An Introduction to Cybersecurity

Information Sharing

MISP - Threat Sharing

Team CIRCL

MISP Project
https://www.misp-project.org/

MISP Training @ FIRST.org 2019
20190617
(11:45 - 12:45) Introduction to Information Sharing with MISP
(12:45 - 13:15) User perspective - diving into MISP functionalities and integration
(13:15 - 14:30) Lunch Break
(14:30 - 16:00) Admin perspective - Figuring out the health of your MISP instance.
(16:45 - 17:45) Building your information sharing communities
(17:45 - 18:15) Future - Sharing Ideas
During a malware analysis workgroup in 2012, we discovered that we worked on the analysis of the same malware.

We wanted to share information in an easy and automated way to avoid duplication of work.

Christophe Vandeplas (then working at the CERT for the Belgian MoD) showed us his work on a platform that later became MISP.

A first version of the MISP Platform was used by the MALWG and the increasing feedback of users helped us to build an improved platform.

MISP is now a community-driven development.
The Computer Incident Response Center Luxembourg (CIRCL) is a government-driven initiative designed to provide a systematic response facility to computer security threats and incidents. CIRCL is the CERT for the private sector, communes and non-governmental entities in Luxembourg and is operated by securitymadein.lu g.i.e.
CIRCL is mandated by the Ministry of Economy and acting as the Luxembourg National CERT for private sector.

CIRCL leads the development of the Open Source MISP threat intelligence platform which is used by many military or intelligence communities, private companies, financial sector, National CERTs and LEAs globally.

CIRCL runs multiple large MISP communities performing active daily threat-intelligence sharing.

Co-financed by the European Union
Connecting Europe Facility
There are many different types of users of an information sharing platform like MISP:

- **Malware reversers** willing to share indicators of analysis with respective colleagues.
- **Security analysts** searching, validating and using indicators in operational security.
- **Intelligence analysts** gathering information about specific adversary groups.
- **Law-enforcement** relying on indicators to support or bootstrap their DFIR cases.
- **Risk analysis teams** willing to know about the new threats, likelihood and occurrences.
- **Fraud analysts** willing to share financial indicators to detect financial frauds.
MISP MODEL OF GOVERNANCE

- GitHub/Gitter Issues
- MISP user-groups (10)
- MISP features and future development
- MISP project core team
- MISP Summit (/year)
- MISP Trainings (5-8)
- Experimental features
Sharing indicators for a detection matter.
  • ‘Do I have infected systems in my infrastructure or the ones I operate?’

Sharing indicators to block.
  • ‘I use these attributes to block, sinkhole or divert traffic.’

Sharing indicators to perform intelligence.
  • ‘Gathering information about campaigns and attacks. Are they related? Who is targeting me? Who are the adversaries?’

→ These objectives can be conflicting (e.g. False-positives have different impacts)
Sharing difficulties are not really technical issues but often it’s a matter of social interactions (e.g. trust).

Legal restriction
- "Our legal framework doesn’t allow us to share information."
- "Risk of information-leak is too high and it’s too risky for our organization or partners."

Practical restriction
- "We don’t have information to share."
- "We don’t have time to process or contribute indicators."
- "Our model of classification doesn’t fit your model."
- "Tools for sharing information are tied to a specific format, we use a different one."

---

1[https://www.misp-project.org/compliance/]
MISP features

- MISP\(^2\) is a threat information sharing free & open source software.
- MISP has **a host of functionalities** that assist users in creating, collaborating & sharing threat information - e.g. flexible sharing groups, **automatic correlation**, free-text import helper, event distribution & proposals.
- Many export formats which support IDSes / IPSes (e.g. Suricata, Bro, Snort), SIEMs (eg CEF), Host scanners (e.g. OpenIOC, STIX, CSV, yara), analysis tools (e.g. Maltego), DNS policies (e.g. RPZ).
- A rich set of MISP modules\(^3\) to add expansion, import and export functionalities.

---

\(^2\)https://github.com/MISP/MISP

\(^3\)https://www.github.com/MISP/misp-modules
To corroborate a finding (e.g. is this the same campaign?), reinforce an analysis (e.g. do other analysts have the same hypothesis?), confirm a specific aspect (e.g. are the sinkhole IP addresses used for one campaign?) or just find if this threat is new or unknown in your community.
Communities using MISP

- Communities are groups of users sharing within a set of common objectives/values.
- CIRCL operates multiple MISP instances with a significant user base (more than 950 organizations with more than 2400 users).
- **Trusted groups** running MISP communities in island mode (air gapped system) or partially connected mode.
- **Financial sector** (banks, ISACs, payment processing organizations) use MISP as a sharing mechanism.
- **Military and international organizations** (NATO, military CSIRTs, n/g CERTs,...).
- **Security vendors** running their own communities (e.g. Fidelis) or interfacing with MISP communities (e.g. OTX).
MISP core distributed sharing functionality

- MISP’s core functionality is sharing where everyone can be a consumer and/or a contributor/producer."
- Quick benefit without the obligation to contribute.
- Low barrier access to get acquainted to the system.
Events, Objects and Attributes in MISP

- MISP events are encapsulations for contextually linked information.
- MISP attributes initially started with a standard set of "cyber security" indicators.
- MISP attributes are purely based on usage (what people and organizations use daily).
- Evolution of MISP attributes is based on practical usage & users (e.g. the addition of financial indicators in 2.4).
- MISP objects are attribute compositions describing points of data using many facets, constructed along the lines of community and user defined templates.
- Galaxies granularly contextualise, classify & categorise data based on threat actors, preventive measures, tools used by adversaries.

attributes can be anything that helps describe the intent of the event package from indicators, vulnerabilities or any relevant information.
Indicators

Indicators contain a pattern that can be used to detect suspicious or malicious cyber activity.

Attributes in MISP can be network indicators (e.g. IP address), system indicators (e.g. a string in memory) or even bank account details.

- **A type (e.g. MD5, url) is how an attribute is described.**

- An attribute is always in a category (e.g. Payload delivery) which puts it in a context.
  - **A category is what describes** an attribute.

- An IDS flag on an attribute allows to determine if an attribute can be automatically used for detection.

---

IoC (Indicator of Compromise) is a subset of indicators
Sharing Attackers Techniques

- MISP integrates at event or attribute level MITRE’s Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK).
SUPPORTING SPECIFIC DATAMODEL
Helping Contributors in MISP

- Contributors can use the UI, API or using the freetext import to add events and attributes.
  - Modules existing in Viper (a binary framework for malware reverser) to populate and use MISP from the vty or via your IDA.

- Contribution can be direct by creating an event but users can propose attributes updates to the event owner.

- Users should not be forced to use a single interface to contribute.
**Example: Freetext Import in MISP**

![Freetext Import Tool](image)

**Freetext Import Results**

Below you can see the attributes that are to be created. Make sure that the categories and the types are correct, often several options will be offered based on an incomplete automatic resolution.

<table>
<thead>
<tr>
<th>Value</th>
<th>Category</th>
<th>Type</th>
<th>IDs</th>
<th>Comment</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.160.122.175</td>
<td>Network activity</td>
<td>ip-dist</td>
<td></td>
<td>Imported via the freetext import.</td>
<td></td>
</tr>
<tr>
<td>host.microsoft.com</td>
<td>Network activity</td>
<td>hostname</td>
<td></td>
<td>Imported via the freetext import.</td>
<td></td>
</tr>
<tr>
<td>b44727a0e3a348881b00301770000cd</td>
<td>Payload delivery</td>
<td>md5</td>
<td></td>
<td>Imported via the freetext import.</td>
<td></td>
</tr>
<tr>
<td><a href="https://www.github.com/MISP/MISP">https://www.github.com/MISP/MISP</a></td>
<td>Network activity</td>
<td>url</td>
<td></td>
<td>Imported via the freetext import.</td>
<td></td>
</tr>
</tbody>
</table>

![Imported via the freetext import.](image)
Supporting Classification

- Tagging is a simple way to attach a classification to an event or an attribute.
- Classification must be globally used to be efficient.
- MISP includes a flexible tagging scheme where users can select from more than 42 existing taxonomies or create their own taxonomy.
Delegate events publication to another organization (introduced in MISP 2.4.18).

- The other organization can take over the ownership of an event and provide **pseudo-anonymity to initial organization**.

Sharing groups allow custom sharing (introduced in MISP 2.4) per event or even at attribute level.

- Sharing communities can be used locally or even cross MISP instances.
- **Sharing groups** can be done at **event level or attributes level** (e.g. financial indicators shared to a financial sharing groups and cyber security indicators to CSIRT community).
Sightings support

- Sightings allow users to notify the community about the activities related to an indicator.
- In recent MISP versions, the sighting system supports negative sightings (FP) and expiration sightings.
- Sightings can be performed via the API, and the UI, even including the import of STIX sighting documents.
- Many use-cases for scoring indicators based on users sighting.
False-positives are a recurring challenge in information sharing.

In MISP 2.4.39, we introduced the misp-warninglists⁶ to help analysts in their day-to-day job.

Predefined lists of well-known indicators which are often false-positives like RFC1918 networks, public DNS resolver are included by default.

⁶https://github.com/MISP/misp-warninglists
Improving support of sharing within and outside an organization

- Even in a single organization, multiple use-cases of MISP can appear (groups using it for dynamic malware analysis correlations, dispatching notification).
- In MISP 2.4.51, we introduced the ability to have local MISP servers connectivity to avoid changes in distribution level. This allows to have mixed synchronization setup within and outside an organization.
- Feed support was also introduced to support synchronization between untrusted and trusted networks.
We maintain the default CIRCL OSINT feeds (TLP:WHITE selected from our communities) in MISP to allow users to ease their bootstrapping.

The format of the OSINT feed is based on standard MISP JSON output pulled from a remote TLS/HTTP server.

Additional content providers can provide their own MISP feeds. ([https://botvrij.eu/](https://botvrij.eu/))

Allows users to **test their MISP installations and synchronisation with a real dataset**.

Opening contribution to other threat intel feeds but also allowing the analysis of overlapping data.\(^7\)

\(^7\)A recurring challenge in information sharing
Information sharing practices come from usage and by example (e.g. learning by imitation from the shared information).

MISP is just a tool. What matters is your sharing practices. The tool should be as transparent as possible to support you.

Enable users to customize MISP to meet their community’s use-cases.

MISP project combines open source software, open standards, best practices and communities to make information sharing a reality.
MISP User Training - General usage of MISP

MISP - Threat Sharing

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
MISP - VM

- **Credentials**
  - MISP admin: admin@admin.test/admin
  - SSH: misp/Password1234

- **Available at the following location (VirtualBox and VMWare):**
  - https://www.circl.lu/misp-images/latest/
It is a bit broken.

- sudo -s
- cd /var/www/MISP/
- sudo pear install INSTALL/dependencies/Console_CommandLine/package.xml
- sudo pear install INSTALL/dependencies/Crypt_GPG/package.xml
- cd /usr/local/src/misp-modules
- pip3 install -r REQUIREMENTS
- pip3 install .
- reboot
Plan for this part of the training

- Data model
- Viewing data
- Creating data
- Co-operation
- Distribution
- Exports
MISP - Event (MISP’s basic building block)

<table>
<thead>
<tr>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creator org</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Analysis</td>
</tr>
<tr>
<td>Threat level</td>
</tr>
<tr>
<td>Distribution</td>
</tr>
</tbody>
</table>
MISP - Event (Attributes, giving meaning to events)
MISP - Event (Correlations on Similar Attributes)
MISP - EVENT (TAGS)

Event
- Creator org
- Description
- Analysis
- Threat level
- Distribution

Tag
- Name
- Colour

Event
- Creator org
- Description
- Analysis
- Threat level
- Distribution

Attribute
- Value
- Category
- Type
- IDS
- Comment
- Distribution

Correlation

Attribute
- Value
- Category
- Type
- IDS
- Comment
- Distribution

Proposal
- Value
- Category
- Type
- IDS
- Comment
- Org
MISP - Event (Taxonomies and Proposal Correlations)
MISP - Event (The state of the art MISP datamodel)
MISP - Viewing the Event Index

- Event Index
  - Event context
  - Tags
  - Distribution
  - Correlations

- Filters
MISP - Viewing an Event

Event View
- Event context
- Attributes
  - Category/type, IDS, Correlations
- Objects
- Galaxies
- Proposals
- Discussions

Tools to find what you are looking for
- Correlation graphs
The main tools to populate an event

- Adding attributes / batch add
- Adding objects and how the object templates work
- Freetext import
- Import
- Templates
- Adding attachments / screenshots
- API
What happens automatically when adding data?
- Automatic correlation
- Input modification via validation and filters (regex)
- Tagging / Galaxy Clusters

Various ways to publish data
- Publish with/without e-mail
- Publishing via the API
- Delegation
MISP - Using the Data

- Correlation graphs
- Downloading the data in various formats
- Cached exports
- API (explained later)
- Collaborating with users (proposals, discussions, emails)
Sync connections
Pull/push model
Previewing instances
Filtering the sync
Connection test tool
Cherry pick mode
MISP - Feeds explained (if no admin training)

- Feed types (MISP, Freetext, CSV)
- Adding/editing feeds
- Previewing feeds
- Local vs Network feeds
MISP - Distributions explained

- Your Organisation Only
- This Community Only
- Connected Communities
- All Communities
- Sharing Group
Download an event
Quick glance at the APIs
Download search results
Cached exports
MISP - Shorthand Admin (if no Admin Training)

- Settings
- Troubleshooting
- Workers
- Logs
MISP Training: MISP Deployment and Integration

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019 20190617
A COMMON INTEGRATION
Provisioning your MISP infrastructure depends heavily on the **number of attributes/events** (whether your dataset is below or above 50 million attributes).

Number of MISP instances and the overall design depends on the following factors:

- Is your community private? Are you gathering MISP events from other communities? Are you **publishing events to external** (trusted/untrusted) communities.
- Do you plan to have **automatic tools** (e.g. sandbox analysis or low-value information needing correlation or an analyst workbench) feeding MISP?
There is a **jungle of formats** with some vendors having little to no interest in keeping their users autonomous.

Attacks and threats require a **dynamic format** to be efficiently shared (e.g. from financial indicators to personal information).

