Report on MISP

CSE 406: Computer Security Sessional

Prepared by:

1805062 - Ayesha Binte Mostofa
1805069 - Sumonta Nandy Amit

Supervised by:

Abdur Rashid Tushar
Lecturer, CSE, BUET

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

Contents

1 Introduction

MISP (Malware Information Sharing Platform & Threat Sharing) is an open-source platform designed for sharing, analyzing, and collaborating on structured threat information within the cybersecurity community. It offers several key features:

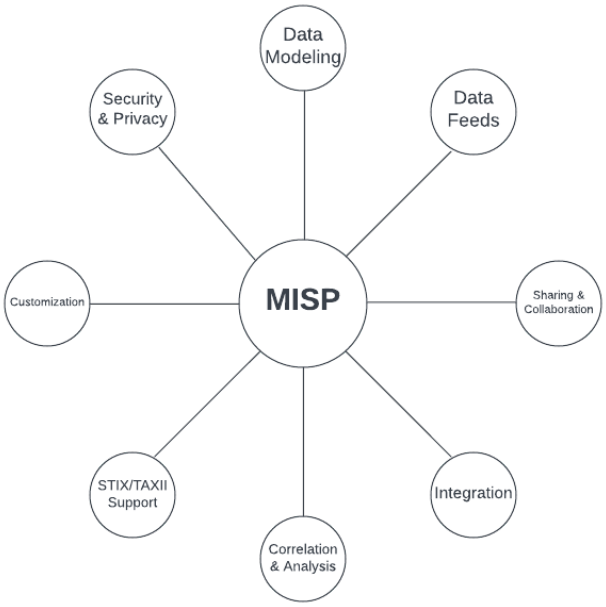


Figure 1: Key Features of MISP

1.1 Data Sharing

MISP facilitates the sharing of cybersecurity threat data among organizations, enhancing the collective ability to respond to evolving threats.

1.2 Data Normalization

The platform enforces a common structure for describing and normalizing threat information, ensuring consistency in data representation and analysis.

1.3 Threat Intelligence Feeds

Users can integrate external threat intelligence feeds, enriching their data with up-to-date information from reputable sources.

1.4 Event Correlation

MISP allows the correlation of related events, providing a contextual understanding of threats and their potential impact.

1.5 Flexible Taxonomies

The platform supports various taxonomies and classification systems, enabling organizations to categorize and label threat information as needed.

1.6 Collaboration

MISP encourages collaboration between organizations, facilitating the sharing of knowledge and expertise to collectively defend against cyber threats.

1.7 Indicators of Compromise (IOCs)

Users can share IOCs like IP addresses, domain names, hashes, and other artifacts to identify and mitigate threats.

1.8 Incident Response

MISP assists in incident response by centralizing information and providing a structured way to analyze and share data related to ongoing security incidents.

1.9 API and Automation

The platform provides an API for task automation, simplifying integration into existing security workflows and tools.

1.10 Customization

Organizations can customize MISP to match their needs by adapting taxonomies, attributes, and configurations.

1.11 Privacy and Access Control

MISP offers granular data access control, allowing organizations to define who can access, share, and contribute information.

2 Source Code Overview

There are a total of 82 repositories under the MISP project. The most popular ones are discussed here. Most of the programs are written in Python.

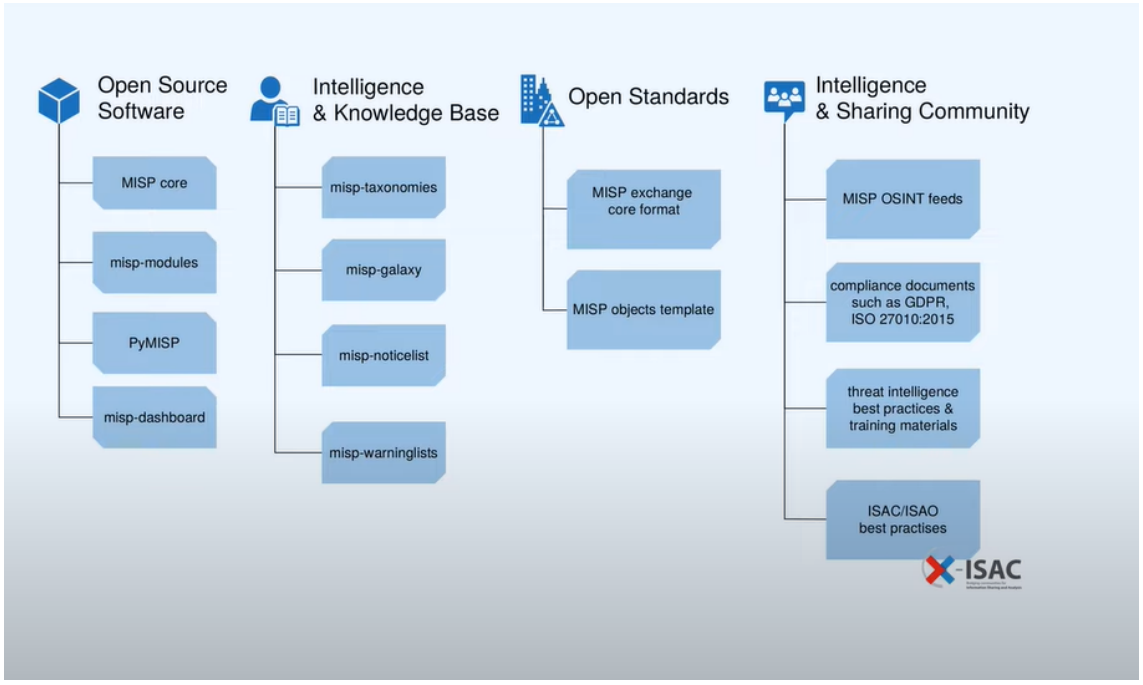


Figure 2: MISP Project

2.1 MISP core

This is the main module of the MISP project. starts with app/index.php

```
<?PHP
...
require 'webroot' . DIRECTORY_SEPARATOR . 'index.php';
```

app/Model/Event.php

```
<?php
App::uses('AppModel', 'Model');
App::uses('CakeEmail', 'Network/Email');
App::uses('AttachmentTool', 'Tools');
App::uses('TmpFileTool', 'Tools');
App::uses('SendEmailTemplate', 'Tools');
App::uses('ProcessTool', 'Tools');
...
class Event extends AppModel
{
```

```
    ...
    private function __attachAttributeTags(array &$events, $excludeLocalTags = false)
    ...
      private function __attachTags(array &$event, $justExportable)
      ...
    public function restSearch(array $user, $returnFormat, $filters, $paramsOnly = false, $jobId = false, &$elementCounter = 0, &$renderView = false)
    {
    ...
     public function clusterEventIds($exportTool, $eventIds)
     ...
    public function add_original_file($file, $original_filename, $event_id, $format)
      ...
    private function getRequiredTaxonomies()
    {
        $this->Taxonomy = ClassRegistry::init('Taxonomy');
        return $this->Taxonomy->find('column', array(
            'conditions' => array('Taxonomy.required' => 1, 'Taxonomy.enabled' => 1),
            'fields' => array('Taxonomy.namespace')
        ));
    }
    ...
    public function extractAllTagNames(array $event)
    {
        $tags = array();
        if (!empty($event['EventTag'])) {
            foreach ($event['EventTag'] as $eventTag) {
                $tagName = $eventTag['Tag']['name'];
                $tags[$tagName] = $tagName;
            }
        }
    ...
    public function getExtendingEventIdsFromEvent($user, $eventID)
    ...
    public function getEventRepublishBanStatus($eventID)
    {
        $banStatus = [
            'error' => false,
            'active' => false,
            'message' => __('Event publish is not banned')
        ];
    ...
    public function exportTypes()
    {
        return array(
            'json' => array(
                'extension' => '.json',
                'type' => 'JSON',
                'scope' => 'Event',
                'requiresPublished' => 0,
                'params' => array('includeAttachments' => 1, 'ignore' => 1, 'returnFormat' => 'json'),
                'description' => __('Click this to download all
```

```
    events and attributes that you have access to in MISP JSON
    format.'),
            ),
    ...
    public function publishEventToZmq($id, $user, &$fullEvent)
    ...
    public function publishEventToKafka($id, $user, &$fullEvent,
    $kafkaTopic)
    ...
    public function getTrendsForTags(array $user, array
    $eventFilters=[], int $baseDayRange, int $rollingWindows=3,
    $tagFilterPrefixes=null): array
    ...
    public function getTrendsForTagsFromEvents(array $events, int
    $baseDayRange, int $rollingWindows=3, $tagFilterPrefixes=null):
     array
    ...
    public function extractRelatedCourseOfActions(array $events):
    array
    ...
```

