessus_cve_cpe
                    [host_name]:
                    nessus_cve_cpe[host_name][
                        cve_id] = set()

                if cpe != "Unknown CPE":
                    nessus_cve_cpe[host_name][
                        cve_id].add(cpe)

    return nessus_cve_cpe

#     Load Nessus file and extract CVEs & CPEs
    grouped by host
nessus_file_path = "/home/musharaf-misp-admin/
    musharaf/misp scan_f20myj.nessus"  # Update
    path accordingly
nessus_cve_cpe_mapping =
    extract_cves_cpes_from_nessus(
    nessus_file_path)

#     Print extracted Nessus CVEs and CPEs
    grouped by hosts
print("\n Extracted Nessus CVEs and CPEs:")
if nessus_cve_cpe_mapping:
    for host, cve_data in
```

```
        nessus_cve_cpe_mapping.items():
            print(f"\n Host: {host}")
            for cve, cpes in cve_data.items():
                print(f"    {cve}")
                for cpe in cpes:
                    print(f"       - CPE: {cpe}")
else:
    print("    No CVEs found in the Nessus scan
        report!")
```

Now once we have got CVEs from both files we will perform correlation

```
    def correlate_misp_nessus(misp_data,
        nessus_data):
"""Correlates CVEs from MISP (Threat Intel)
   and Nessus (Vulnerability Scan)"""
correlation_results = {}
common_cves = set()

# Extracting CVEs from MISP (Flattening event-
   wise mapping)
misp_cve_list = set()
for event_cves in misp_data.values():
    misp_cve_list.update(event_cves)

# Extracting CVEs from Nessus (Flattening host
   -wise mapping)
nessus_cve_list = set()
nessus_cpe_map = {}  # To store CPEs per CVE
```

```
    for host, cve_data in nessus_data.items():
        for cve, cpes in cve_data.items():
            nessus_cve_list.add(cve)
            if cve not in nessus_cpe_map:
                nessus_cpe_map[cve] = set()
            nessus_cpe_map[cve].update(cpes)

    # Find common CVEs
    common_cves = misp_cve_list & nessus_cve_list

    if common_cves:
        for cve in common_cves:
            correlation_results[cve] = {
                "misp_events": [event for event,
                  cves in misp_data.items() if cve
                   in cves],
                "nessus_hosts": [host for host,
                  cve_data in nessus_data.items()
                  if cve in cve_data],
                "nessus_cpes": list(nessus_cpe_map
                  .get(cve, set()))  # Get
                  correlated CPEs
            }

    return correlation_results, misp_cve_list,
      nessus_cve_list, common_cves

#     Perform correlation
correlation_results, misp_cves, nessus_cves,
  common_cves = correlate_misp_nessus(
  misp_cve_mapping, nessus_cve_cpe_mapping)

#     Print correlation results
print("\n Correlation Between MISP Threat Intel &
  Nessus Scan:")

if common_cves:
    for cve, data in correlation_results.items():
```

```
        print(f"\n CVE: {cve}")
        print("    MISP Events Containing This CVE
          :", data["misp_events"])
        print("    Nessus Hosts Affected:", data["
          nessus_hosts"])
        print("    CPEs Found in Nessus:", data["
          nessus_cpes"] if data["nessus_cpes"]
          else "None")
else:
    print("\ n   No Matching CVEs Found Between
      MISP and Nessus!")
    print("\n MISP CVEs:", sorted(misp_cves))
    print("\n Nessus CVEs:", sorted(nessus_cves))
```

18 Detailed Project Repository without using elk

If you find hard to follow these python codes here you can use my github repository for the same: MISP and Nessus Scan Correlation (GitHub)