**Review your current list of formats/vendors** to ensure a limited loss of information, especially when exporting from MISP to other formats (e.g. STIX not supporting financial indicators or taxonomies/galaxies).
Use case: Normalizing OSINT and Private Feeds

- Normalizing external input and feed into MISP (e.g. feed importer).
- Comparing feeds before import (how many similarities? false-positives?).
- Evaluating quality of information before import (warning-list lookup at feed evaluation).
One of the main goals of MISP is to feed protective or detection tools with data

- IDSes / IPSes (e.g. Suricata, Bro, Snort format as included in Cisco products)
- SIEMs (e.g. CEF, CSV or real-time ZMQ pub-sub or Sigma)
- Host scanners (e.g. OpenIOC, STIX, yara rule-set, CSV)
- Various analysis tools (e.g. Maltego)
- DNS policies (e.g. RPZ)

Various ways of exporting this data (downloads of the selected data, full exports, APIs)

The idea was to leave the selection process of the subset of data to be pushed to these up to the user using APIs.
SIEMs and MISP can be integrated with different techniques depending on the processes at your SOC or IR:

- Pulling events (via the API) or indicator lists at regular intervals in a given time frame to perform lookups.
- Subscribing to the MISP ZMQ pub-sub channel to directly get the published events and use these in a lookup process.
- Lookup expansion module in MISP towards the SIEM to have a direct view of the attributes matched against the SIEM.

The above options can be combined, depending on your organisation or requirements to increase coverage and detection.
A dashboard showing live data and statistics from the ZMQ pub-sub of one or more MISP instances.

Building **low-latency software** by consuming pub-sub channel provides significant advantages over standard API use.

Process information in **real-time** when it’s updated, created, published or gathered in MISP.

Demo!
New integrations: IR and threat hunting using MISP

- Close co-operation with the **Hive project** for IR
  - Interact with MISP directly from the Hive
  - Use both the MISP modules and the **Cortex** analysers in MISP or the Hive directly

- Using MISP to support your threat hunting via **McAfee OpenDXL**

The Hive integration

Alert Sources (SIEM, email, ...)

Alert Feeders

Raise alerts

Export cases

Poll events

Analyze observables

MISP

Enrich events

Additional analyzers

Expansion Modules

Search observables within MISP events

Analyzers
As **Sightings** can be positive, negative or even based on expiration, different use cases are possible:

- **Sightings** allow users to notify a MISP instance about the activities related to an indicator.
- Activities can be from a SIEM (e.g. Splunk lookup validation or **false-positive feedback**), a NIDS or honeypot devices\(^1\).
- Sighting can affect the API to limit the NIDS exports and improve the NIDS rule-set directly.

\(^1\)https://www.github.com/MISP/misp-sighting-tools
- info@circl.lu (if you want to join the CIRCL MISP sharing community)
- We welcome any contributions to the project, be it pull requests, ideas, github issues,...
Viper - Using MISP from your terminal

MISP - Threat Sharing

Team CIRCL

MISP Project
https://www.misp-project.org/

MISP Training @ FIRST.org 2019 20190617
Viper is a binary analysis and management framework. Its fundamental objective is to provide a solution to easily organize your collection of malware and exploit samples as well as your collection of scripts you created or found over the time to facilitate your daily research. Think of it as a Metasploit for malware researchers: it provides a terminal interface that you can use to store, search and analyze arbitrary files with and a framework to easily create plugins of any sort.
Solid CLI

Plenty of modules (PE files, *office, ELF, APK, ...)

Connection to 3rd party services (MISP, VirusTotal, cuckoo)

Connectors to 3rd party tools (IDA, radare)

Locale storage of your own zoo

Django interface is available (I’ve been told)
### Command List

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apk</td>
<td>Parse Android Applications</td>
</tr>
<tr>
<td>clamav</td>
<td>Scan file from local ClamAV daemon</td>
</tr>
<tr>
<td>cuckoo</td>
<td>Submit the file to Cuckoo Sandbox</td>
</tr>
<tr>
<td>debug</td>
<td>Parse McAfee BUP Files</td>
</tr>
<tr>
<td>edidistance</td>
<td>Edit distance on the filenames</td>
</tr>
<tr>
<td>elf</td>
<td>Extract information from ELF headers</td>
</tr>
<tr>
<td>email</td>
<td>Parse eml and msg email files</td>
</tr>
<tr>
<td>exif</td>
<td>Extract Exif MetaData</td>
</tr>
<tr>
<td>fuzzy</td>
<td>Search for similar files through fuzzy hashing</td>
</tr>
<tr>
<td>html</td>
<td>Parse html files and extract content</td>
</tr>
<tr>
<td>ida</td>
<td>Start IDA Pro</td>
</tr>
<tr>
<td>idx</td>
<td>Parse Java IDX files</td>
</tr>
<tr>
<td>image</td>
<td>Perform analysis on images</td>
</tr>
<tr>
<td>jar</td>
<td>Parse Java JAR archives</td>
</tr>
<tr>
<td>koodous</td>
<td>Interact with Koodous</td>
</tr>
<tr>
<td>lastline</td>
<td>Submit files and retrieve reports from LastLine (default will print short summary)</td>
</tr>
<tr>
<td>macho</td>
<td>Get Macho OSX Headers</td>
</tr>
<tr>
<td>misp</td>
<td>Upload and query IOCs to/from a MISP instance</td>
</tr>
<tr>
<td>office</td>
<td>Office Document Parser</td>
</tr>
<tr>
<td>pdf</td>
<td>Parse and analyze PDF documents</td>
</tr>
<tr>
<td>pdns</td>
<td>Query a Passive DNS server</td>
</tr>
<tr>
<td>pe</td>
<td>Extract information from PE32 headers</td>
</tr>
<tr>
<td>pssl</td>
<td>Query a Passive SSL server</td>
</tr>
<tr>
<td>pst</td>
<td>Process PST Files for Attachment</td>
</tr>
<tr>
<td>r2</td>
<td>Start Radare2</td>
</tr>
<tr>
<td>rat</td>
<td>Extract information from known RAT families</td>
</tr>
<tr>
<td>reports</td>
<td>Online Sandboxes Reports</td>
</tr>
<tr>
<td>shellcode</td>
<td>Search for known shellcode patterns</td>
</tr>
<tr>
<td>size</td>
<td>Size command to show/scan/cluster files</td>
</tr>
<tr>
<td>strings</td>
<td>Extract strings from file</td>
</tr>
<tr>
<td>swf</td>
<td>Parse, analyze and decompress Flash objects</td>
</tr>
<tr>
<td>triage</td>
<td>Perform some initial triaging and tagging of the file</td>
</tr>
</tbody>
</table>
Full featured **CLI for MISP**

- **Remote storage** of your zoo
- Search / **Cross check with VirusTotal**
- Create / Update / Show / Publish Event
- Download / Upload Samples
- Mass export / Upload / Download
- Get Yara rules
viper > misp -h
usage: misp [-h] [--url URL] [-k KEY] [-v]
{upload,download,search,check_hashes,yara,pull,create_event,add,show,open,
publish,version,store}
...

Upload and query IOCs to/from a MISP instance

positional arguments:
{upload,download,search,check_hashes,yara,pull,create_event,add,show,open,publish,version,store}
  upload                  Send malware sample to MISP.
  download               Download malware samples from MISP.
  search                 Search in all the attributes.
  check_hashes           Crosscheck hashes on VT.
  yara                   Get YARA rules of an event.
  pull                   Initialize the session with an existing MISP event.
  create_event           Create a new event on MISP and initialize the session
                         with it.
  add                    Add attributes to an existing MISP event.
  show                   Show attributes to an existing MISP event.
  open                   Open a sample from the temp directory.
  publish                Publish an existing MISP event.
  version                Returns the version of the MISP instance.
  store                  Store the current MISP event in the current project.

optional arguments:
-h, --help              show this help message and exit
--url URL               URL of the MISP instance
-k KEY, --key KEY       Your key on the MISP instance
-v, --verify            Disable certificate verification (for self-signed)
Viper & VT

- Searches for hashes/ips/domains/URLs from the current MISP event, or download the samples
- Download samples from current MISP event
- Download all samples from all the MISP events of the current session
Lookup the file on VirusTotal

optional arguments:
-h, --help
   show this help message and exit
--search SEARCH
   Search a hash.
-c COMMENT [COMMENT ...], --comment COMMENT [COMMENT ...]
   Comment to add to the file
-d, --download
   Hash of the file to download
-dl, --download_list
   List the downloaded files
-do DOWNLOAD_OPEN, --download_open DOWNLOAD_OPEN
   Open a file from the list of the DL files (ID)
-don DOWNLOAD_OPEN_NAME, --download_open_name DOWNLOAD_OPEN_NAME
   Open a file by name from the list of the DL files (NAME)
-dd DOWNLOAD_DELETE, --download_delete DOWNLOAD_DELETE
   Delete a file from the list of the DL files can be an ID or all.
-s, --submit
   Submit file or a URL to VirusTotal (by default it only looks up the hash/url)
-i IP, --ip IP
   IP address to lookup in the passive DNS
-dm DOMAIN, --domain DOMAIN
   Domain to lookup in the passive DNS
-u URL, --url URL
   URL to lookup on VT
-v, --verbose
   Turn on verbose mode.
-m {hashes,ips,domains,urls,download,download_all}, --misp {hashes,ips,domains,urls,download,download_all}
   Searches for the hashes, ips, domains or URLs from the current MISP event, or download the samples if possible. Be careful with download_all: it will download *all* the samples of all the MISP events in the current project.
Extra features

- Link to a MISP event
- Local storage of the MISP event
- On the fly cross-check of MISP attributes with 3rd party services
- Never leaving your CLI!
Other modules

- Fully featured CLI for **Passive SSL**
- Fully featured CLI for **Passive DNS**
- Can launch Radare2 or IDA
viper > pssl -h

optional arguments:
-h, --help show this help message and exit
--url URL URL of the Passive SSL server (No path)
-u USER, --user USER Username on the PSSL instance
-p PASSWORD, --password PASSWORD Password on the PSSL instance
-i IP, --ip IP IP to query (can be a block, max /23).
-c CERT, --cert CERT SHA1 of the certificate to search.
-f FETCH, --fetch FETCH SHA1 of the certificate to fetch.
-v, --verbose Turn on verbose mode.
-m {ips}, --misp {ips} Searches for the ips from the current MISP event
viper > pdns -h
[query]

Query a Passive DNS server

positional arguments:
  query  Domain or IP address to query

optional arguments:
  -h, --help        show this help message and exit
  --url URL         URL of the Passive DNS server
  -u USER, --user USER  Username on the PDNS instance
  -p PASSWORD, --password PASSWORD
  -v, --verbose    Turn on verbose mode.
  -m {ips,domains}, --misp {ips,domains}  Searches for the ips or domains from the current MISP event
We welcome new functionalities and pull requests.
mail_to_misp

Connect your mail infrastructure to MISP to create

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
You receive emails with IoC's inside
How to create an event out of it?
Create event manually and copy paste
→ This works once or twice
Forwarding the email would be nice
→ mail_to_misp
Features: Email Handling

- Extraction of URLs and IP addresses and port numbers
- Extraction of hostnames from URLs
- Extraction of hashes (MD5, SHA1, SHA256)
- DNS expansion
- Subject filters
- Refanging of URLs ('hxxp://...')
- ... and more
Add tags automatically
Ignore ’whitelisted’ domains
Configurable list of attributes not to enable the IDS flag
DNS expansion
Automatically create ’external analysis’ links based on filter list (e.g. VirusTotal, malwr.com)
Automatically filter out attributes that are on a server side warning list
Support for value sighting
... and more
IMPLEMENTATION

- Legacy
  - Email → Apple Mail → Mail rule → AppleScript → AppleScript → mail_to_misp → PyMISP → MISP
  - Email → Thunderbird → Mail rule → filterscript → thunderbird_wrapper → mail_to_misp → PyMISP → MISP

- Postfix and others
  - Email → mail_to_misp
**INSTALLATION**

- **mail_to_misp**
  1. `git clone`  
     `git://github.com/MISP/mail_to_misp.git`  
  2. Install dependencies - See Github site

- **MTA (Postfix or alike)**
  1. Setup a new email address in the aliases file (e.g. `/etc/aliases`)  
     `misp_handler: "|/path/to/mail_to_misp.py -"`  
  2. Rebuild the DB  
     `sudo newaliases`  
  3. Configure `mail_to_misp_config.py`

```python
misp_url = 'http://127.0.0.1/'  
misp_key = 's5jPWCIud36Z8XHgsiCVI7SaL1XsMTyfEsN45tTe'  
misp_verifycert = True  
body_config_prefix = 'm2m'  
...
```
Exercise: mail_to_misp.py

Bonus:
https://github.com/MISP/mail_to_misp_test

./mail_to_misp.py --r mail_to_misp_test/simple_forward.eml

Bonus: Fake-SMTPD spamtrap

./fake_smtp.py
telnet 127.0.0.1 2526
  Trying 127.0.0.1...
  Connected to 127.0.0.1.
  Escape character is '^]'.
  220 misp Python SMTP 1.1
  helo misp
  250 misp
  mail from: mikel
  250 OK
  rcpt to: mtwo
  250 OK
  data
  354 End data with <CR><LF>.<CR><LF>
MISP User Training - Administration of MISP 2.4

MISP Threat Sharing

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
VM can be downloaded at https://www.circl.lu/misp-training/

Credentials
- MISP admin: admin@admin.test/admin
- SSH: misp/Password1234

2 network interfaces
- NAT
- Host only adapter

Start the enrichment system by typing:
- cd /home/misp/misp-modules/bin
- python3 misp-modules.py
Plan for this part of the training
- User and Organisation administration
- Sharing group creation
- Templates
- Tags and Taxonomy
- Whitelisting and Regexp entries
- Setting up the synchronisation
- Scheduled tasks
- Feeds
- Settings and diagnostics
- Logging
- Troubleshooting and updating
Add new user (andras.iklody@circl.lu)
NIDS SID, Organisation, disable user
Fetch the PGP key
Roles
  - Re-using standard roles
  - Creating a new custom role
Send out credentials
MISP - Creating Organisations