app/Model/Attribute.php

```
    class Attribute extends AppModel
    {
    ...
     const EDITABLE_FIELDS = [
        'timestamp',
        'category',
        'value',
        'value1',
    ...
```

app/Model/EventGraph.php

```
    <?php
    App::uses('AppModel', 'Model');

    class EventGraph extends AppModel
    {
    ...
    public function getPictureData($eventGraph)
        {
    ...
```

app/Model/EventReport.php

```
    <?php
    App::uses('AppModel', 'Model');
    ...
    class EventReport extends AppModel
    {
    ...
     public function captureReport(array $user, array $report,
    $eventId)
    {
    ...
     public function editReport(array $user, array $report,
    $eventId, $fromPull = false, &$nothingToChange = false)
```

```
    {
    ...
    public function deleteReport(array $user, $report, $hard=false)
    {
    ...
```

app/Model/EventBlocklist.php

```
    <?php
    App::uses('AppModel', 'Model');

    class EventBlocklist extends AppModel
    {
    ...
     public function isBlocked($eventUuid)
    {
        ...
     public function removeBlockedEvents(array &$eventArray)
    {
    ...
```

app/Model/Feed.php

```
    <?php
    App::uses('AppModel', 'Model');
    App::uses('RandomTool', 'Tools');
    App::uses('TmpFileTool', 'Tools');
    App::uses('AttributeValidationTool', 'Tools');

    class Feed extends AppModel
    {
    ...
     private function getCachedFeedsOrServers(array $user, $scope)
    {
    ...
     private function downloadFromFeed(array $actions, array $feed,
     HttpSocket $HttpSocket = null, array $user, $jobId = false)
    {
    ...
     private function __createFeedRequest($headers = false)
    {
    ...
```

app/Model/Galaxy.php

```
    <?php
    App::uses('AppModel', 'Model');

    /**
     * @property GalaxyCluster $GalaxyCluster
     * @property Galaxy $Galaxy
     */
    class Galaxy extends AppModel
    ...
    private function __load_galaxies($force = false)
    {
    ...
    private function __getPreExistingClusters(array $galaxies,
    array $cluster_package)
```

```
    {
    ...
     private function __createClusters($cluster_package, $template)
    {
    ...
    public function captureGalaxy(array $user, array $galaxy)
    {
    ...
     public function importGalaxyAndClusters(array $user, array
    $clusters)
    {
    ...
```

app/Model/Job.php

```
    <?php
    App::uses('AppModel', 'Model');

    class Job extends AppModel
    {
    ...
     public function createJob($user, $worker, $jobType, $jobInput,
     $message = '')
    {
    ...
    public function saveProgress($jobId = null, $message = null,
    $progress = null)
    {
    ...
     public function saveStatus($jobId = null, $success = true,
    $message = null)
    {
    ...
```

app/Model/MispObject.php

```
    <?php
    App::uses('AppModel', 'Model');
    App::uses('TmpFileTool', 'Tools');
    App::uses('AttributeValidationTool', 'Tools');
    App::uses('FileAccessTool', 'Tools');

    /**
     * @property Event $Event
     * @property SharingGroup $SharingGroup
     * @property Attribute $Attribute
     * @property ObjectReference $ObjectReference
     * @property ObjectTemplate $ObjectTemplate
     */
    class MispObject extends AppModel
    {
    ...
    $simple_params = array(
                'Object' => array(
                    'object_name' => array('function' => '
    set_filter_object_name'),
                    'object_template_uuid' => array('function' => '
    set_filter_object_template_uuid'),
```

```
                    'object_template_version' => array('function' => 'set_filter_object_template_version'),
                    'deleted' => array('function' => 'set_filter_deleted')
                ),
                ...

 private function checkForDuplicateObjects($object, $eventId, &$duplicatedObjectId, &$duplicateObjectUuid)
{
...
 public function saveObject(array $object, $eventId, $template = false, array $user, $errorBehaviour = 'drop', $breakOnDuplicate = false)
{
...
 public function deltaMerge(array $object, array $objectToSave, $onlyAddNewAttribute=false, array $user)
{
...
```

app/Model/Tag.php

```
<?php
App::uses('AppModel', 'Model');

/**
 * @property EventTag $EventTag
 * @property User $User
 * @property AttributeTag $AttributeTag
 * @property FavouriteTag $FavouriteTag
 * @property Organisation $Organisation
 */
class Tag extends AppModel
{
...
 public function lookupTagIdForUser(array $user, $tagName)
{
...
 public function fetchUsableTags(array $user, $isGalaxy = null)
{
...
```

app/Model/Taxonomy.php

```
<?php
App::uses('AppModel', 'Model');

/**
 * @property TaxonomyPredicate $TaxonomyPredicate
 */
class Taxonomy extends AppModel
{
...
private function __getTaxonomy($id, $filter = false)
{
...
 public function getAllTaxonomyTags($inverse = false, $user = false, $full = false, $hideUnselectable = true, $local_tag =
```

```
false)
{
...
public function getTaxonomyTags($id, $upperCase = false, $existingOnly = false)
{
...
 public function addTags($id, $tagList = false)
{
...
 public function checkIfNewTagIsAllowedByTaxonomy($newTagName, array $tagNameList=array())
{
...
```

2.2 PyMISP

Repository Link: https://github.com/MISP/PyMISP/ PyMISP is a Python library to access MISP platforms via their REST API. PyMISP allows you to fetch events, add or update events/attributes, add or update samples, or search for attributes.
The functionalities were used when we integrated the MISP instance with the Cortex in our demo.

Dependencies:

- Python 3.10
- fileobjects: to create PE/ELF/Mach-o objects
- openioc: to import files in OpenIOC format (not really maintained)
- virustotal: to query VirusTotal and generate the appropriate objects
- docs: to generate te documentation
- pdfexport: to generate PDF reports out of MISP events
- url: to generate URL objects out of URLs with Pyfaup
- email: to generate MISP Email objects
- brotli: to use the brotli compression when interacting with a MISP instance

PyMISP classes:

- PyMISP: This class is responsible for processing the URL of the MISP instance you want to connect to and The API key of the user you want to use
- MISPAbstract

- MISPEncode
- MISPEvent
- MISPEventBlocklist
- MISPEventDelegation
- MISPAttribute
- MISPObject
- MISPObjectAttribute
- MISPObjectReference
- MISPObjectTemplate
- MISPTag
- MISPUser
- MISPUserSetting
- MISPOrganisation
- MISPOrganisationBlocklist
- MISPFeed
- MISPInbox
- MISPLog
- MISPNoticelist
- MISPRole
- MISPServer
- MISPShadowAttribute
- MISPSharingGroup
- MISPSighting
- MISPTaxonomy
- MISPWarninglist

PyMISP Tools

- File Object
- ELF Object

- PE Object
- Mach-O Object
- VT Report Object
- STIX
- OpenIOC

How to use PyMISP: We can find scripts and examples in the example directory

```
cd examples
cp keys.py.sample keys.py
vim keys.py
```

2.3 misp-taxonomies

Repository Link: https://github.com/MISP/misp-taxonomies MISP Taxonomies is a set of common classification libraries to tag, classify, and organize information. Taxonomy allows to express same vocabulary among a distributed set of users and organizations.