- Adding a new organisation
- UUID
- Local vs External organisation
- Making an organisation self sustaining with Org Admins
- Creating a sync user
MISP - Sharing groups

- The concept of a sharing group
- Creating a sharing group
- Adding extending rights to an organisation
- Include all organisations of an instance
- Not specifying an instance
- Making a sharing group active
- Reviewing the sharing group
Why templating?
Create a basic template
Text fields
Attribute fields
Attachment fields
Automatic tagging
git submodule init && git submodule update

Loading taxonomies

Enabling taxonomies and associated tags

Tag management

Exportable tags
- git submodule init && git submodule update
- Enabling objects (and what about versioning)
MISP - Whitelisting, Regexp entries, Warninglists

- Block from exports - whitelisting
- Block from imports - blacklisting via regexp
- Modify on import - modification via regexp
- Maintaining the warninglists
MISP - Setting up the synchronisation

- Requirements - versions
- Pull/Push
- One way vs Two way synchronisation
- Exchanging sync users
- Certificates
- Filtering
- Connection test tool
- Previewing an instance
- Cherry picking and keeping the list updated
How to schedule the next execution
Frequency, next execution
What happens if a job fails?
MISP - Setting up the synchronisation

- MISP Feeds and their generation
- PyMISP
- Default free feeds
- Enabling a feed
- Previewing a feed and cherry picking
- Feed filters
- Auto tagging
MISP - Settings and diagnostics

- Settings
  - Settings interface
  - The tabs explained at a glance
  - Issues and their severity
  - Setting guidance and how to best use it
- Basic instance setup
- Additional features released as hotfixes
- Customise the look and feel of your MISP
- Default behaviour (encryption, e-mailing, default distributions)
- Maintenance mode
- Disabling the e-mail alerts for an initial sync
Plugins

- Enrichment Modules
- RPZ
- ZeroMQ
Diagnostics
- Updating MISP
- Writeable Directories
- PHP settings
- Dependency diagnostics
Workers

- What do the background workers do?
- Queues
- Restarting workers, adding workers, removing workers
- Worker diagnostics (queue size, jobs page)
- Clearing worker queues
- Worker and background job debugging
Seeking help

- Dump your settings to a file!
- Make sure to sanitise it
- Send it to us together with your issue to make our lives easier
- Ask Github (https://github.com/MISP/MISP)
- Have a chat with us on gitter (https://gitter.im/MISP/MISP)
- Ask the MISP mailing list
- If this is security related, drop us a PGP encrypted email to mailto:info@circl.lu
MISP - Logging

- Audit logs in MISP
- Enable IP logging / API logging
- Search the logs, the fields explained
- External logs
  - /var/www/MISP/app/tmp/logs/error.log
  - /var/www/MISP/app/tmp/logs/resque-worker-error.log
  - /var/www/MISP/app/tmp/logs/resque-scheduler-error.log
  - /var/www/MISP/app/tmp/logs/resque-[date].log
  - /var/www/MISP/app/tmp/logs/error.log
  - apache access logs
MISP - Updating MISP

- git pull
- git submodule init && git submodule update
- reset the permissions if it goes wrong according to the INSTALL.txt
- when MISP complains about missing fields, make sure to clear the caches
  - in /var/www/MISP/app/tmp/cache/models remove myapp*
  - in /var/www/MISP/app/tmp/cache/persistent remove myapp*
- No additional action required on hotfix level
- Read the migration guide for major and minor version changes
MISP - Administrative tools

- Upgrade scripts for minor / major versions
- Maintenance scripts
Information Sharing and Taxonomies

Practical Classification of Threat Indicators using MISP

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019 20190617
Tagging is a simple way to attach a classification to an event or an attribute.

In the early version of MISP, tagging was local to an instance.

Classification must be globally used to be efficient.

After evaluating different solutions of classification, we build a new scheme using the concept of machine tags.
Triple tag or machine tag was introduced in 2004 to extend geotagging on images.

A machine tag is just a tag expressed in way that allows systems to parse and interpret it.

Still have a human-readable version:

```
admiralty-scale:Source Reliability="Fairly reliable"
```
Taxonomies are implemented in a simple JSON format.
Anyone can create their own taxonomy or reuse an existing one.
The taxonomies are in an independent git repository¹.
These can be freely reused and integrated in other threat intel tools.
Taxonomies are licensed under CCo (public domain) except if the taxonomy author decided to use another license.

¹https://www.github.com/MISP/misp-taxonomies/
Existing Taxonomies

- NATO - **Admiralty Scale**
- CIRCL Taxonomy - **Schemes of Classification in Incident Response and Detection**
- eCSIRT and IntelMQ incident classification
- EUCI **EU classified information marking**
- Information Security Marking Metadata from DNI (Director of National Intelligence - US)
- NATO Classification Marking
- OSINT **Open Source Intelligence - Classification**
- TLP - **Traffic Light Protocol**
- Vocabulary for Event Recording and Incident Sharing - **VERIS**
- and many more like ENISA, Europol, or the draft FIRST SIG Information Exchange Policy.
Want to write your own taxonomy? 1/2

```json
1 {  
2   "namespace": "admiralty-scale",
3   "description": "The Admiralty Scale (also called the NATO System) is used to rank the reliability of a source and the credibility of an information."
4   "version": 1,
5   "predicates": [  
6     {  
7       "value": "source-reliability",
8       "expanded": "Source Reliability"
9     },
10    {  
11      "value": "information-credibility",
12      "expanded": "Information Credibility"
13    }
14   ],
15  ....
```
Publishing your taxonomy is as easy as a simple git pull request on misp-taxonomies².

²https://github.com/MISP/misp-taxonomies
MISP administrator can just import (or even cherry pick) the namespace or predicates they want to use as tag.
- Tags can be exported to other instances.
- Tags are also accessible via the MISP REST API.
Applying rules for distribution based on tags:
Tags can be used to set events or attributes for **further processing by external tools** (e.g. VirusTotal auto-expansion using Viper).

Ensuring a classification manager **classifies the events before release** (e.g. release of information from air-gapped/classified networks).

**Enriching IDS export** with tags to fit your NIDS deployment.

Using **IntelMQ** and MISP together to process events (tags limited per organization introduced in MISP 2.4.49).
Future functionalities related to MISP taxonomies

- **Sighting** support (thanks to NCSC-NL) is integrated in MISP allowing to auto expire IOC based on user detection.

- Adjusting taxonomies (adding/removing tags) based on their score or visibility via sighting.

- Simple taxonomy editors to help non-technical users to create their taxonomies.

- **Filtering mechanisms** in MISP to rename or replace taxonomies/tags at pull and push synchronisation.

- More public taxonomies to be included.
■ **Python module** to handle the taxonomies
■ **Offline** and online mode (fetch the newest taxonomies from GitHub)
■ Simple **search** to make tagging easy
■ Totally independant from MISP
■ **No external dependencies** in offline mode
■ Python3 only
■ Can be used to create & **dump a new taxonomy**
from pytaxonomies import Taxonomies

taxonomies = Taxonomies()
taxonomies.version
# => '20160725'
taxonomies.description
# => 'Manifest file of MISP taxonomies available.'
list(taxonomies.keys())
# => ['tlp', 'eu—critical—sectors', 'de—vs', 'osint', 'circl', 'veris',
#     'ecsirt', 'dhs—ciip—sectors', 'fr—classif', 'misp', 'admiralty—scale', ...]
taxonomies.get('enisa').description
# 'The present threat taxonomy is an initial version that has been developed on
# the basis of available ENISA material. This material has been used as an ENISA—internal
# structuring aid for information collection and threat consolidation purposes.
# It emerged in the time period 2012—2015.'
print(taxonomies.get('circl'))
# circl:incident—classification="vulnerability"
# circl:incident—classification="malware"
# circl:incident—classification="fastflux"
# circl:incident—classification="system—compromise"
# circl:incident—classification="sql—injection"
# ....
print(taxonomies.get('circl').machinetags_expanded())
# circl:incident—classification="Phishing"
# circl:incident—classification="Malware"
# circl:incident—classification="XSS"
# circl:incident—classification="Copyright issue"
# circl:incident—classification="Spam"
# circl:incident—classification="SQL Injection"
False-positive is a **common issue** in threat intelligence sharing.

It’s often a contextual issue:
- false-positive might be different per community of users sharing information.
- organization might have their **own view** on false-positive.

Based on the success of the MISP taxonomy model, we build misp-warninglists.
MISP warning lists

- MISP warning lists are lists of well-known indicators that can be associated to potential false positives, errors or mistakes.

- Simple JSON files

```json
{  
  "name": "List of known public DNS resolvers",
  "version": 2,
  "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", ...
  ]
}
```
The warning lists are integrated in MISP to display an info/warning box at the event and attribute level.

Enforceable via the API where all attributes that have a hit on a warninglist will be excluded.

This can be enabled at MISP instance level.

Default warning lists can be enabled or disabled like known public resolver, multicast IP addresses, hashes for empty values, rfc1918, TLDs or known google domains.

The warning lists can be expanded or added in JSON locally or via pull requests.

Warning lists can be also used for critical or core infrastructure warning, personally identifiable information...
https://github.com/MISP/MISP
https://github.com/MISP/misp-taxonomies
https://github.com/MISP/PyTaxonomies
https://github.com/MISP/misp-warninglists
info@circl.lu (if you want to join one of the MISP community operated by CIRCL)
PGP key fingerprint: CA57 2205 C002 4E06 BA70 BE89 EAAD CFFC 22BD 4CD5
Extending MISP with Python modules

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
Why we want to go more modular...

Ways to extend MISP before modules

▶ APIs (PyMISP, MISP API)
  ■ Works really well
  ■ No integration with the UI

▶ Change the core code
  ■ Have to change the core of MISP, diverge from upstream
  ■ Needs a deep understanding of MISP internals
  ■ Let’s not beat around the bush: Everyone hates PHP
Goals for the module system

- Have a way to extend MISP without altering the core
- Get started **quickly** without a need to study the internals
- Make the **modules as light weight as possible**
  - Module developers should only have to worry about the data transformation
  - Modules should have a simple and clean skeleton
- In a friendlier language - **Python**
MISP modules - extending MISP with Python scripts

- Extending MISP with expansion modules with zero customization in MISP.
- A simple ReST API between the modules and MISP allowing auto-discovery of new modules with their features.
- Benefit from existing Python modules in Viper or any other tools.
- MISP modules functionality introduced in MISP 2.4.28.
- MISP import/export modules introduced in MISP 2.4.50.
MISP modules can be run on the same system or on a remote server.

Python 3 is required to run MISP modules.

- `sudo apt-get install python3-dev python3-pip libpq5`
- `cd /usr/local/src/`
- `sudo git clone https://github.com/MISP/misp-modules.git`
- `cd misp-modules`
- `sudo pip3 install -I -r REQUIREMENTS`
- `sudo pip3 install -I .`
- `sudo vi /etc/rc.local, add this line: ‘sudo -u www-data misp-modules -s &’`
http://127.0.0.1:6666/modules - introspection interface to get all modules available
  ▶ returns a JSON with a description of each module

http://127.0.0.1:6666/query - interface to query a specific module
  ▶ to send a JSON to query the module

MISP autodiscovers the available modules and the MISP site administrator can enable modules as they wish.

If a configuration is required for a module, **MISP adds automatically the option** in the server settings.
Finding available MISP modules

```bash
    curl -s http://127.0.0.1:6666/modules
```

```json
    {
      "type": "expansion",
      "name": "dns",
      "meta": {
        "module-type": ["expansion", "hover"],
        "description": "Simple DNS expansion service to resolve IP address from MISP attributes",
        "author": "Alexandre Dulaunoy",
        "version": "0.1"
      },
      "mispattributes": {
        "output": [
          "ip-src",
          "ip-dst"
        ],
        "input": [
          "hostname",
          "domain"
        ]
      }
    }
```
Querying a module

```
curl -s http://127.0.0.1:6666/query -H "Content-Type: application/json" -data @body.json -X POST
```

Body.json

```
1  {"module": "dns", "hostname": "www.circl.lu"}
```

and the response of the dns module:

```
1  {"results": [{"values": ["149.13.33.14"],
2  "types": ["ip-src", "ip-dst"]}]}
```
MISP modules - How it’s integrated in the UI?

Choose the enrichment module that you wish to use for the expansion

Enrichment Results

Below you can see the attributes that are to be created. Make sure that the categories and the types are correct; often several options will be offered based on an inconclusive automatic resolution.
## Server settings

<table>
<thead>
<tr>
<th>Priority</th>
<th>Setting</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>Plugin.Enrichment_services_enable</td>
<td>true</td>
<td>Enable/disable the enrichment service.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_services_url</td>
<td><a href="http://127.0.0.1">http://127.0.0.1</a></td>
<td>The url used to access the service.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_services_port</td>
<td>6666</td>
<td>The port used to access the service.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_cve_enabled</td>
<td>false</td>
<td>Enable or disable the cve module.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_dns_enabled</td>
<td>true</td>
<td>Enable or disable the dns module.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_sourcecache_enabled</td>
<td>false</td>
<td>Enable or disable the source cache module.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_sourcecache_archivelpath</td>
<td></td>
<td>Set this required module specified.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_passivetotal_enabled</td>
<td>true</td>
<td>Enable or disable the passivetotal module.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_passivetotal_username</td>
<td><a href="mailto:alexandre.dulaunoy@circl.lu">alexandre.dulaunoy@circl.lu</a></td>
<td>Set this required module specified.</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_passivetotal_password</td>
<td></td>
<td>Set this required module specified.</td>
</tr>
</tbody>
</table>
MISP MODULES - MAIN TYPES OF MODULES

- Expansion modules - enrich data that is in MISP
  - Hover type - showing the expanded values directly on the attributes
  - Expansion type - showing and adding the expanded values via a proposal form
- Import modules - import new data into MISP
- Export modules - export existing data from MISP
import json
import dns.resolver

misperrors = {'error': 'Error'}
mispattributestore = {'input': [], 'output': []}
moduleinfo = {'version': '', 'author': '',
              'description': '', 'module-type': []}

def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    r = {'results': [{ 'types': [], 'values': []} ]}
    return r

def introspection():
    return mispattributes

def version():
    return moduleinfo
misperrors = {'error': 'Error'}
mispattributes = {'input': ['hostname', 'domain'], 'output': ['ip-src', 'ip-dst']}
moduleinfo = {'version': '', 'author': '',
              'description': '', 'module-type': []}
misperrors = {'error': 'Error'}
mispattributes = {'input': ['hostname', 'domain'], 'output': ['ip-src', 'ip-dst']}
moduleinfo = {'version': '0.1', 'author': 'Alexandre Dulaunoy',
             'description': 'Simple DNS expansion service to resolve IP address from MISP attributes', 'module-type': ['expansion', 'hover']}
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    # MAGIC
    # MORE MAGIC
    r = {'results': [
        {'types': output_types, 'values': values},
        {'types': output_types2, 'values': values2}
    ]}
    return r
if request.get('hostname'):
    toquery = request['hostname']
elif request.get('domain'):
    toquery = request['domain']
else:
    return False
r = dns.resolver.Resolver()
r.timeout = 2
r.lifetime = 2
r.nameservers = ['8.8.8.8']
try:
    answer = r.query(toquery, 'A')
except dns.resolver.NXDOMAIN:
    miserrors['error'] = "NXDOMAIN"
    return miserrors
except dns.exception.Timeout:
    miserrors['error'] = "Timeout"
    return miserrors
except:
    miserrors['error'] = "DNS_resolving_error"
    return miserrors
r = {'results': [{'types': mispattributes['output'], 'values':[str(answer[o])]}]}
return r
import json
import dns.resolver
misperrors = {'error': 'Error'}
mispattributes = {'input': ['hostname', 'domain'], 'output': ['ip-src', 'ip-dst']}
moduleinfo = {'version': '0.1', 'author': 'Alexandre Dulaunoy',
'description': 'Simple DNS expansion service to resolve IP address from MISP attributes', 'module-type': ['expansion', 'hover']}
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    if request.get('hostname'):
        toquery = request['hostname']
    elif request.get('domain'):
        toquery = request['domain']
    else:
        return False
    r = dns.resolver.Resolver()
    r.timeout = 2
    r.lifetime = 2
    r.nameservers = ['8.8.8.8']
    try:
        answer = r.query(toquery, 'A')
    except dns.resolver.NXDOMAIN:
        misperrors['error'] = 'NXDOMAIN'
        return misperrors
    except dns.exception.Timeout:
        misperrors['error'] = 'Timeout'
        return misperrors
    except:
        misperrors['error'] = 'DNS_resolving_error'
    return misperrors
r = {'results': [{'types': mispattributes['output'], 'values':[str(answer[0])]}]}
return r

def introspection():
    return mispattributes

def version():
    return moduleinfo
Testing your module

- Copy your module dns.py in modules/expansion/
- Restart the server misp-modules.py

```
[adulau:~/git/misp-modules/bin] $ python3 misp-modules.py
2016-03-20 19:25:43,748 - misp-modules - INFO - MISP modules passivetotal imported
2016-03-20 19:25:43,787 - misp-modules - INFO - MISP modules sourcecache imported
2016-03-20 19:25:43,789 - misp-modules - INFO - MISP modules cve imported
2016-03-20 19:25:43,790 - misp-modules - INFO - MISP modules dns imported
2016-03-20 19:25:43,797 - misp-modules - INFO - MISP modules server started on TCP port 6666
```

- Check if your module is present in the introspection
- curl -s http://127.0.0.1:6666/modules
- If yes, test it directly with MISP or via curl
# Configuration at the top
moduleconfig = ['username', 'password']

# Code block in the handler
    if request.get('config'):
        if (request['config'].get('username') is None) or (request['config'].get('password') is None):
            miserrors['error'] = 'CIRCL Passive SSL authentication is missing'
        return miserrors

x = pypssl.PyPSSL(basic_auth=(request['config']['username'], request['config']['password']))
Default expansion module set

- asn history
- CIRCL Passive DNS
- CIRCL Passive SSL
- Country code lookup
- CVE information expansion
- DNS resolver
- DomainTools
- eupi (checking url in phishing database)
- IntelMQ (experimental)
- ipasn
- PassiveTotal - http://blog.passivetotal.org/misp-sharing-done-differently
- sourcecache
- Virustotal
- Whois
Import modules

Similar to expansion modules
Input is a file upload or a text paste
Output is a list of parsed attributes to be edited and verified by the user

Some examples
- Cuckoo JSON import
- email import
- OCR module
- Open IoC import
import json

misperrors = {'error': 'Error'}
userConfig = {
    'number1': {
        'type': 'Integer',
        'regex': '/^[0-4]$/',
        'errorMessage': 'Expected a number in range [0-4]',
        'message': 'Column_number_used_for_value'}
}

moduleconfig=[]

def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    request['data'] = base64.b64decode(request['data'])
    r = {'results': [{'categories': [], 'types': [], 'values':[]}]}
    return r

def introspection():
    return {'userConfig': userConfig, 'inputSource': inputSource, 'moduleConfig': moduleConfig}

def version():
    return moduleinfo
Creating your import module (userConfig and inputSource)

userConfig = {
    'number1': {
        'type': 'Integer',
        'regex': '/^\[0-4]$/i',
        'errorMessage': 'Expected a number in range [0-4]',
        'message': 'Column number used for value'
    }
};
inputSource = ['file', 'paste']
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    request['data'] = base64.b64decode(request['data'])
    r = {'results': [{'categories': [], 'types': [], 'values': []}]}
    return r
def introspection():
    modulesetup = {}
    try:
        userConfig
        modulesetup[’userConfig’] = userConfig
    except NameError:
        pass
    try:
        moduleConfig
        modulesetup[’moduleConfig’] = moduleConfig
    except NameError:
        pass
    try:
        inputSource
        modulesetup[’inputSource’] = inputSource
    except NameError:
        pass
    return modulesetup
**EXPORT MODULES**

- Input is currently only a single event
- Dynamic settings
- Later on to be expanded to event collections / attribute collections
- Output is a file in the export format served back to the user
- Export modules was recently introduced but a CEF export module already available
- Lots of ideas for upcoming modules and including interaction with misp-darwin
import json
inputSource = ['event']
outputFileExtension = 'txt'
responseType = 'application/text'
moduleinfo = {'version': '0.1', 'author': 'Andras Iklody',
              'description': 'Skeleton export module',
              'module-type': ['export']}

def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    # insert your magic here!
    output = my_magic(request["data"])
    r = {"data":base64.b64encode(output.encode('utf-8')).decode('utf-8')}
    return r

def introspection():
    return {'userConfig': userConfig, 'inputSource': inputSource, 'moduleConfig': moduleConfig, 'outputFileExtension': outputFileExtension}

def version():
    return moduleinfo
inputSource = ['event']
outputFileExtension = 'txt'
responseType = 'application/txt'
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    # insert your magic here!
    output = my_magic(request["data"])
    r = {"data":base64.b64encode(output.encode('utf-8')).decode('utf-8')}
    return r
def introspection():
    modulesetup = {}
    try:
        responseType
        modulesetup[‘responseType’] = responseType
    except NameError:
        pass
    try:
        userConfig
        modulesetup[‘userConfig’] = userConfig
    except NameError:
        pass
    try:
        moduleConfig
        modulesetup[‘moduleConfig’] = moduleConfig
    except NameError:
        pass
    try:
        outputFileExtension
        modulesetup[‘outputFileExtension’] = outputFileExtension
    except NameError:
        pass
    try:
        inputSource
        modulesetup[‘inputSource’] = inputSource
    except NameError:
Backward compatible - an additional field to extend the format

```python
misp_attributes = {'input': [...], 'output': [...], 'format': 'misp_standard'}
```

Takes a standard MISP attribute as input

Can return MISP attributes, objects & tags

Supports relationships

First modules supporting this new export format

- urlhaus expansion module
- Joe Sandbox import & query module
# New Expansion & Import Modules View (MISP 2.4.110)

**Enrichment Results**

Below you can see the attributes and objects that are to be created from the results of the enrichment modules.

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID</td>
<td></td>
<td>1229</td>
</tr>
<tr>
<td>Event UUd</td>
<td></td>
<td>5cc3042c-8b64-4b37-96d4-47ace96451a</td>
</tr>
<tr>
<td>Event creator org</td>
<td></td>
<td>ORGNAME</td>
</tr>
<tr>
<td>Event Info</td>
<td></td>
<td>urhaus test</td>
</tr>
<tr>
<td>iResolved Attributes</td>
<td></td>
<td>14 (2 Objects)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: viroscan-report[]</th>
<th>References: 0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>detection-ratio: text</td>
<td>19 / 68</td>
</tr>
<tr>
<td>External analysis</td>
<td>permalink: link</td>
<td><a href="https://www.virusotal.com/files/2d3a911b398b7c5d8b231bc6b07307bc7f8578eff752zed">https://www.virusotal.com/files/2d3a911b398b7c5d8b231bc6b07307bc7f8578eff752zed</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID: 12700</th>
<th>Name: file[]</th>
<th>References: 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload delivery</td>
<td>sha256: sha256</td>
<td>2bcb40b8-a2c4-bfe6-939e-3d64209b0f25b</td>
</tr>
<tr>
<td>Other</td>
<td>size-in-bytes: size-in-bytes</td>
<td>98380</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network activity url</th>
<th><a href="http://automotive/dreamteam.com/vexe">http://automotive/dreamteam.com/vexe</a></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Network activity url</td>
<td><a href="http://shoppingpopscoop.co.uk/vexe">http://shoppingpopscoop.co.uk/vexe</a></td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://popepe.scopemaster.fr/vexe">http://popepe.scopemaster.fr/vexe</a></td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://oxygenpopeopencounters.co.uk/vexe">http://oxygenpopeopencounters.co.uk/vexe</a></td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://laptops.popeoprop.net/vexe">http://laptops.popeoprop.net/vexe</a></td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://laptops.popeoprop">http://laptops.popeoprop</a> ethnic/vexe</td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://laptops.popeoprop.motech/vexe">http://laptops.popeoprop.motech/vexe</a></td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://laptops.popeoprop.info/vexe">http://laptops.popeoprop.info/vexe</a></td>
<td></td>
</tr>
<tr>
<td>Network activity url</td>
<td><a href="http://laptops.popeoprop.best/vexe">http://laptops.popeoprop.best/vexe</a></td>
<td></td>
</tr>
</tbody>
</table>
Upcoming additions to the module system - General

- Expose the modules to the APIs
- Move the modules to background processes with a messaging system
- Difficulty is dealing with uncertain results on import (without the user having final say)
MISP modules can be designed as standalone application.

- https://github.com/MISP/misp-modules
- https://github.com/MISP/

We welcome new modules and pull requests.
MISP Galaxy

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
MISP started out as a platform for technical indicator sharing

The need for a way to describe threat actors, tools and other commonalities became more and more pressing

**Taxonomies quickly became essential for classifying events**

The weakness of the tagging approach is that it’s not very descriptive

We needed a way to attach **more complex structures to data**

Also, with the different naming conventions for the same "thing" attribution was a mess

This is where the Galaxy concept came in
Pre-crafted galaxy "clusters" via GitHub project

Attach them to an event and attribute(s)

The main design principle was that these higher level informations are meant for human consumption

This means flexibility - key value pairs, describe them dynamically

Technical indicators remain strongly typed and validated, galaxies are loose key value lists
The galaxy object stack

- **Galaxy**: The type of data described (Threat actor, Tool, ...)
- **Cluster**: An individual instance of the galaxy (Sofacy, Turla, ...)
- **Element**: Key value pairs describing the cluster (Country: RU, Synonym: APT28, Fancy Bear)
- **Reference**: Referenced galaxy cluster (Such as a threat actor using a specific tool)
(Some) Existing Galaxies

- **Exploit-Kit:** An enumeration of known exploitation kits used by adversaries
- **Microsoft activity group:** Adversary groups as defined by Microsoft
- **Preventive measure:** Potential preventive measures against threats
- **Ransomware:** List of known ransomwares
- **TDS:** Traffic Direction System used by adversaries
- **Threat-Actor:** Known or estimated adversary groups
- **Tool:** Tools used by adversaries (from Malware to common tools)
- **MITRE ATT&CK:** Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK™)
## What a Cluster Looks Like

### Galaxies

**Threat Actor**

- **Sofacy**

**Description**
The Sofacy Group (also known as APT28, Pawn Storm, Fancy Bear, and Sedinta) is a cyber espionage group believed to have ties to the Russian government. Likely operating since 2007, the group is known to target government, military, and security organizations. It has been characterized as an advanced persistent threat.

**Synonyms**
- APT 28
- APT28
- Pawn Storm
- Fancy Bear
- Sedinta
- TsarTeam
- TG-4127
- Group-4127
- STRONTIUM
- Grey-Cloud

**Source**
- MISP Project

**Authors**
- Alexandre Dulaunoy
- Florian Roth
- Thomas Schreck
- Timo Steffens
- Various

**Country**
- RU

**Refs**
Internally simply using a taxonomy-like tag to attach them to events

Example: misp-galaxy:threat-actor="Sofacy"

Synchronisation works out of the box with older instances too. They will simply see the tags until they upgrade.

Currently, as mentioned we rely on the community’s contribution of galaxies
ATTACHING CLUSTERS

- Use a searchable synonym database to find what you’re after

![Synonym Database Example](image-url)
Creating your own galaxy

- Creating galaxy clusters has to be straightforward to get the community to contribute
- Building on the prior success of the taxonomies and warning lists
- Simple JSON format in similar fashion
- Just drop the JSON in the proper directory and let MISP ingest it
- We always look forward to contributions to our galaxies repository
If you want to create a completely new galaxy instead of enriching an existing one

```
{
    "name": "Threat Actor",
    "type": "threat-actor",
    "description": "Threat actors are characteristics of malicious actors (or adversaries) representing a cyber attack threat including presumed intent and historically observed behaviour.",
    "version": 1,
    "uuid": "698774c7-8022-42c4-917f-8d6e4f06ada3"
}
```
Clusters contain the meat of the data

Skeleton structure as follows

```json
{  
  "values": [  
    {  
      "meta": {},  
      "description": "",  
      "value": "",  
      "related_clusters": [{},],  
    }  
  ]
}
```
The Sofacy Group (also known as APT 28, Pawn Storm, Fancy Bear and Sednit) is a cyber espionage group believed to have ties to the Russian government. Likely operating since 2007, the group is known to target government, military, and security organizations. It has been characterized as an advanced persistent threat.