Available taxonomies: There are a lot of built-in taxonomies available in this directory. e.g.: CERT-XLM, DFRLab-dichotomies-of-disinformation, Gray-Zone etc.

Taxonomy structure A JSON file describing taxonomy as triple tags inside a directory-matching namespace.

How to add private taxonomy?

```
cd /var/www/MISP/app/files/taxonomies/
mkdir privatetaxonomy
cd privatetaxonomy
vi machinetag.json
```

2.4 misp-galaxy

Repository Link: https://github.com/MISP/misp-galaxy MISP galaxy is a simple method to express a large object called a cluster that can be attached to MISP events or attributes. A cluster can be composed of one or more elements. Elements are expressed as key values. There are default knowledge bases (such as Threat Actors, Tools, Ransomware, ATT & CK matrixes) available in MISP galaxy but those can be overwritten, replaced, updated, forked, and shared. Some available galaxies are 360.net Threat Actors, Android, Azure Threat Research Matrix, etc.

Testing the galaxies:

```
    sudo apt install jq moreutils python3-jsonschema
    sudo wget -O /usr/local/bin/jsonschema https://gist.
    githubusercontent.com/SteveClement/
    e6ac60e153e9657913000216fc77c6ef/raw/
    c273ace06ad338d609dd2c84a0a6e215a268ea11/jsonschema
    sudo chmod +x /usr/local/bin/jsonschema # This will only work
    with jsonschema >2.4 (before no CLI interface was available)
```

2.5 misp-warninglists

Repository link: https://github.com/MISP/misp-warninglists
misp-warninglists are lists of well-known indicators that can be associated with potential false positives, errors, or mistakes.
warning list format:

```
    {
  "name": "List of known public DNS resolvers",
  "version": 1,
  "description": "Event contains one or more public DNS resolvers
    as attribute with an IDS flag set",
  "matching_attributes": [
    "ip-src",
    "ip-dst"
  ],
  "list": [
    "8.8.8.8",
    "8.8.4.4",
    "208.67.222.222",
    "208.67.220.220",
    "195.46.39.39",
    "195.46.39.40"
  ]
}
```

types of warning list

- string
- substring
- hostname
- cidr

3 Installation Guide

3.1 Resources

- Installation in VM and Ubuntu: https://www.youtube.com/watch?v=nZcTc60YsIs
  Documentation: https://misp.github.io/MISP/INSTALL.ubuntu2004
- Installation using Docker: https://youtu.be/h_lxGcvjg8U

3.2 Possible errors

- **ERROR**: cannot verify raw.githubusercontent.com's certificate, issued by 'CN=DigiCert TLS RSA SHA256 2020 CA1,O=DigiCert Inc,C=US':

  Sol ^{n}:

  - create a *~/.wgetrc* file
  - add *ca_certificate=/etc/ssl/certs/ca-certificates.crt* in it

3.3 Launching MISP in ubuntu and vm

- open shell
- find 'inet' for enp0s3 value using 'ifconfig'
- open browser and put the 'inet' value in the address bar
- give the credentials

```
            Email: admin@admin.test
            Password: admin

```

- change the password

3.4 Launching MISP in Docker

- Change Localhost:8080 in docker-compose.yml file.
- Login using admin@admin.test email and pass
- change the password

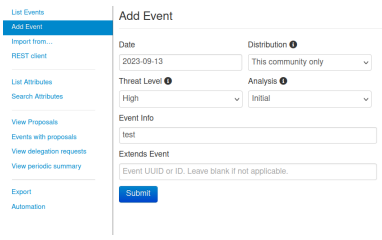
3.5 Forget password issue resolve

- Use this command line : sudo /var/www/MISP/app/Console/cake Password email newpassword

4 Features

4.1 Add events

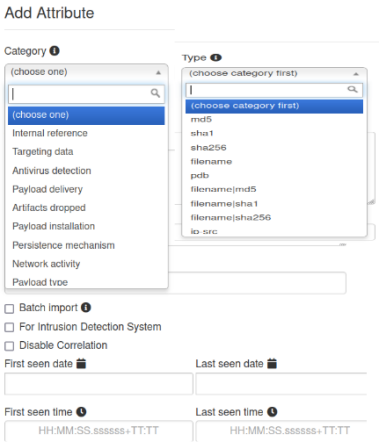
- A community can share threat events among the community



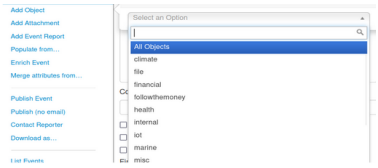
4.2 Data modelling

- MISP uses a flexible data model that allows users to define and share threat intelligence information in a structured way. This includes information about malware, indicators of compromise (IoCs), attack techniques, and more. Attributes and object types, Event Structure and Data sharing formats can be selected from various options in MISP.

- Add attribute options in adding events

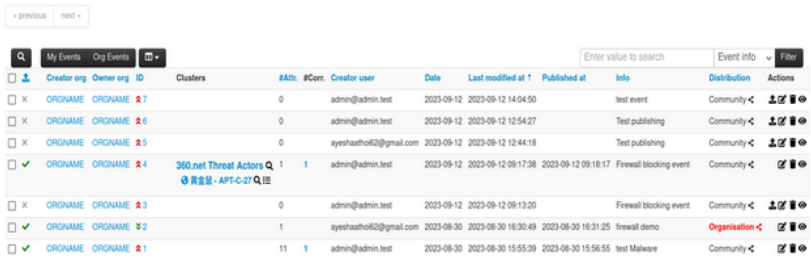


- Add object options in adding events



4.3 Data feeds

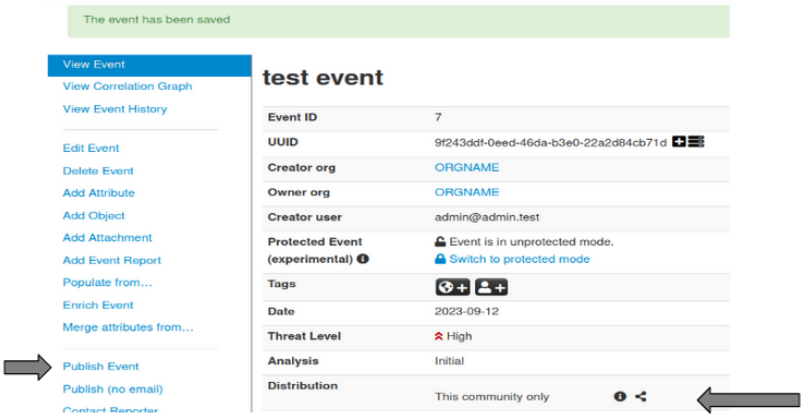
- Users in the community can see the news and threat events whenever they log in.
- Event feed



- News feed

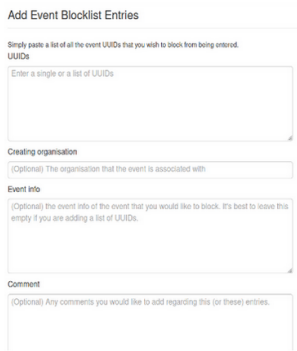
4.4 Sharing and Collaboration

- When the admin publishes the event among the users, it is shared among the community