Or adding your own meta fields.
"description": "Putter Panda were the subject of an extensive report by CrowdStrike, which stated: 'The CrowdStrike Intelligence team has been tracking this particular unit since 2012, under the codename PUTTER PANDA, and has documented activity dating back to 2007. The report identifies Chen Ping, aka cpyy, and the primary location of Unit 61486.'",

"meta": {
  "cfr-suspected-state-sponsor": "China",
  "cfr-suspected-victims": [
    "U.S. satellite and aerospace sector"
  ],
  "cfr-target-category": [
    "Private sector",
    "Government"
  ],
  "cfr-type-of-incident": "Espionage",
  "country": "CN",
  "refs": [
    "http://cdn0.vox-cdn.com/assets/4589853/crowdstrike-intelligence-report-putter-panda.original.pdf",
    "https://www.cfr.org/interactive/cyber-operations/putter-panda"
  ]}

"description": "Universal Development and Security Guidelines as Applicable to Election Technology."
"icon": "map",
"kill_chain_order": {
  "example-of-threats": [
    "setup | party/candidate-registration",
    "setup | electoral-rolls",
    "campaign | campaign-IT",
    "all-phases | governement-IT",
    "voting | election-technology",
    "campaign/public-communication | media/press"
  ]
},
"name": "Election guidelines",
"namespace": "misp",
"type": "guidelines",
"uuid": "c1dc03b2-89b3-42a5-9d41-782ef726435a",
"version": 1
Cluster JSON matrix-like

```json
{
    "description": "DoS or overload of party/campaign registration, causing them to miss the deadline",
    "meta": {
        "date": "March 2018."
    },
    "kill_chain": [
        "example-of-threats:setup | party/candidate-registration"
    ],
    "refs": [
    ],
    "uuid": "154c6186-a007-4460-a029-ea23163448fe",
    "value": "DoS or overload of party/campaign registration, causing them to miss the deadline"
}
```
Cluster can be related to one or more clusters using default relationships from MISP objects and a list of tags to classify the relation.

```
    "related": [
        {
            "dest-uuid": "5ce5392a-3a6c-4e07-9df3-9b6a9159ac45",
            "tags": [
                "estimative-language:likelihood-probability="likely"
            ],
            "type": "similar"
        }
    ],
    "uuid": "Oca45163-e223-4167-b1af-f088ed14a93d",
    "value": "Putter Panda"
```
```python
from pymispgalaxies import Clusters

c = Clusters()
list(g.keys())
# ['threat–actor', 'ransomware', 'exploit–kit', 'tds', 'tool', 'rat', 'mitre–attack–pattern',
# 'mitre–intrusion–set', 'preventive–measure']

print(c.get("rat"))
# misp–galaxy: rat="Brat"
# misp–galaxy: rat="Loki RAT"
# misp–galaxy: rat="join.me"
# misp–galaxy: rat="Setro"
# misp–galaxy: rat="drat"
# misp–galaxy: rat="Plasma RAT"
# misp–galaxy: rat="NanoCore"
# misp–galaxy: rat="DarkTrack"
# misp–galaxy: rat="Theef"
# misp–galaxy: rat="Greame"
# misp–galaxy: rat="Nuclear RAT"
# misp–galaxy: rat="DameWare Mini Remote Control"
# misp–galaxy: rat="ProRat"
# misp–galaxy: rat="death"
# misp–galaxy: rat="Dark DDoSeR"
# ....

print(c.get("rat").description)
# remote administration tool or remote access tool (RAT), also called sometimes remote
# access trojan, is a piece of software or programming that allows a remote "operator"
# to control a system as if they have physical access to that system.
```
- info@circl.lu (if you want to join the CIRCL MISP sharing community)
- OpenPGP fingerprint: 3B12 DCC2 82FA 2931 2F5B 709A 09E2 CD49 44E6 CBCD
- https://github.com/MISP/-
  http://www.misp-project.org/
- We welcome any contributions to the project, be it pull requests, ideas, github issues,...
MISP Object Template
Building custom and open data models

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019 20190617
Attributes are a simple but powerful tool to describe data
Lacking the capability to create containers around attributes describing a common concept
The goal was to develop something semi-standardised, with the option to **dynamically build templates**
We have considered a list of different solutions such as simple boolean operators, but found that the current implementation was superior.
The result is a simple template that uses the basic attribute types as building blocks along with some meta data
The template does **not have to be known** in order to use the constructed objects
What we maintain now is a set of common objects, but similarly to our other JSON formats, users can extend it with their own ideas.
Using a similar JSON format as the taxonomies, galaxies, warninglists.

You can find the default set of object templates in the git repository¹.

Some of the object templates capture objects from other standards or mimic the output of tools.

We tried to capture the most common use-cases coming from our own use-case as well as those of various partners that got involved.

Improvements or pull requests for new object templates are of course always welcome.

¹https://www.github.com/MISP/misp-objects/
Existing Object examples

- **AIL-leak** - *AIL object, an example for an object catering to the output of another tool*
- **Android permission** - *An object used to further contextualise another object*
- **Bank account**
- **File** *Generic object to describe a file*
- **Passive DNS**
- **Regex**
- **Sandbox report**
- **Vulnerability** *Enabling new use-cases such as pre-sharing of vulnerability information*
- **x509**
- **Yara** *Verbatim sharing of rule sets along with meta-data*
Object Template skeleton

```json
{
    "requiredOneOf": [],
    "required": [],
    "attributes": {},
    "version": 1,
    "description": "My description",
    "meta-category": "Chosen meta category",
    "uuid": "Object template uuid",
    "name": "Object template name"
}
```
Adding elements to an object template

```json
1  "regexp-type": {
2      "description": "Type of the regular expression syntax.",
3      "disable_correlation": true,
4      "ui-priority": 0,
5      "misp-attribute": "text",
6      "values_list": [
7          "PCRE",
8          "PCRE2",
9          "POSIX BRE",
10         "POSIX ERE"
11      ]
12  },
```
ATTRIBUTE KEYS

- Primary key: Object relation
- description: A description of the attribute in relation to the object
- disable_correlation: You can disable correlations for attributes in the resulting object
- ui-priority: Not implemented yet, but the idea is to have a "quick view" of objects only showing certain priority levels
- misp-attribute: The misp attribute type used as the building block
- values_list: an optional list of values from which the user must choose instead of entering a value manually
- sane_defaults: an optional list of values from which the user may choose instead of entering a value
- multiple: Allow the user to add more than one of this attribute
The template also defines which of the added attributes are mandatory.

Requirements are pointed to via their object relations names.

We differentiate between two types of rule sets:

- Required: Everything in this list has to be set in order for the object to validate.
- Required One Of: Any of the attributes in this list will satisfy the requirements.
What will the template actually do?

- Templates create a form that can be used to populate an event.
- When using templates, MISP will enforce everything according to the template rules.
- However, these are only optional, users can avoid using the templates when creating events via the API.
- The reason for this is that you do not need to have the template in order to create an object.
- The limitation of this system: You **cannot modify** objects that were created with unknown templates.
## Templates as Rendered in the UI

### Add File Object

<table>
<thead>
<tr>
<th>Object Template</th>
<th>Description</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>File v10</td>
<td>File object describing a file with meta-information</td>
<td><strong>Required one of:</strong> filename, size-in-bytes, authentihash, ssdeep, imphash, pehash, md5, sha1, sha224, sha256, sha384, sha512, sha512/224, sha512/256, tlsh, pattern-in-file, x509-fingerprint-sha1, malware-sample</td>
</tr>
</tbody>
</table>

**Meta category:** File

**Distribution:**

- Inherit event

**Comment:**

### Save Name :: type

<table>
<thead>
<tr>
<th>Name :: type</th>
<th>Description</th>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Md5 :: md5</strong></td>
<td>[Insecure] MD5 hash (128 bits)</td>
<td>Payload delivery</td>
<td></td>
</tr>
<tr>
<td><strong>Pattern-in-file :: pattern-in-file</strong></td>
<td>Pattern that can be found in the file</td>
<td>Payload installation</td>
<td></td>
</tr>
<tr>
<td><strong>Sha256 :: sha256</strong></td>
<td>Secure Hash Algorithm 2 (256 bits)</td>
<td>Payload delivery</td>
<td></td>
</tr>
<tr>
<td><strong>Sha512 :: sha512</strong></td>
<td>Secure Hash Algorithm 2 (512 bits)</td>
<td>Payload delivery</td>
<td></td>
</tr>
<tr>
<td>Filename :: filename</td>
<td>Filename in</td>
<td>Payload delivery</td>
<td></td>
</tr>
</tbody>
</table>
### Templates as rendered in the UI

<table>
<thead>
<tr>
<th>Date</th>
<th>Category</th>
<th>Filename</th>
<th>File Size (bytes)</th>
<th>Entropy (bits)</th>
<th>MD5</th>
<th>SHA1</th>
<th>SHA256</th>
<th>SHA512</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>fileYTE</td>
<td>774200</td>
<td>6.72e5997226</td>
<td>bc123d3ebe9d10784d75ae5e4df34001</td>
<td>5ef9b1be6f892c546322d5b6b52a39e0a0e1ef</td>
<td>616e43f7504670134138705e0b2118ed727af8b77a5531aa593194485e</td>
<td>e171f94f5b3be30c2c66b53f2877042324c3d455c9f3f8b7f9f385332f6d1d3a3c2e1953273ebd8d836d48b89ba336d4f510625f970c1633ca76</td>
</tr>
</tbody>
</table>

**References:**
- File: fileYTE
- File Size: 774200 bytes
- Entropy: 6.72e5997226 bits
- MD5: bc123d3ebe9d10784d75ae5e4df34001
- SHA1: 5ef9b1be6f892c546322d5b6b52a39e0a0e1ef
- SHA256: 616e43f7504670134138705e0b2118ed727af8b77a5531aa593194485e
- SHA512: e171f94f5b3be30c2c66b53f2877042324c3d455c9f3f8b7f9f385332f6d1d3a3c2e1953273ebd8d836d48b89ba336d4f510625f970c1633ca76
Q&A

- https://github.com/MISP/MISP
- https://github.com/MISP/misp-objects
- info@circl.lu (if you want to join one of the MISP community operated by CIRCL)
- PGP key fingerprint: CA57 2205 CO02 4E06 BA70 BE89 EAAD CFFC 22BD 4CD5
MISP Dashboard
Real-time overview of threat intelligence from

Team CIRCL
info@circl.lu
June 18, 2019
MISP ZeroMQ

MISP includes a flexible publish-subscribe model to allow real-time integration of the MISP activities:

- Event publication
- Attribute creation or removal
- Sighting
- User login

→ Operates at global level in MISP
MISP ZeroMQ functionality can be used for various model of integration or to extend MISP functionalities:

- Real-time search of indicators into a SIEM\(^1\)
- Dashboard activities
- Logging mechanisms
- Continuous indexing
- Custom software or scripting

\(^1\)Security Information & Event Management
MISP-Dashboard: An introduction
MISP-DASHBOARD - REALTIME ACTIVITIES AND THREAT INTELLIGENCE
Subscribe to multiple ZMQ MISP instances
Provides historical geolocalised information
Present an experimental Gamification of the platform
Shows when and how MISP is used
Provides real time information showing current threats and activity
MISP-Dashboard: Architecture and development
Setting up the dashboard

1. Be sure to have a running redis server: e.g.
   - `redis-server -p 6250`
2. Update your configuration in `config.cfg`
3. Activate your virtualenv:
   - `./DASHENV/bin/activate`
4. Listen to the MISP feed by starting the zmq_subscriber:
   - `./zmq_subscriber.py`
5. Start the dispatcher to process received messages:
   - `./zmq_dispatcher.py`
6. Start the Flask server:
   - `./server.py`
7. Access the interface at `http://localhost:8001/`
MISP-Dashboard architecture
# Register your handler

dico_action = {
    "misp_json": handler_dispatcher,
    "misp_json_event": handler_event,
    "misp_json_self": handler_keepalive,
    "misp_json_attribute": handler_attribute,
    "misp_json_object": handler_object,
    "misp_json_sighting": YOUR_CUSTOM_SIGHTINGS_HANDLER,
    "misp_json_organisation": handler_log,
    "misp_json_user": handler_user,
    "misp_json_conversation": handler_conversation,
    "misp_json_object_reference": handler_log,
}
# Implement your handler

# e.g. user handler
def handler_user(zmq_name, jsondata):
    # json action performed by the user
    action = jsondata['action']
    # user json data
    json_user = jsondata['User']
    # organisation json data
    json_org = jsondata['Organisation']
    # organisation name
    org = json_org['name']
    # only consider user login
    if action == 'login':
        timestamp = time.time()
        # users_helper is a class to interact with the DB
        users_helper.add_user_login(timestamp, org)
Optimizing contribution scoring and model to encourage sharing and contributions enrichment

Increasing geolocation coverage

Global filtering capabilities
- Geolocation: Showing wanted attribute or only on specific region
- Trendings: Showing only specified taxonomies

Tighter integration with MISP
- Present in MISP by default
- Authenticated / ACL enabled version
CONCLUSION

MISP-Dashboard can provide realtime information to support security teams, CSIRTs or SOC showing current threats and activity by providing:

- Historical geolocalised information
- Geospatial information from specific regions
- The most active events, categories, tags, attributes, ...

It also propose a prototype of gamification of the platform providing incentive to share and contribute to the community.
Contributing to the MISP Project

Become part of the community to design, develop and improve information sharing

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
The MISP project has a Contributor Covenant Code of Conduct\(^1\).

The goal of the code of conduct is to foster an open, fun and welcoming environment.

Another important aspect of the MISP projects is to welcome different areas of expertise in information sharing and analysis. The diversity of the MISP community is important to make the project useful for everyone.

\(^1\)https://github.com/MISP/MISP/code_of_conduct.md
The most common way to contribute to the MISP project is to report a bug, issues or suggesting features.

Each project (MISP core, misp-modules, misp-book, misp-taxonomies, misp-galaxy, misp-object or PyMISP) has their own issue management.

Don’t forget that you can cross-reference issues from other sub-projects.

If you know an answer or could help on a specific issue, we welcome all contributions including useful comments to reach a resolution.
If you find security vulnerabilities (even minor ones) in MISP project, send an encrypted email (info@circl.lu) with the details and especially how to reproduce the issues. Avoid to share publicly the vulnerability before a fix is available in MISP. PGP key fingerprint: CA77 2205 C002 4E06 BA70 BE89 EAAD CFFC 22BD 4CD5.

We usually fix reported and confirmed security vulnerabilities in less than 48 hours.

We will request a CVE number if the reporters didn’t ask for one (don’t forget to mention how you want to be credited).
The majority of the repositories within the MISP GitHub organisation includes automatic integration with TravisCI.

If you contribute and make a pull-request, **verify if your changes affect the result of the tests.**

Automatic integration is not perfect including Travis but it’s a quick win to catch new bugs or major issues in contribution.

When you do a pull-request, TravisCI is automatically called².

▶ If this fails, no worries, **review the output at Travis** (it’s not always you).

We are working on additional automatic tests including unit testing for the MISP core software (contributors are welcome).

²https://travis-ci.org/MISP
All JSON format (galaxy, taxonomies, objects or warning-lists) are described in a JSON Schema\(^3\).

The TravisCI tests are including JSON validation (via \texttt{jq}) and validated with the associated JSON schema.

How to contribute a JSON library (objects, taxonomies, galaxy or warning-list):

▶ If you update a JSON library, don’t forget to run \texttt{jq\_all\_the\_things.sh}. It’s fast and easy. If it fails, review your JSON.
▶ Commit your code and make a pull-request.

Documentations (in PDF and HTML format) for the libraries are automatically generated from the JSON via asciidoctor\(^4\).

\(^3\)\texttt{schema\_name.json}
\(^4\)\texttt{example https://github.com/MISP/misp-galaxy/blob/master/tools/adoc\_galaxy.py}
In addition to the automatic generation of documentations from JSON files, we maintain **misp-book** which is a generic documentation for MISP including usage, API documentation, best practices and specific configuration settings.  

The book is generated in HTML, PDF, epub and mobi using GitBook which is a framework to write documentation in MarkDown format.  

TravisCI is included in misp-book and **the book generation is tested at each commit**.  

The MISP book is regularly published on misp-project.org and circl.lu website.  

Contributors are welcome especially for new topics and also fixing our broken English.

---

[6] [https://github.com/GitbookIO](https://github.com/GitbookIO)  
[7] Topics of interest are analysts best-practices,
If you want to contribute to our IETF Internet-Draft for the MISP standard, misp-rfc is the repository where to contribute.

Update only the markdown file, the XML and ASCII for the IETF I-D are automatically generated.

If a major release or updates happen in the format, we will publish the I-D to the IETF.

The process is always MISP implementation → IETF I-D updates.

---

8https://github.com/MISP/misp-rfc
9https://datatracker.ietf.org/doc/search/?name=misp&activedrafts=on&rfcs=on
MISP core development crash course
How I learned to stop worrying and love the PHP

Team CIRCL

MISP Training @ FIRST.org 2019 20190617
MISP is based on PHP 5.6+

Using the MVC framework CakePHP 2.x

What we’ll look at now will be a quick glance at the structuring / layout of the code
MVC Frameworks in General

- separation of business logic and views, interconnected by controllers
- main advantage is clear separation of the various components
- lean controllers, fat models (kinda...)
- domain based code reuse
- No interaction between Model and Views, ever
Structure of MISP Core app directories

- Config: general configuration files
- Console: command line tools
- Controller: Code dealing with requests/responses, generating data for views based on interactions with the models
- Lib: Generic reusable code / libraries
- Model: Business logic, data gathering and modification
- Plugin: Alternative location for plugin specific codes, ordered into controller, model, view files
- View: UI views, populated by the controller
Each public function in a controller is exposed as an API action
request routing (admin routing)
multi-use functions (POST/GET)
request/response objects
contains the action code, telling the application what data fetching/modifying calls to make, preparing the resulting data for the resulting view
grouped into controller files based on model actions
Accessed via UI, API, AJAX calls directly by users
For code reuse: behaviours
Each controller bound to a model
Controllers - functionalities of controllers

- pagination functionality
- logging functionality
- Controllers actions can access functionality / variables of Models
- Controllers cannot access code of other controller actions (kind of...)
- Access to the authenticated user’s data
- beforeFilter(), afterFilter() methods
- Inherited code in AppController
Components = reusable code for Controllers
- Authentication components
- RestResponse component
- ACL component
- Cidr component
- IOCImport component (should be moved)
Controllers - additional functionalities

- code handling API requests
- auth/session management
- ACL management
- API management
- Security component
- Important: quertString/PyMISP versions, MISP version handler
- Future improvements to the export mechanisms
Controls anything that has to do with:
- finding subsets of data
- altering existing data
- inherited model: AppModel
- reusable code for models: Behaviours
- regex, trim
Models - hooking system

- Versatile hooking system
  - manipulate the data at certain stages of execution
  - code can be located in 3 places: Model hook, AppModel hook, behaviour
Hooks / model pipeline for data creation / edits

- `beforeValidate()` (lowercase all hashes)
- `validate()` (check hash format)
- `afterValidate()` (we never use it)
- `beforeSave()` (purge existing correlations for an attribute)
- `afterSave()` (create new correlations for an attribute / zmq)
Hooks for deletions
  ► beforeDelete() (purge correlations for an attribute)
  ► afterDelete() (zmq)

Hooks for retrieving data
  ► beforeFind() (modify the find parameters before execution, we don’t use it)
  ► afterFind() (json decode json fields)
- code to handle version upgrades contained in AppModel
- generic cleanup/data migration tools
- centralised redis/pubsub handlers
- (Show example of adding an attribute with trace)
Views - Scope and Structure

- Templates for views
- Layouts
- Reusable template code: elements
  - Attribute list, rows (if reused)
- Reusable code: helpers
  - Commandhelper (for discussion boards), highlighter for searches, tag colour helper
- Views per controller
Views - Types of views and helpers

- ajax views vs normal views
- data views vs normal views vs serialisation in the controller
- sanitisation h()
- creating forms
  - sanitisation
  - CSRF
algorithm for checking if a user has access to an attribute
creator vs owner organisation
distribution levels and inheritance (events -> objects -> attributes)
shorthand inherit level
sharing groups (org list, instance list)
correlation distribution
algorithms for safe data fetching (fetchEvents(), fetchAttributes(),...)

Testing your code

- Functional testing
- Impact scope
  - View code changes: only impacts request type based views
  - Controller code changes: Should only affect given action
  - Model code changes: can have impact on entire application
  - Lib changes: can have affect on the entire application
- Don’t forget: queryACL, change querystring
Deep-dive into PyMISP

MISP - Threat Sharing

Team CIRCL

http://www.misp-project.org/
Twitter: @MISPProject

MISP Training @ FIRST.org 2019
20190617
MISP is a large project
Your production environment is even more complex
3rd party services are even worse
Querying MISP via CURL is doable, but get’s painful fast
Talking to MySQL directly can be dangerous
POST a JSON blob, receive a JSON blob. You can do it manually(-ish)
Core goal: providing stable access to APIs, respect access control

- Simplifying handling & automation of indicators in 3rd party tools
- Hiding complexity of the JSON blobs
- Providing pre-cooked examples for commonly used operations
- Helping integration with existing infrastructure
There are 4 main cases here:

- **Metadata of the events that have been modified**
  - `search_index` → timestamp (1h, 1d, 7d, ...), returns list of all the modified events

- **Full events (metadata + attributes)**
  - `search` → timestamp (1h, 1d, 7d, ...)

- **Modified attributes**
  - `search` → controller = attributes and timestamp (1h, 1d, 7d, ...)

- **Other use case: get last published events** by using the last parameter in the `search` method.
There are 3 main cases here:

- Easy, but slow: full text search with `search_all`
- Faster: use the `search` method and search by tag, type, enforce the warning lists, with(-out) attachments, dates interval, ...
- Get malware samples (if available on the instance).
There are 3 main cases here:

- Add Event, edit its metadata
- Add attributes or objects to event
- (un)Tag event or attribute (soon object)
- Edit Attributes medatada
- Upload malware sample (and automatically expand it)
Administrative tasks

Assuming you have the right to do it on the instance.

- Managing users
- Managing organisations
- Managing sync servers
Other Capabilities

- Upload/download samples
- **Proposals**: add, edit, accept, discard
- **Sightings**: Get, set, update
- Export **statistics**
- Manage **feeds**
- Get MISP server version, recommended PyMISP version
- And more, look at the api file
from pymisp import MISPEvent, EncodeUpdate

# Create a new event with default values
event = MISPEvent()

# Load an existing JSON dump (optional)
event.load_file('Path/to/event.json')
event.info = 'My cool event'  # Duh.

# Add an attribute of type ip-dst
event.add_attribute('ip-dst', '8.8.8.8')

# Mark an attribute as deleted (From 2.4.60)
event.delete_attribute('<Attribute_UUID>')

# Dump as json
event_as_jsondump = json.dumps(event, cls=EncodeUpdate)
Basics

- Python 3.5+ is recommended
- PyMISP is always inline with current version (pip3 install pymisp)
- Dev version: pip3 install git+https://github.com/MISP/PyMISP.git
- Get your auth key from: https://misppriv.circl.lu/events/automation
  - Not available: you don’t have "Auth key access" role. Contact your instance admin.
- Source available here: git clone https://github.com/MISP/PyMISP.git
**Examples**

- **PyMISP needs to be installed (duh)**
- **Usage:**
  - Create examples/keys.py with the following content
    ```python
    misp_url = "https://url-to-your-misp"
misp_key = "<API_KEY>"
misp_verifycert = True
    ```
- **Proxy support:**
  ```python
  proxies = {
    'http': 'http://127.0.0.1:8123',
    'https': 'http://127.0.0.1:8123',
  }
  PyMISP(misp_url, misp_key, misp_verifycert, proxies=proxies)
  ```
Examples

- Lots of ideas on how to use the API
- You may also want to look at the tests directory
- All the examples use argparse. Help usage is available: `script.py -h`
  - `add_file_object.py`: Attach a file (PE/ELF/Mach-O) object to an event
  - `upload.py`: Upload a malware sample (use advanced expansion is available on the server)
  - `last.py`: Returns all the most recent events (on a timeframe)
  - `add_named_attribute.py`: Add attribute to an event
  - `sighting.py`: Update sightings on an attribute
  - `stats.py`: Returns the stats of a MISP instance
  - `{add,edit,create}_user.py`: Add, Edit, Create a user on MISP
Basic example

```python
from pymisp import PyMISP
api = PyMISP(url, apikey, verifycert=True, debug=False, proxies=None)
response = api.<function>
if response['error']:
    # <something went wrong>
else:
    # <do something with the output>
```
Concept behind AbstractMISP

- JSON blobs are python dictionaries
- ... Accessing content can be a pain
- AbstractMISP inherits collections.MutableMapping, they are all dictionaries!
- ... Has helpers to load, dump, and edit JSON blobs
- **Important**: All the public attributes (not starting with a _) defined in a class are dumped to JSON
- **Tags**: Events and Attributes have tags, soon Objects. Tag handling is defined in this class.
- **edited**: When pushing a full MISPEvent, only the objects without a timestamp, or with a newer timestamp will be updated. This method recursively finds updated events, and removes the timestamp key from the object.
Pythonic representation of MISP elements

Easy manipulation

- Load an existing event
- Update the metadata, add attributes, objects, tags, mark an attribute as deleted, ...
- Set relations between objects
- Load and add attachments or malware samples as pseudo files

Dump to JSON
**MISPEvent - Main entrypoints**

- `load_file(event_path)`
- `load(json_event)`
- `add_attribute(type, value, **kwargs)`
- `add_object(obj=None, **kwargs)`
- `add_attribute_tag(tag, attribute_identifier)`
- `get_attribute_tag(attribute_identifier)`
- `add_tag(tag=None, **kwargs)`
- `objects[], attributes[], tags[]`
- `edited, all other parameters of the MISPEvent element (info, date, ...)`
- `to_json()`
MISPObject - Main entrypoints

- add_attribute(object_relation, **value)
- add_reference(referenced_uuid, relationship_type, comment=None, **kwargs)
- has_attributes_by_relation(list_of_relations)
- get_attributes_by_relation(object_relation)
- attributes[], relations[]
- edited, all other parameters of the MISPObject element (name, comment, ...)
- to_json()

- Can be validated against their template
- Can have default parameters applied to all attributes (i.e. distribution, category, ...)
MISPAttribute - Main entrypoints

- add_tag(tag=None, **kwargs)
- delete()
- malware_binary (if relevant)
- tags[]
- edited, all other parameters of the MISPObject element (value, comment, ...)
- to_json()
Libraries requiring specific 3rd party dependencies
Callable via PyMISP for specific usecases
Currently implemented:
- OpenIOC to MISP Event
- MISP to Neo4J
File - PE/ELF/MachO - Sections
VirusTotal
Generic object generator
- `debug=True` passed to the constructor enable debug to stdout
- Configurable using the standard logging module
- Show everything send to the server and received by the client

```python
import pymisp
import logging

logger = logging.getLogger('pymisp')
logger.setLevel(logging.DEBUG)  # enable debug to stdout

logging.basicConfig(level=logging.DEBUG,  # Enable debug to file
                    filename="debug.log",
                    filemode='w',
                    format=pymisp.FORMAT)
```
https://github.com/MISP/PyMISP
https://github.com/MISP/
https://pymisp.readthedocs.io/
We welcome new functionalities and pull requests.
MISP feeds - A simple and secure approach to generate, select and collect intelligence
Providing ready-to-use threat intelligence in MISP

Team CIRCL
TLP:WHITE

http://www.misp-project.org/
Twitter: @MISPPProject

MISP Training @ FIRST.org 2019
20190617
MISP Feeds provide a way to

- **Exchange information via any transports** (e.g. HTTP, TLS, USB keys)
- Preview events along with their attributes, objects
- Select and import events
- **Correlate attributes using caching**

MISP Feeds have the following advantages

- Feeds work without the need of MISP synchronisation (reducing attack surface and complexity to a static directory with the events)
- **Feeds can be produced without a MISP instance** (e.g. security devices, honeypot sensors)
By default, MISP is bundled with ~50 default feeds (MISP feeds, CSV or freetext feeds) which are not enabled by default and described in a simple JSON file\(^1\).