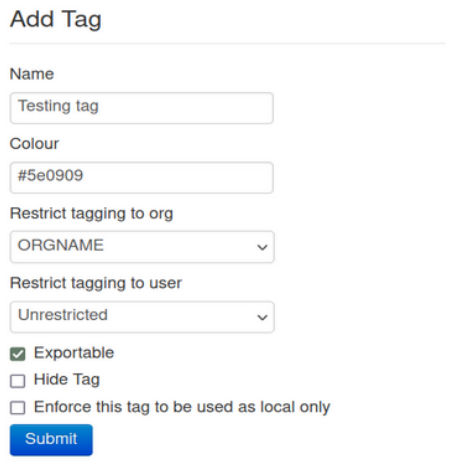
4.5 Block Events

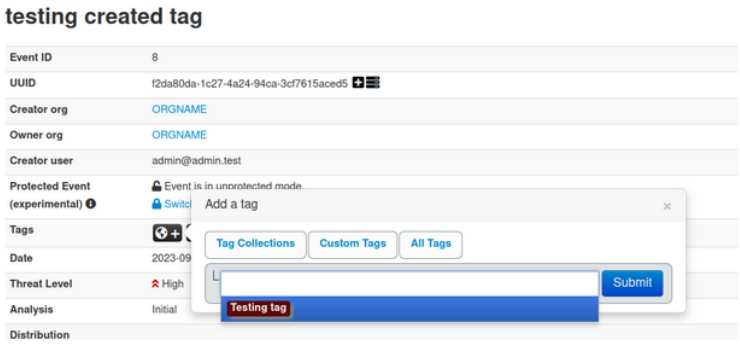
- If an event is undesirable, it can be blocked



4.6 Customized tag creation and usage

- Tag can be created and can be used in adding events

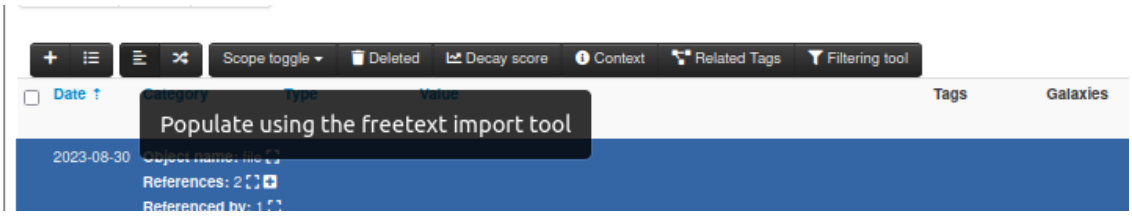


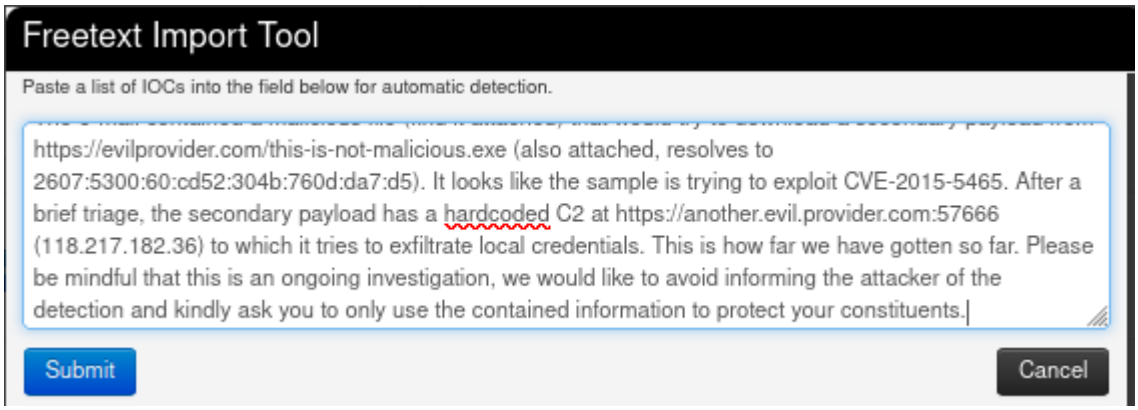


4.7 Adding attributes automatically

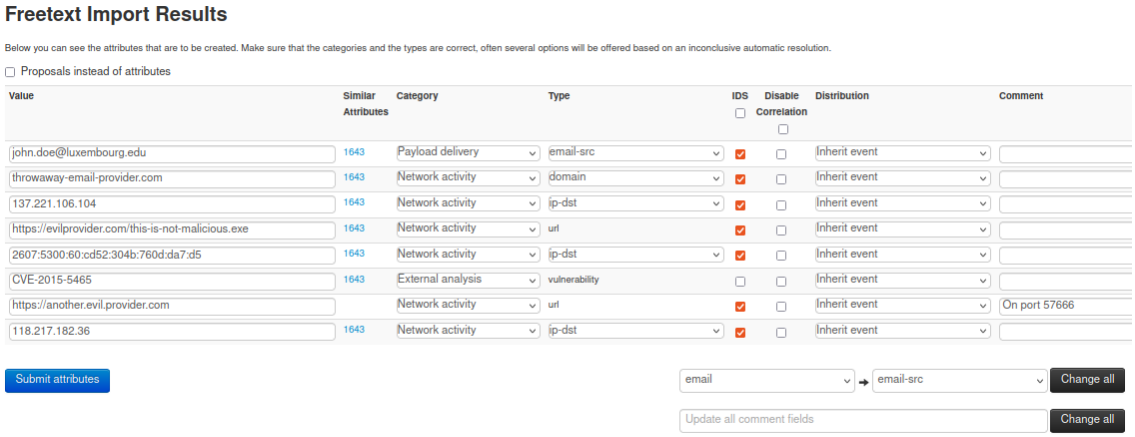
This helps to find the attributes of a given text. The user does not need to add attributes manually for every event.

- select an event to add attributes
- Select 'populate using the free text import tool'

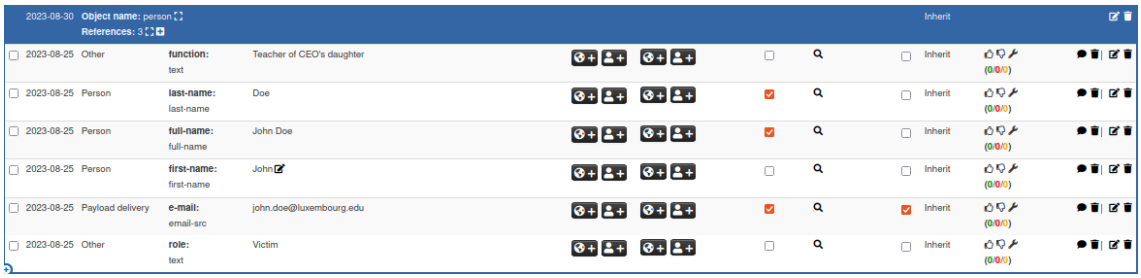




- submit the text to encode
- submit attributes



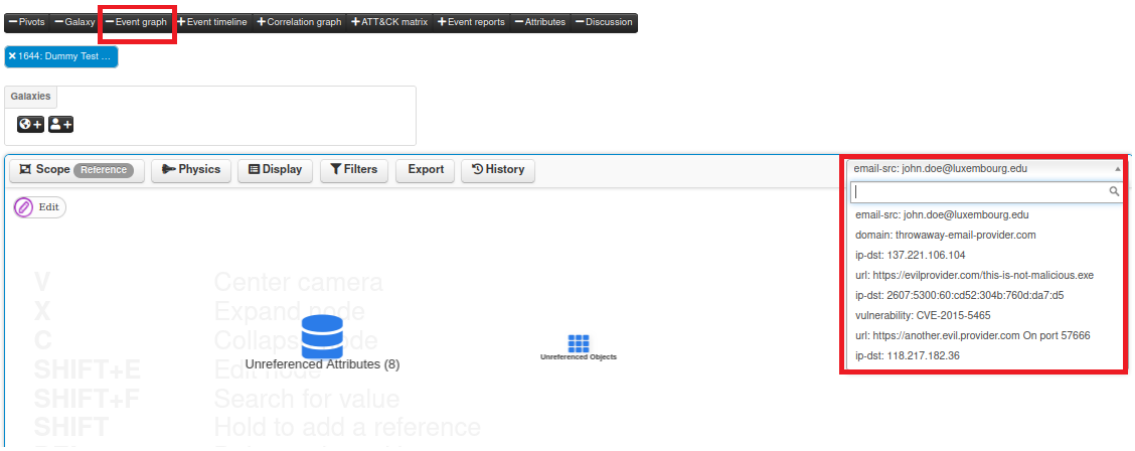
- Attributes will then be added automatically



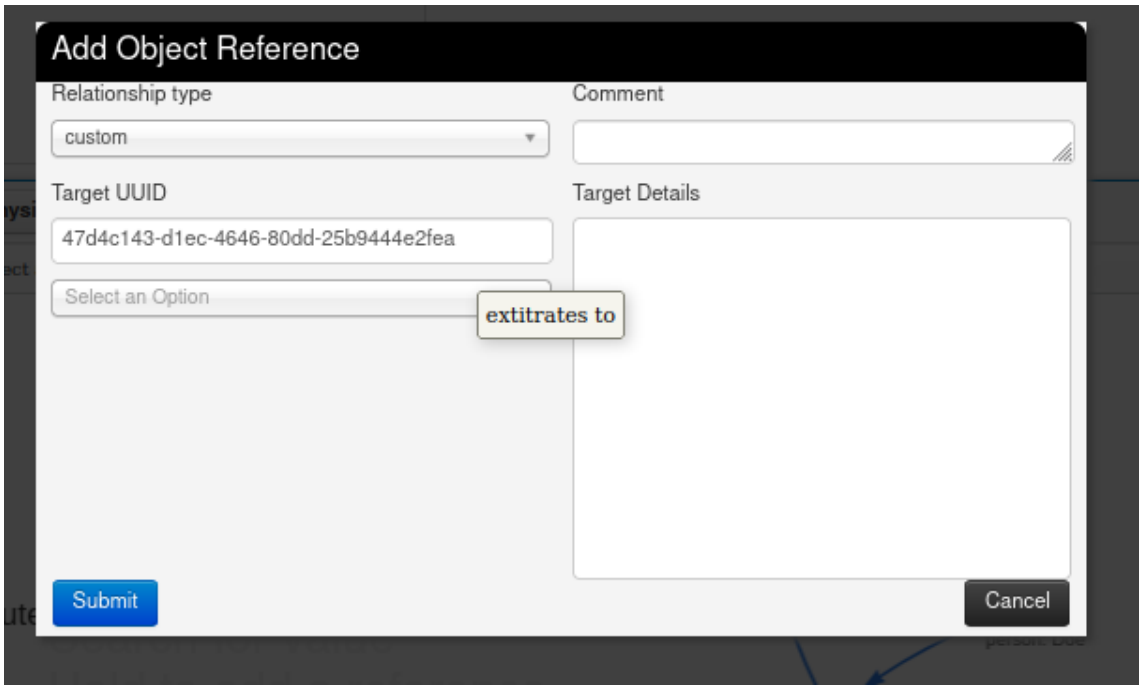
4.8 Event Graph Generation

An event graph helps to visualize the flow of incidents.

- select an event to generate an event graph
- Select the event graph option and load all attributes

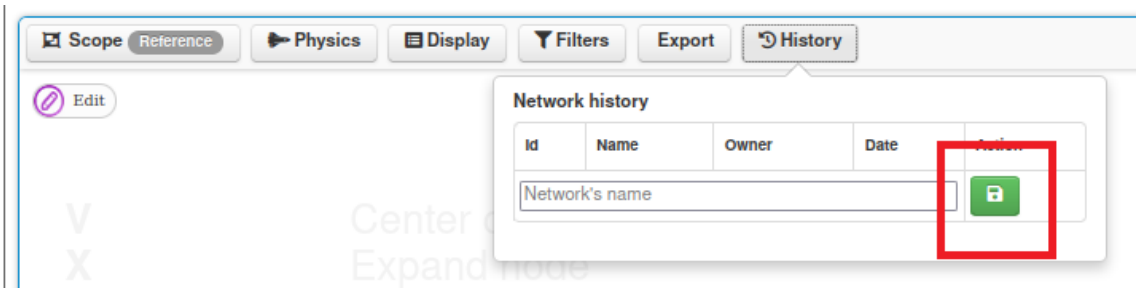


- The reference between objects and attributes can be added using the 'Edit ¿ Add Reference' option



- Finally, the event graph will look like this

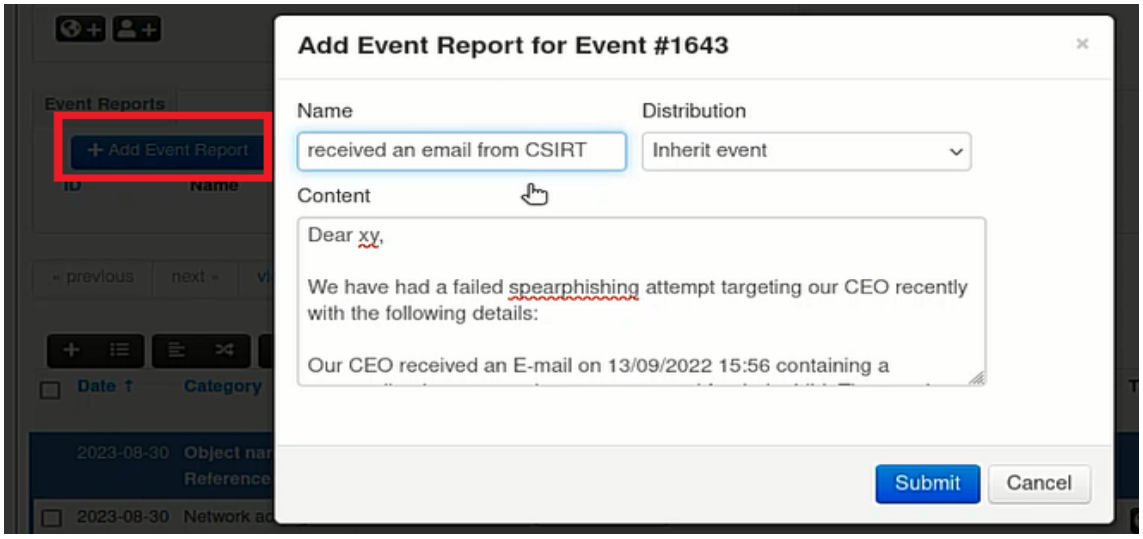
- The network should be saved to be used later