The feeds include CIRCL OSINT feed but also feeds like abuse.ch, Tor exit nodes or many more \(^2\).

\(^1\)https://github.com/MISP/MISP/blob/2.4/app/files/feed-metadata/defaults.json
\(^2\)http://www.misp-project.org/feeds/
Feed - Operations

- Cache feed attributes for correlation (not imported but visible in MISP)
- Disable feed
- Explore remote events
- Fetch all events (imported in MISP as event)
- Edit the feed configuration (e.g. authentication, URL,...)
- Remove feed
- Download feed metadata (to share feed details)
feed generator fetches events (matching some filtering) from a MISP instance and construct the manifest (defined in MISP core format) needed to export data.

Particularly,

- Used to generate the CIRCL OSINT feed
- Export events as json based on tags, organisation, events, ...
- Automatically update the dumps and the metadata file
- Comparable to a lightweight TAXII interface
url = 'your/misp/url'
key = 'YourAPIKey'
ssl = True
outputdir = 'output_directory'

filters = {
    'tag':'tlp:white|feed-export|!privint',
    'org':'CIRCL'
}

# the above would generate a feed for all events created by CIRCL, tagged tlp:white and/or feed-export but exclude anything tagged privint

valid_attribute_distribution_levels = ['0', '1', '2', '3', '4', '5']

# 0: Your Organisation Only
# 4: Sharing Group
# 5: Inherit Event
The PyMISP feed generator is great but may be inadequate or inefficient:
- Batch import of attributes/objects
- Data producer doesn’t have a MISP instance at hand and only wants to **produce a directly consumable feed**:
Real-time Feed generator - Usage

- generator.py exposes a class allowing to generate a MISP feed in real-time
- Each items can be appended on daily generated events

Example:

```python
# Init generator
generator = FeedGenerator()

# Adding an attribute to the daily event
attr_type = "ip-src"
attr_value = "8.8.8.8"
additional_data = {}
generator.add_attribute_to_event(attr_type,
                                  attr_value,
                                  **additional_data)
```
# Adding a MISP object (cowrie) to the daily event

```python
obj_name = "cowrie"
obj_data = {
    "session": "session_id",
    "username": "admin",
    "password": "admin",
    "protocol": "telnet"
}
generator.add_object_to_event(obj_name, **obj_data)
```
Adding custom feed to MISP

- Enabled
- Lookup visible
- Name
- Provider
- Source Format
- URL
- Source Format
- Headers
- Distribution
- Default Tag
- Filter rules
We welcome new functionalities and pull requests.
MISP workshop

Introduction into Information Sharing using MISP

Team CIRCL
TLP:WHITE

MISP Training @ FIRST.org 2019
20190617
Plan for this session

- Explanation of the CSIRT use case for information sharing and what CIRCL does
- Building an information sharing community and best practices

---

¹We published the complete guidelines in https://www.x-isac.org/assets/images/guidelines_to_set-up_an_ISAC.pdf
As a CSIRT, CIRCL operates a wide range of communities. We use it as an *internal tool* to cover various day-to-day activities. Whilst being the main driving force behind the development, we’re also one of the largest consumers. Different communities have different needs and restrictions.
Communities operated by CIRCL

- **Private sector community**
  - Our largest sharing community
  - Over *900 organisations*
  - *2000 users*
  - Functions as a central hub for a lot of sharing communities
  - Private organisations, Researchers, Various SoCs, some CSIRTs, etc

- **CSIRT community**
  - Tighter community
  - National CSIRTs, connections to international organisations, etc
Communities operated by CIRCL

- Financial sector community
  - Banks, payment processors, etc.
  - Sharing of **mule accounts** and **non-cyber threat infomartion**

- X-ISAC
  - **Bridging the gap** between the various sectorial and geographical ISACs
  - New, but ambitious initiative
  - Goal is to **bootstrap the cross-sectorial sharing** along with building the infrastructure to enable sharing when needed
Coming up - the ATT&CK EU community
- Work on attacker modelling
- With the assistance of Mitre themselves
- Unique opportunity to standardise on TTPs
- Looking for organisations that want to get involved!
Communities supported by CIRCL

- FIRST.org’s MISP community
- Telecom and Mobile operators’ community
- Various ad-hoc communities for exercises for example
  - Most recently for example for the ENISA exercise a few weeks ago
Sharing Scenarios in MISP

Sharing can happen for **many different reasons**. Let’s see what we believe are the typical CSIRT scenarios.

We can generally split these activities into 4 main groups when we’re talking about traditional CSIRT tasks:

- Core services
- Proactive services
- Advanced services
- Sharing communities managed by CSIRTs for various tasks
CSIRT Core Services

- **Incident response**
  - Internal storage of incident response data
  - Sharing of indicators derived from incident response
  - Correlating data derived and using the built in analysis tools
  - Enrichment services
  - Collaboration with affected parties via MISP during IR
  - Co-ordination and collaboration
  - Takedown requests

- Alerting of information leaks (integration with AIL²)

²https://github.com/CIRCL/AIL-framework
CSIRT proactive services

- **Contextualising** both internal and external data
- **Collection** and **dissimination** of data from various sources (including OSINT)
- Storing, correlating and sharing own manual research (reversing, behavioural analysis)
- Aggregating automated collection (**sandboxing, honeypots, spamtraps, sensors**)
  - MISP allows for the creation of **internal MISP "clouds"**
  - Store **large specialised datasets** (for example honeypot data)
  - MISP has **interactions with** a large set of such **tools** (Cuckoo, Mail2MISP, etc)

- **Situational awareness** tools to monitor trends and adversary TTPs within my sector/geographical region (MISP-dashboard, built in statistics)
CSIRT PROACTIVE SERVICES - MISP DASHBOARD
CSIRT advanced services

- Supporting **forensic analysts**
- Collaboration with **law enforcement**
- **Vulnerability** information sharing
  - **Notifications** to the constituency about relevant vulnerabilities
  - **Co-ordinating** with vendors for notifications (*)
  - Internal / closed community sharing of pentest results
  - We’re planning on starting a series of hackathons to find
CSIRTs’ management of sharing communities for constituent actions:

- **Reporting** non-identifying information about incidents (such as outlined in NISD)
- **Seeking** and engaging in **collaboration** with CSIRT or other parties during an incident
- Pre-sharing information to **request for help** / additional information from the community
- **Pseudo-anonymised sharing** through 3rd parties to **avoid attribution** of a potential target
- Building processes for **other types of sharing** to get the community engaged and acquainted with the methodologies of sharing (mule account information, border control, etc)
**A Quick Note on Compliance...**

- Collaboration with Deloitte as part of a CEF project for creating compliance documents
  - Information sharing and cooperation **enabled by GDPR**
  - How MISP enables stakeholders identified by the **NISD** to perform key activities
  - **AIL** and MISP

- For more information: [https://github.com/CIRCl/compliance](https://github.com/CIRCl/compliance)
We generally all end up sharing with peers that face similar threats.

Division is either sectorial or geographical.

So why even bother with trying to bridge these communities?
Advantages of cross sectorial sharing

- **Reuse of TTPs** across sectors
- Being hit by something that another sector has faced before
- **Hybrid threats** - how seemingly unrelated things may be interesting to correlate
- Prepare other communities for the capability and **culture of sharing** for when the need arises for them to reach out to CSIRT
- Generally our field is ahead of several other sectors when it comes to information sharing, might as well **spread the love**

**Sharing is caring!**
Starting a sharing community is **both easy and difficult** at the same time.

Many moving parts and most importantly, you’ll be dealing with a diverse group of people.

Understanding and working with your constituents to help them face their challenges is key.
Getting started with building your own sharing community

When you are starting out - you are in a unique position to drive the community and set best practices...

WITH GREAT POWER COMES GREAT RESPONSIBILITY.
RUNNING A SHARING COMMUNITY USING MISP - HOW TO GET GOING?

- Different models for constituents
  - Connecting to a MISP instance hosted by a CSIRT
  - Hosting their own instance and connecting to CSIRT’s MISP
  - Becoming member of a sectorial MISP community that is connected to CSIRT’s community

- Planning ahead for future growth
  - Estimating requirements
  - Deciding early on common vocabularies
  - Offering services through MISP
Rely on our instincts to imitate over expecting adherence to rules

- Lead by example - the power of imitation
- Encourage improving by doing instead of blocking sharing with unrealistic quality controls
  - What should the information look like?
  - How should it be contextualise
  - What do you consider as useful information?
  - What tools did you use to get your conclusions?
- Side effect is that you will end up raising the capabilities of your constituents
What counts as valuable data?

Sharing comes in many shapes and sizes
- Sharing results / reports is the classical example
- Sharing enhancements to existing data
- Validating data / flagging false positives
- Asking for support from the community

Embrace all of them. Even the ones that don’t do either, you’ll never know when they change their minds...
From our own communities, only about 30% of the organisations actively share data.

We have come across some communities with sharing requirements.

In our experience, this sets you up for failure because:

- Organisations will lose protection who would possibly benefit the most from it.
- Organisations that want to stay above the thresholds will start sharing junk / fake data.
- You lose organisations that might turn into valuable contributors in the future.
So how does one convert the passive organisations into actively sharing ones?

- Rely on organic growth
- Help them increase their capabilities
- As mentioned before, lead by example
- Rely on the inherent value to one’s self when sharing information (validation, enrichments, correlations)
- Give credit where credit is due, never steal the accolades of your community (that is incredibly demotivating)
Dispelling the myths around blockers when it comes to information sharing

- Sharing difficulties are not really technical issues but often it’s a matter of social interactions (e.g. trust).
  - You can play a role here: organise regular workshops, conferences, have face to face meetings

- Legal restrictions
  - "Our legal framework doesn’t allow us to share information."
  - "Risk of information leak is too high and it’s too risky for our organization or partners."

- Practical restrictions
  - "We don’t have information to share."
  - "We don’t have time to process or contribute indicators."
  - "Our model of classification doesn’t fit your model."
  - "Tools for sharing information are tied to a specific format, we use a different one."
Sharing technical information is a great start
However, to truly create valuable information for your community, always consider the context:
▶ Your IDS might not care why it should alert on a rule
▶ But your analysts will be interested in the threat landscape and the "big picture"

Classify data to make sure your partners understand why it is important for them

Massively important once an organisation has the maturity to filter the most critical subsets of information for their own defense
MISP has a versatile system (taxonomies) for classifying and marking data.

However, this includes different vocabularies with obvious overlaps.

MISP allows you to pick and choose vocabularies to use and enforce in a community.

Good idea to start with this process early.

If you don’t find what you’re looking for:

▶ Create your own (JSON format, no coding skills required)
▶ If it makes sense, share it with us via a pull request for redistribution
The MISPProject in co-operation with partners provides a curated list of galaxy information. Can include information packages of different types, for example:

- Threat actor information
- Specialised information such as Ransomware, Exploit kits, etc
- Methodology information such as preventative actions
- Classification systems for methodologies used by adversaries - ATT&CK

Consider improving the default libraries or contributing your own (simple JSON format)

If there is something you cannot share, run your own galaxies and share it out of bound with partners

Pull requests are always welcome
You might often fall into the trap of discarding seemingly "junk" data.

Besides volume limitations (which are absolutely valid, fear of false-positives is the most common reason why people discard data) - Our recommendation:

- Be lenient when considering what to keep
- Be strict when you are feeding tools

MISP allows you to filter out the relevant data on demand when feeding protective tools.

What may seem like junk to you may be absolutely critical to other users.
Sharing indicators for a detection matter.
▶ ’Do I have infected systems in my infrastructure or the ones I operate?’

Sharing indicators to block.
▶ ’I use these attributes to block, sinkhole or divert traffic.’

Sharing indicators to perform intelligence.
▶ ’Gathering information about campaigns and attacks. Are they related? Who is targeting me? Who are the adversaries?’

→ These objectives can be conflicting (e.g. False-positives have different impacts)
Analysts will often be interested in the modus operandi of threat actors over long periods of time. Even cleaned up infected hosts might become interesting again (embedded in code, recurring reuse). Use the tools provided to eliminate obvious false positives instead and limit your data-set to the most relevant sets.
Managing sub-communities

- Often within a community smaller bubbles of information sharing will form
- For example: Within a national private sector sharing community, specific community for financial institutions
- Sharing groups serve this purpose mainly
- As a CSIRT running a national community, consider bootstrapping these sharing communities
- Organisations can of course self-organise, but you are the ones with the know-how to get them started
Consider compartmentalisation - does it make sense to move a secret squirrel club to their own sharing hub to avoid accidental leaks?

Use your best judgement to decide which communities should be separated from one another

Create sharing hubs with manual data transfer

Some organisations will even have their data air-gapped - Feed system

Create guidance on what should be shared outside of their bubbles - organisations often lack the insight / experience to decide how to get going. Take the initiative!
Getting started with building a new community can be daunting. Feel free to get in touch with us if you have any questions!

- Contact: info@circl.lu
- https://www.circl.lu/
- https://github.com/MISP
- https://gitter.im/MISP/MISP
- https://twitter.com/MISPPProject
Various users and organisations can share data via MISP, multiple parties can be involved
▶ Trust, data quality and time-to-live issues
▶ Each user/organisation has different use-cases and interests

Attributes can be shared in large quantities (more than 1.3 million on MISPPRIV)
▶ Partial info about their validity (sightings)
▶ Partial info about their freshness (last update)
▶ Various conflicting interests such as operational security, attribution, source reliability evaluation...
Sightings add temporal context to indicators. A user, script or an IDS can extend the information related to indicators by reporting back to MISP that an indicator has been seen, or that an indicator can be considered as a false-positive.

- Sightings give more credibility/visibility to indicators.
- This information can be used to **prioritise and decay indicators**.
MISP is a peer-to-peer system, information passes through multiple instances.