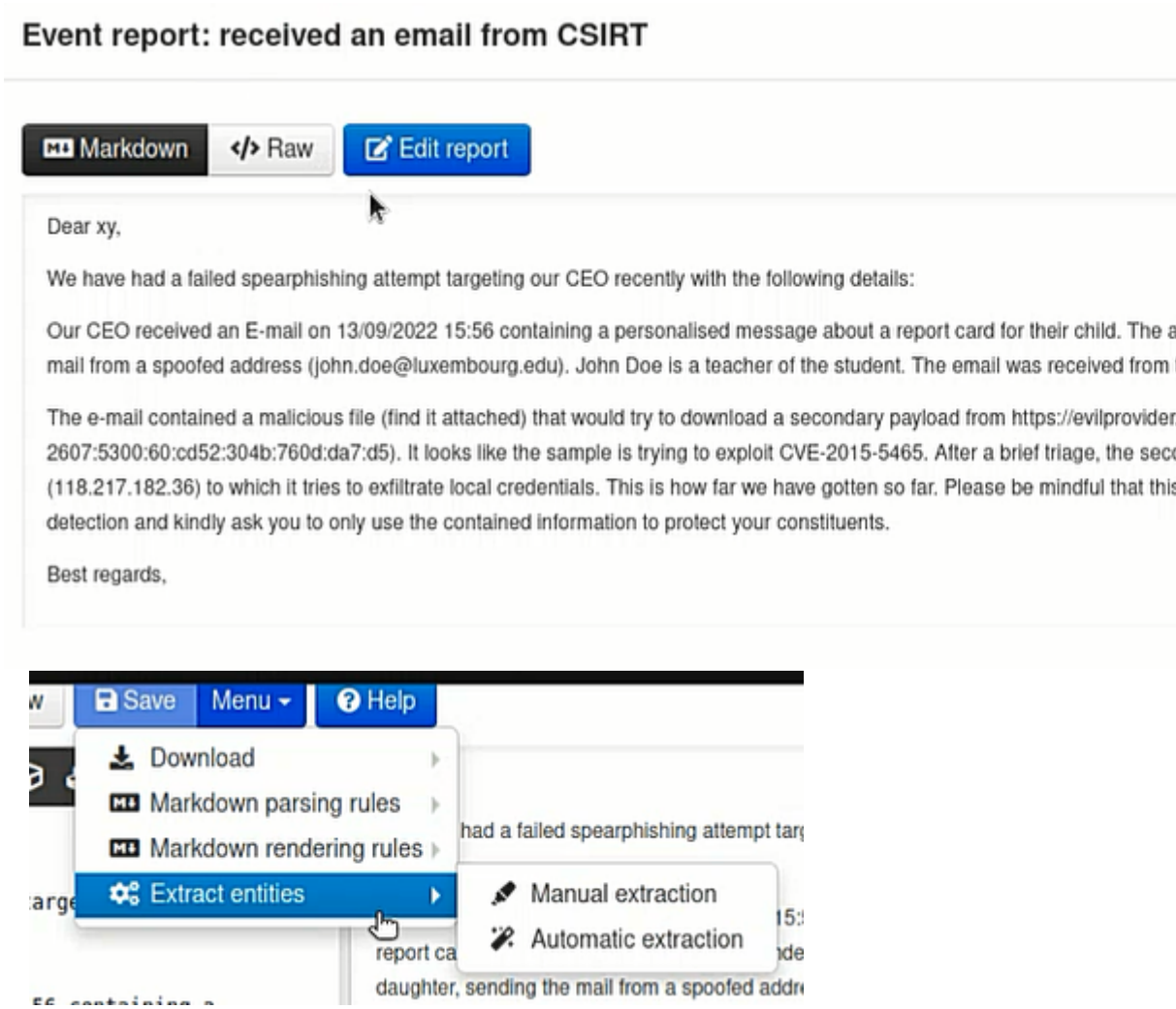
4.9 Generation of Event Report

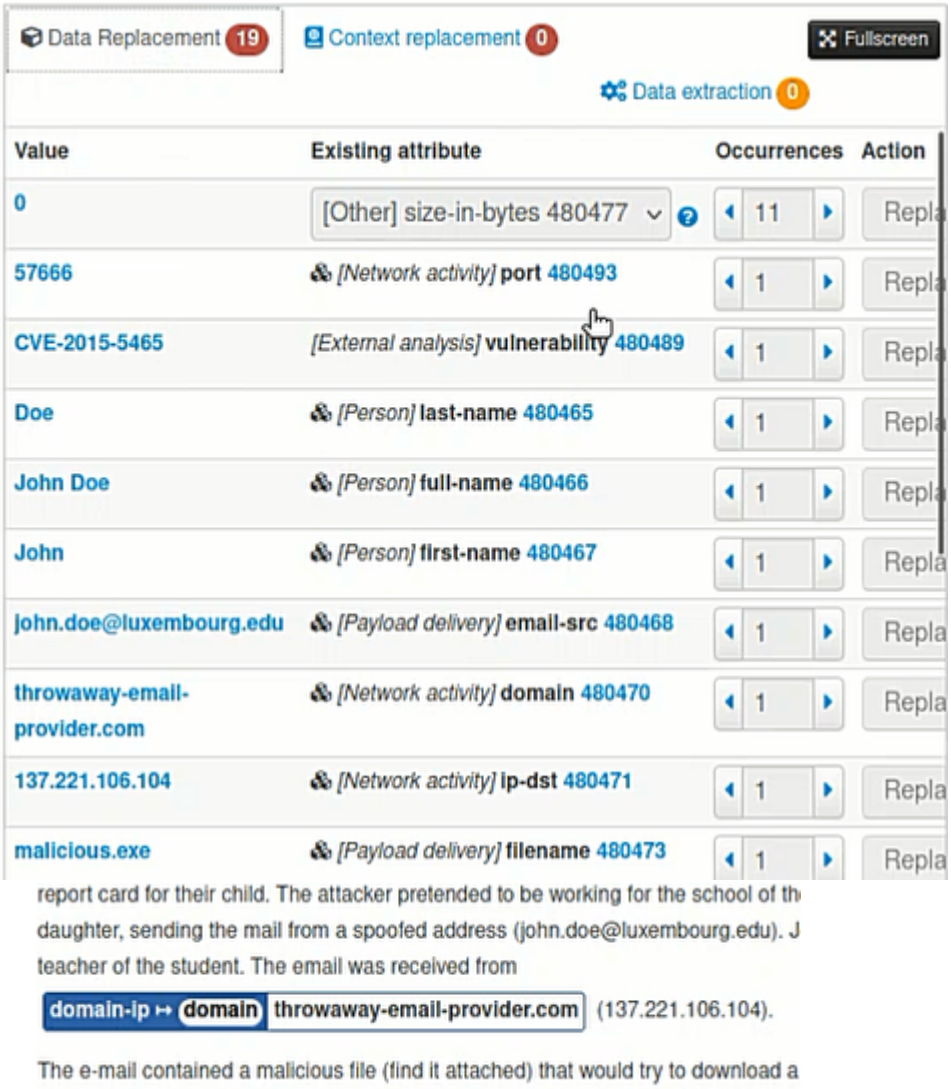
A documentation of an event.

- Create an event report



- Edit the report, adding the encoded entities in the report

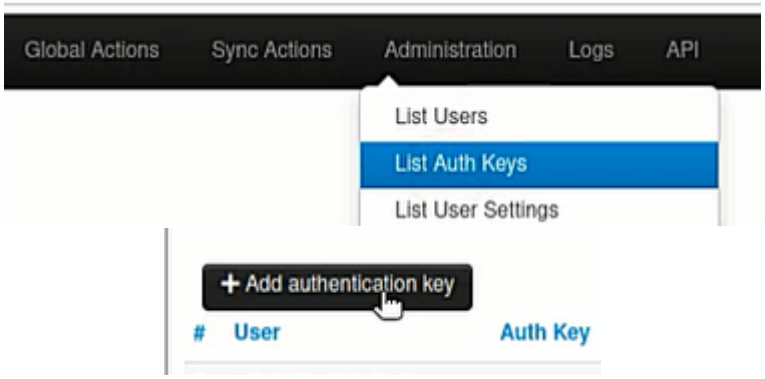

Event report: received an email from CSIRT
Markdown
Raw
Edit report
Dear xy,
We have had a failed spearphishing attempt targeting our CEO recently with the following details:
Our CEO received an E-mail on 13/09/2022 15:56 containing a personalised message about a report card for their child. The a
mail from a spoofed address (john.doe@luxembourg.edu). John Doe is a teacher of the student. The email was received from
The e-mail contained a malicious file (find it attached) that would try to download a secondary payload from https://evilprovider
2607:5300:60:cd52:304b:760d:da7:d5). It looks like the sample is trying to exploit CVE-2015-5465. After a brief triage, the seco
(118.217.182.36) to which it tries to exfiltrate local credentials. This is how far we have gotten so far. Please be mindful that this
detection and kindly ask you to only use the contained information to protect your constituents.
Best regards,
Save
Menu
Help
Download
Markdown parsing rules
Markdown rendering rules
Extract entities
Manual extraction
Automatic extraction
had a failed spearphishing attempt tar
report ca
daughter, sending the mail from a spoofed addr



4.10 MISP API

We can integrate the MIST database with any analyzer through its API feature. Here, we will incorporate the MISP instance with Cortex. At first, we create the API authentication key.

- From administration, select List Auth keys



- Fill in the data as required.