- Producers can add context (such as tags from taxonomies, galaxies) about their asserted confidence or the reliability of the data
- Consumers can have different levels of trust in the producers and/or analysts themselves

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely reliable</td>
<td>100</td>
<td>Confirmed by other sources</td>
<td>100</td>
</tr>
<tr>
<td>Usually reliable</td>
<td>75</td>
<td>Probably true</td>
<td>75</td>
</tr>
<tr>
<td>Fairly reliable</td>
<td>50</td>
<td>Possibly true</td>
<td>50</td>
</tr>
<tr>
<td>Not usually reliable</td>
<td>25</td>
<td>Doubtful</td>
<td>25</td>
</tr>
<tr>
<td>Unreliable</td>
<td>0</td>
<td>Improbable</td>
<td>0</td>
</tr>
<tr>
<td>Reliability cannot be judged</td>
<td>50</td>
<td>Truth cannot be judged</td>
<td>50</td>
</tr>
<tr>
<td>Deliberately deceptive</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scoring Indicators 1/2

When scoring indicators\(^1\), multiple parameters\(^2\) can be taken into account. The **base score** is calculated with the following in mind:

- The reliability in the producer
- The trust in the data as signaled by the producer

\[
\text{base\_score} = \text{weight}_{tg} \cdot \text{tags} + \omega_{sc} \cdot \text{source\_confidence}
\]

\(^1\)Paper available: https://arxiv.org/pdf/1803.11052
\(^2\)at a variable extent as required
The weighted score is calculated using:

- The lifetime of the indicator (e.g. IP address vs hash value of a file)
  - The lifespan of the indicator (short for an IP - long for an hash): $\tau$
  - The decay rate $\delta$ → Speed at which an attribute loses value:
  - Weighthed score is reset to its base score as new sightings are received

$$score = base\_score \cdot \left(1 - \left(\frac{t}{\tau_a}\right)\frac{1}{\delta_a}\right)$$
Setting thresholds and retrieving the information should be simple and straightforward for the user:

- Automatic scoring based on default values
- User-friendly UI to manually set lifetime parameters
- Interaction through the API
Forensic support in MISP
Tools and visualization to support digital forensic

Team CIRCL
info@circl.lu
June 18, 2019
DFIR AND MISP DIGITAL EVIDENCES

- **Share analyses and reports** of digital forensic evidences.
- **Propose changes** to existing analyses or reports.
- Extending existing events with additional evidences for local or use in limited distribution sharing (sharing can be defined at event level or attribute level).
- **Evaluate correlations**\(^1\) of evidences against external or local attributes.
- **Report sightings** such as false-positive or true-positive (e.g. a partner/analyst has seen a similar indicator).

\(^1\)MISP has a flexible correlation engine which can correlate on 1-to-1 value matches, but also on fuzzy hashing (e.g. ssdeep) or CIDR block matching.
Benefits of using MISP

- LE can leverage the long-standing experience in information sharing and **bridge their use-cases** with MISP’s information sharing mechanisms.

- **Accessing existing MISP information sharing communities** by receiving actionable information from CSIRT/CERT networks or security researchers.

- **Bridging LE communities with other communities.** Sharing groups can be created (and managed) cross-sectors to support specific use-cases.

- The **MISP standard** is a flexible format which can be extended by users using the MISP platform. A MISP object template can be created in under 30 minutes, allowing users to rapidly share information using their own data-models with existing communities.
Challenges and implementations

- Standard sharing mechanism for forensic cases
  - MISP allows for the efficient collaborative analysis of digital evidences
  - Correlation on certain attributes

- Importing disk images and file system data activity (Mactime)
  - Development of an adaptable import tool: From Mactime to MISP Mactime object

- Create, modify and visualise the timeline of events
  - Development of a flexible timeline system at the event level
Possibility to import *Mactime* files [done]
Pick only relevant files [done]
MISPObject will be created [done]
Data visualization (MISP zoidberg branch)

- View: start-date only, spanning and search [dev-branch]
- Manipulate: Edit, Drag and Expand [dev-branch]
- Others: Timezone support [dev-branch]

→ For now [dev-branch], supports up to **micro-seconds** in the database and up to **milliseconds** in the web interface.
MISP restSearch API

An easy way to query, add and update your threat intelligence in MISP

Team CIRCL

MISP Training @ FIRST.org 2019
20190617
The MISP API has grown gradually with a UI first design in many cases.

Endpoints all solved specific issues with their own rulesets.

Growth was organic - whenever the need to add a new functionality / filter popped up we’ve added it.

Lead to Frankenmonsters such as this:

Goals we’ve set for ourselves

- Open up every functionality in MISP available via the UI to the API
- Including ones related to instance management
- APIs that expect input objects for data creation should be self-describing
- URL parameters should be discouraged, but still usable by legacy tools (deprecation)
- APIs should be heavily tested (Raphael Vinot’s exhaustive test suite in PyMISP)
- Largest focus on Export APIs
Export API’s reimagined

- Scrapped all existing type specific APIs (deprecated, documentation moved to legacy, still available)
- **Single entry point** - all export APIs baked into restSearch
- Queries consist of a combination of:
  - **Scope** (Event, Attribute, Sighting, more coming in the future)
  - **Filter parameters** - passed via JSON objects, url parameters (key value or ordered list)
  - A **return format**
- Everything that we could do before the rework we should be able to accomplish after the rework
- Under the hood now also used by the UI search and exports
One of our largest issues solved: **pagination**

- **Scope specific** pagination (number of events, attributes, etc)
- Simply control it via the framework friendly **page / limit** parameters
- Alternatively, use the improved **time based controls** (timestamp, publish_timestamp windows)
Performance tuning

- Single execution with subqueries
- Internal pagination aligned with memory limits
  - Probing of available memory for the current process
  - **Chunking of the query results** to fit in object specific memory envelopes
  - Constructing export set on disk in chunks has slashed memory usage considerably

Designing tools that use the APIs can be complex, but there’s help

- The result of our own frustration
- Built in **ReST client** with templating
- Extensive query builder UI by Sami Mokaddem
- Build queries in a simple interface, automatically set URLs, headers, etc
- Uses the self documentation of APIs
- Export your queries as **cURL or Python scripts**
- Built in testing tools (performance measurements, result parsers)
- Store queries for reuse and download the results directly
The maturity of the communities and threat intel sharing at large has improved.

We are sharing more.

Most importantly: we are sharing more context along with technical indicators.

This allows us to manage our data more accurately before feeding them to our protective tools.

Different contexts (APT targeting me? Persisting techniques?)
- lifecycle management

Use several queries / boolean operators to select the slice of data most relevant for the task.
CLI TOOLS FOR THE CLI GOD, AUTOMATION FOR THE AUTOMATION THRONE

- Open up commonly used system management tasks to the CLI
  - sync servers/feeds
  - caching feeds
  - Password resets
  - Server settings
  - Brute force protection resets
  - Enrichment
  - Worker management

- Goal was also to move away from the often malfunctioning scheduler and have cron friendly CLI scripts
So what does all of this look like in practice?

Demo time!
Plans for the Future

- Add export modules to the restSearch API
- Improve the query language to support some missing features (such as AND boolean operators)
- Support for extended events via the restSearch API
  - We’re missing a framing structure in the export module system (how are a list of conversions encapsulated and delimited?)
  - Proof of concept of the system implemented by Christian Studer already works using the STIX / STIX2 export subsystems
  - Would open us up to simple customiseable search APIs
- Open up search APIs to other scopes (objects, users, organisations, proposals, feeds, galaxies, taxonomies)
Virtual Machine (MISP Training VM)

The MISP Training VM is available at the following location: https://www.circl.lu/misp-images/latest/. The VM can be imported in VirtualBox or VMWare as an appliance (OVA).

The MISP training VM includes multiple applications and packages which are configured by default without production-ready secure settings. We strongly recommend to not use this VM for production and/or for storing sensitive information.

Normal URL and (username/password)
- MISP web interface - http://127.0.0.1 (NAT: http://127.0.0.1:8080) (admin@admin.test/admin)
- MISP-modules - http://127.0.0.1:6666
- MISP-dashboard - http://127.0.0.1:8001
- Viper-web - http://127.0.0.1:8888 (admin/Password1234)
- jupyter-notebook - http://127.0.0.1:8889
- system credentials via ssh/terminal - (misp/Password1234)

How to get the API key of my user?
Go to the MISP web interface, and simply click your username in the right upper corner to see your user profile which includes your API key.

How to reset a password in MISP?
If you did any specific mistake while setting up your password at the first logging. You can reset the password by login on the system (via SSH or terminal) and type the following command: /var/www/MISP/app/Console/cake Password admin@admin.test YourTemporaryPassword

How to reset the bruteforce login protection?
While trying to log into MISP multiple times unsuccessfully, the bruteforce protection might be triggered. You can reset the bruteforce login protection’s state by logging into the system (via SSH or terminal) and typing the following command: /var/www/MISP/app/Console/cake Admin clearBruteforce

How to upgrade MISP to the latest version?
Log in via SSH or terminal and type the following commands (your VM must have an Internet access):
1. cd /var/www/MISP
2. git pull origin 2.4
3. git submodule update --init --recursive

Getting OSINT information into your MISP
By default, a fresh installation of MISP is empty as we prefer to leave it up to the users to store, gather and share the information they need. If you would like to populate your MISP with some real-life data, simply enable the CIRCL OSINT feed, which contains cybersecurity threat-related information. In order to enable the OSINT feed, go to → Sync Actions then → List Feeds. Then select the first feed’s (called CIRCL OSINT Feed) checkbox and click on top Enable Selected. Then on the right side of the CIRCL OSINT Feed row, simply click the icon depicting a downward pointing arrow in a circle. Once you go back to the event index, the events will start appearing gradually.

Training materials and documentation
The MISP training materials are available at the following location https://www.circl.lu/services/misp-training-materials/ and are freely licensed under CC-BY-SA. MISP book is available at the following location https://www.circl.lu/doc/misp/.
## List of features to explain: User (MISP trainer support)

<table>
<thead>
<tr>
<th>Check</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add events</td>
<td>- via Standard UI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Distribution levels and publication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Different type of timestamps</td>
<td></td>
</tr>
<tr>
<td>Add attributes</td>
<td>- via Freetext</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- via Standard UI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- via Template</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- via ReST API (including freetext API?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- via EventGraph</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>- add Object</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- add References</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- show via EventGraph</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- add additional elements via the EventGraph</td>
<td></td>
</tr>
<tr>
<td>*.lists</td>
<td>- Warninglists: show warnings raised in steps above</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Noticelists: show warnings when adding data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Import Regexp: avoid leaking private/personal data</td>
<td></td>
</tr>
<tr>
<td>Correlations</td>
<td>- show correlations that were added</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- pivot to events via correlations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- show correlations graph</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- feeds &amp; servers correlation</td>
<td></td>
</tr>
<tr>
<td>Tags and Galaxies</td>
<td>- add Tag from Taxonomy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- add GalaxyCluster</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- add ATT&amp;CK pattern</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Creating and using Tag Collection</td>
<td></td>
</tr>
<tr>
<td>Sighting</td>
<td>- via UI + custom via UI (new source or expiration sighting)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- via API</td>
<td></td>
</tr>
<tr>
<td>Delegation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete (including soft versus hard delete)</td>
<td>- Event blacklist when deleting</td>
<td></td>
</tr>
<tr>
<td>Extending event (how and when to use it)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extracting the data</td>
<td>- download from</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- download from via modules</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- .json routing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- mass export</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- RestSearch</td>
<td></td>
</tr>
<tr>
<td>Searching for data</td>
<td>- Attribute search</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Event index filter search</td>
<td></td>
</tr>
</tbody>
</table>
List of features to explain: Administrator (MISP trainer support)

<table>
<thead>
<tr>
<th>Check</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td><strong>User</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- administration and contact via standard UI</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Roles</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td><strong>Organisations</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- local and remote</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- administration: Creation and merge</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Org admins and sync users</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td><strong>Sharing group</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- administration via standard UI</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td><strong>Templates</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- administration via standard UI</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Pulling and Updating</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td><strong>Jobs and Workers</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- administration via standard UI</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Scheduled Tasks and CRON jobs</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td><strong>Black listing</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Events</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Organisations</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td><strong>Searching</strong></td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Dashboard</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Event index</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Attributes: values, [not] tag</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Event level: quickfilter, contextual, distribution</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- Event level: event graph</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>- RestSearch</td>
<td></td>
</tr>
</tbody>
</table>
MISP Training Slide Decks

MISP\(^1\) is a threat intelligence platform for gathering, sharing, storing and correlating Indicators of Compromise of targeted attacks, threat intelligence, financial fraud information, vulnerability information or even counter-terrorism information.

This document includes the slides which are the support materials\(^2\) used for MISP trainings. The content is dual-licensed under CC-BY-SA version 4 license or GNU Affero General Public License version 3 which allows you to freely use, remixes and share-alike the slides while still mentioning the contributors under the same conditions.

Contributors

- Steve Clement [https://github.com/SteveClement](https://github.com/SteveClement)
- Alexandre Dulaunoy [https://github.com/adulau](https://github.com/adulau)
- Andras Iklody [https://github.com/iglocska](https://github.com/iglocska)
- Sami Mokaddem [https://github.com/mokaddem](https://github.com/mokaddem)
- Sascha Rommelfangen [https://github.com/rommelfs](https://github.com/rommelfs)
- Christian Studer [https://github.com/chrisr3d](https://github.com/chrisr3d)
- Raphaël Vinot [https://github.com/rafiot](https://github.com/rafiot)
- Gerard Wagener [https://github.com/haegardev](https://github.com/haegardev)

Acknowledgment

The MISP project is co-financed and resource supported by CIRCL Computer Incident Response Center Luxembourg\(^3\) and co-financed by a CEF (Connecting Europe Facility) funding under CEF-TC-2016-3 - Cyber Security as Improving MISP as building blocks for next-generation information sharing.

\(^1\)[https://www.misp-project.org/](https://www.misp-project.org/)
\(^2\)[https://github.com/MISP/misp-training](https://github.com/MISP/misp-training)
\(^3\)[https://www.circl.lu/](https://www.circl.lu/)