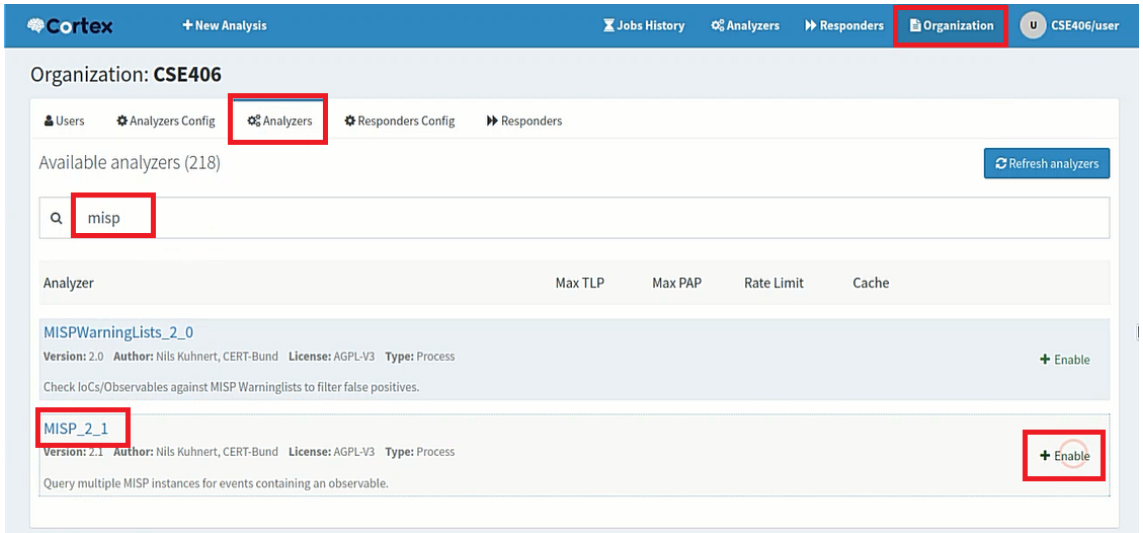
- Save the generated Auth key, as we cannot access it later.

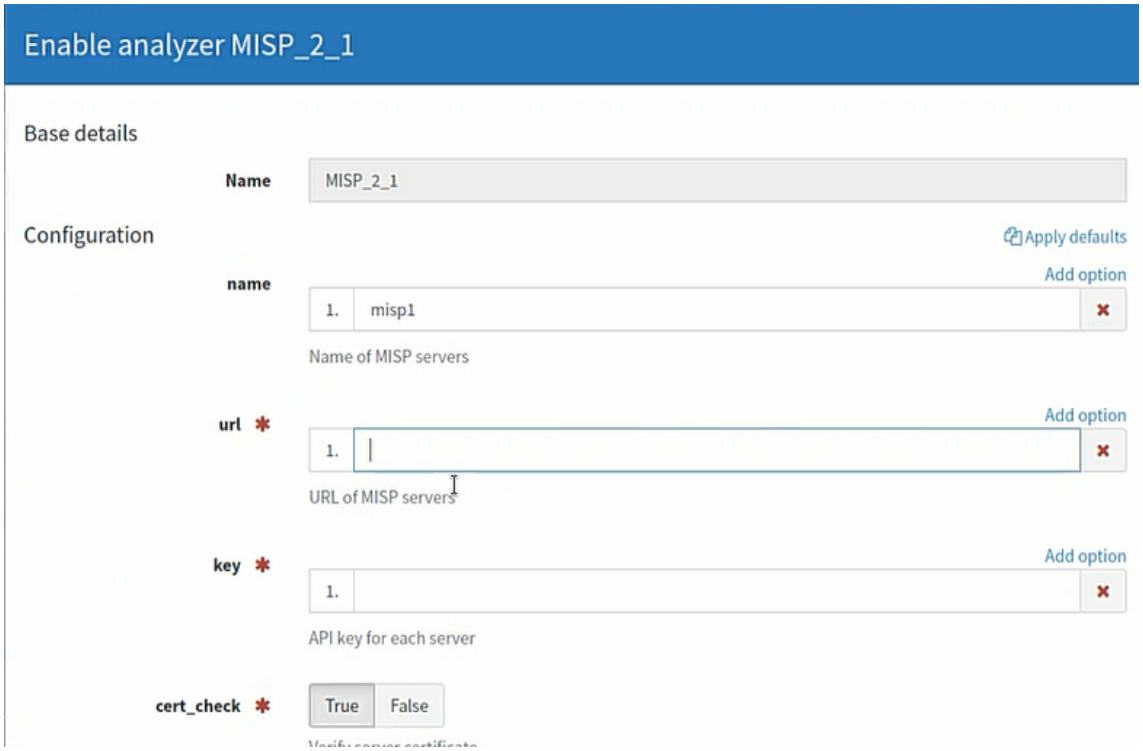
- The auth key is listed and ready to share.

#	User	Auth Key	Expiration	Last used
1	admin@admin.test	xvys••••••••••••••••••••••••••••••••XGxB	Indefinite	Never
2	admin@admin.test	8Zw0••••••••••••••••••••••••••••••••LiaG	2023-12-31 00:00:00	Never

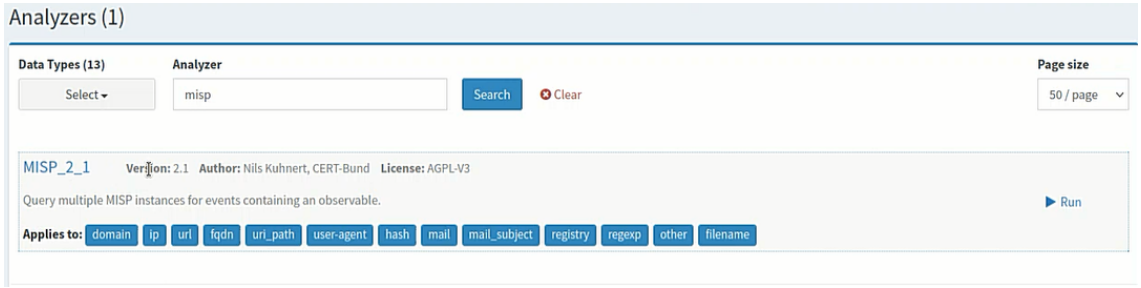
Now, We incorporate this with Cortex.

- log into the cortex.
- add the MISP instance



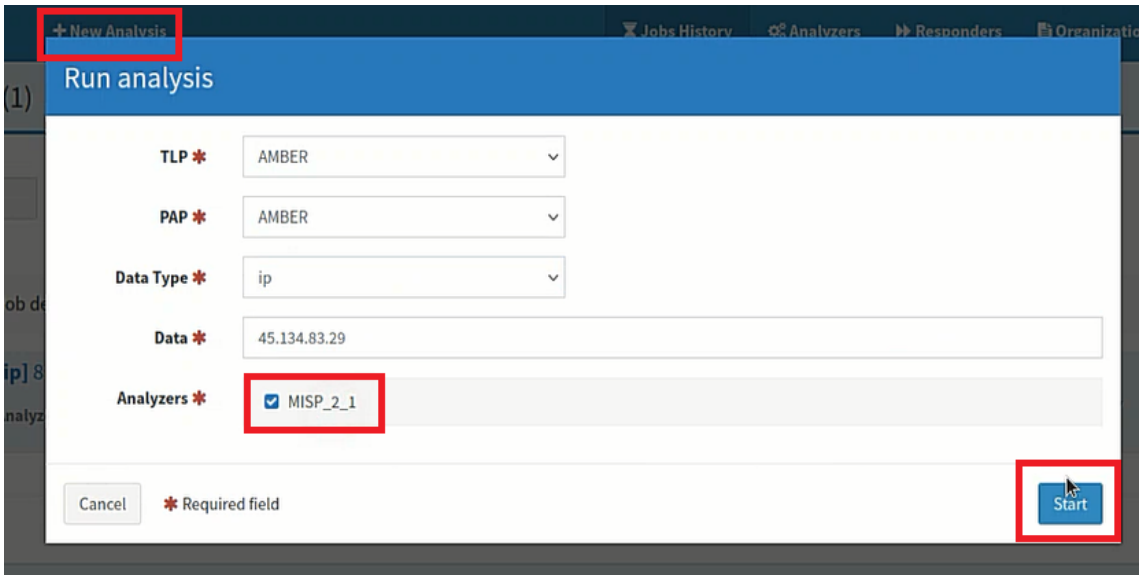


- MISP instance added



How to use it?

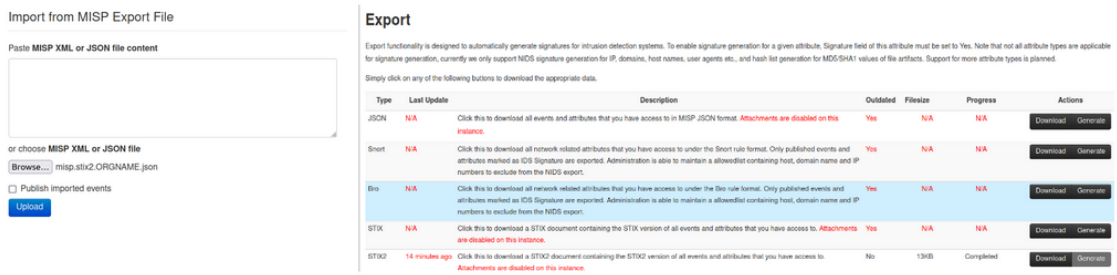
- Now we want to search for an IOC
- We need to select a new analysis and fill in the IOC info



- We get the complete details of IOC if available

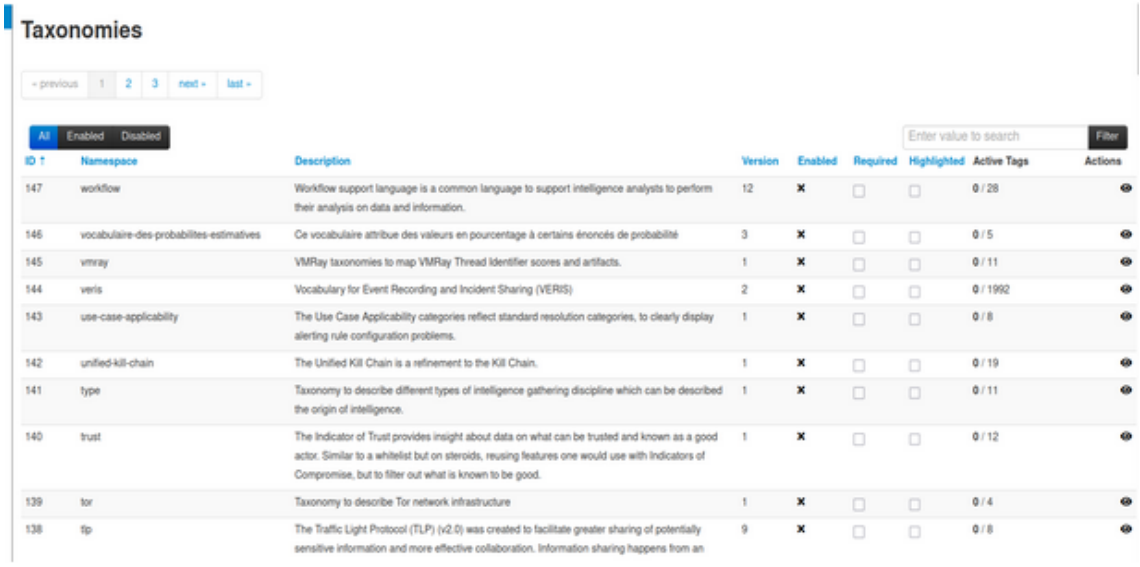
4.11 Import and Export events

- We can export the events we have created. Again we can import new events from a file to share in the community.



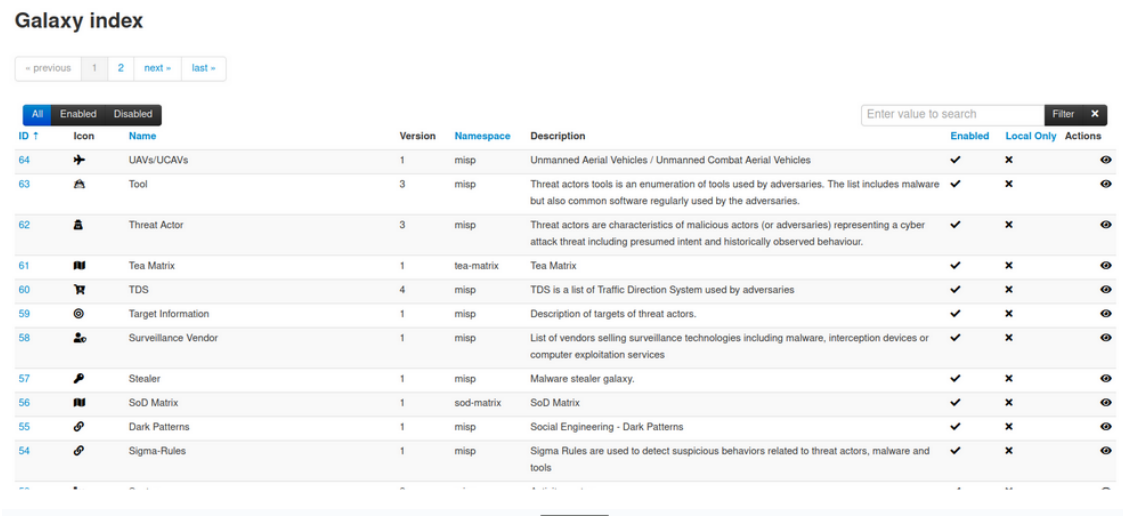
4.12 Taxonomy

- MISP taxonomies is a public repository of known vocabularies that can be used in threat information sharing.



4.13 Galaxy

- Galaxies in MISP are a method used to express a large object called a cluster that can be attached to MISP events or attributes.



Galaxy index

« previous | 1 | 2 | next » | last »

All | Enabled | Disabled

Enter value to search | Filter

ID ↑	Icon	Name	Version	Namespace	Description	Enabled	Local Only	Actions
64		UAVs/UCAVs	1	misp	Unmanned Aerial Vehicles / Unmanned Combat Aerial Vehicles	✓	✗	
63		Tool	3	misp	Threat actors tools is an enumeration of tools used by adversaries. The list includes malware but also common software regularly used by the adversaries.	✓	✗	
62		Threat Actor	3	misp	Threat actors are characteristics of malicious actors (or adversaries) representing a cyber attack threat including presumed intent and historically observed behaviour.	✓	✗	
61		Tea Matrix	1	tea-matrix	Tea Matrix	✓	✗	
60		TDS	4	misp	TDS is a list of Traffic Direction System used by adversaries	✓	✗	
59		Target Information	1	misp	Description of targets of threat actors.	✓	✗	
58		Surveillance Vendor	1	misp	List of vendors selling surveillance technologies including malware, interception devices or computer exploitation services	✓	✗	
57		Stealer	1	misp	Malware stealer galaxy.	✓	✗	
56		SoD Matrix	1	sod-matrix	SoD Matrix	✓	✗	
55		Dark Patterns	1	misp	Social Engineering - Dark Patterns	✓	✗	
54		Sigma-Rules	1	misp	Sigma Rules are used to detect suspicious behaviors related to threat actors, malware and tools	✓	✗	

5 unlisted youtube video link created by us

- https://youtu.be/fQAXdNxt4l8

6 Resources

- https://hdoc.csirt-tooling.org/tq-qyvTQTLeZ0wy-OPXjiw?view&fbclid=IwAR2ezxXYHF-PLIJ33tyQ2S_CwnT1dpoL_2GOGEuwRUBVNDeuIcBLk3A8J34
- Cortex installation guide: https://github.com/TheHive-Project/CortexDocs/blob/master/installation/install-guide.md