An Introduction to Cybersecurity Information Sharing

MISP - Threat Sharing

CIRCL / Team MISP Project

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

FIRST workshop
(13:00 - 14:00) Introduction to Information Sharing with MISP
(14:00 - 15:00) MISP hands on #1
(15:00 - 15:15) Break
(15:15 - 16:15) MISP hands on #2
(16:15 - 17:00) MISP Community building (or tomorrow depending on time) - QA session
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.
**What is MISP?**

- MISP is a **threat information sharing** platform that is free & open source software.
- A tool that **collects** information from partners, your analysts, your tools, feeds.
- Normalises, **correlates, enriches** the data.
- Allows teams and communities to **collaborate**.
- **Feeds** automated protective tools and analyst tools with the output.
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 Summit (/year)
- MISP Trainings (5-8)
- MISP project core team
- Experimental features
Many objectives from different user-groups

- 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)
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 1200 organizations with more than 4000 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).
- **Topical communities** set up to tackle individual specific issues (COVID-19 MISP)
Sharing Difficulties

Sharing difficulties are not really technical issues but often it’s a matter of social interactions (e.g. trust).

Legal restriction¹

- "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."

¹https://www.misp-project.org/compliance/
MISP Project Overview

Open Source Software
- MISP core
- misp-modules
- PyMISP
- misp-dashboard

Intelligence & Knowledge Base
- misp-taxonomies
- misp-galaxy
- misp-noticelist
- misp-warninglists

Open Standards
- MISP exchange core format
- MISP objects template

Intelligence & Sharing Community
- MISP OSINT feeds
- Compliance documents such as GDPR, ISO 27010:2015
- Threat intelligence best practices & training materials
- ISAC/ISAO best practices
Getting some naming conventions out of the way...

Data layer
- **Events** are encapsulations for contextually linked information
- **Attributes** are individual data points, which can be indicators or supporting data
- **Objects** are custom templated Attribute compositions
- **Object references** are the relationships between other building blocks
- **Sightings** are time-specific occurrences of a given data-point detected

Context layer
- **Tags** are labels attached to events/attributes and can come from Taxonomies
- **Galaxy-clusters** are knowledge base items used to label events/attributes and come from Galaxies
- **Cluster relationships** denote pre-defined relationships between clusters
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.

---

2IoC (Indicator of Compromise) is a subset of indicators
A RICH DATA-MODEL: TELLING STORIES VIA RELATIONSHIPS
MISP integrates at the event and the attribute levels MITRE’s Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK).
Sharing in MISP

- Sharing via distribution lists - **Sharing groups**
- **Delegation** for pseudo-anonymised information sharing
- **Proposals** and **Extended events** for collaborated information sharing
- Synchronisation, Feed system, air-gapped sharing
- User defined **filtered sharing** for all the above mentioned methods
- Cross-instance information **caching** for quick lookups of large data-sets
- Support for multi-MISP internal enclaves
MISP core distributed sharing functionality

- MISPs’ 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.
Information quality management

- Correlating data
- Feedback loop from detections via **Sightings**
- **False positive management** via the warninglist system
- **Enrichment system** via MISP-modules
- **Integrations** with a plethora of tools and formats
- Flexible **API** and support **libraries** such as PyMISP to ease integration
- **Timelines** and giving information a temporal context
- Full chain for **indicator life-cycle management**
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.
Sightings support

- Has a data-point been **sighted** by me or the community before?
- Additionally, the sighting system supports negative sightings (FP) and expiration sightings.
- Sightings can be performed via the API or the UI.
- Many use-cases for **scoring indicators** based on users sighting.
- For large quantities of data, **SightingDB** by Devo
TIMELINES AND GIVING INFORMATION A TEMPORAL CONTEXT

- Recently introduced **first_seen** and **last_seen** data points
- All data-points can be placed in time
- Enables the **visualisation** and **adjustment** of indicators timeframes
Life-cycle management via decaying of indicators

Decay score toggle button

- Shows Score for each Models associated to the Attribute type
Decaying of Indicators: Fine Tuning Tool

Create, modify, visualise, perform mapping
Simulate Attributes with different Models
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/)

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 data³.

---

³A 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

CIRCL / Team MISP Project

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

FIRST workshop
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)
MISP - Event (Attributes, giving meaning to events)
MISP - Event (Correlations on similar attributes)
MISP - Event (Proposals)
MISP - Event (Tags)

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

Tag
- Name
- Colour

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

Proposal
- Value
- Category
- Type
- IDS
- Comment
- Org

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

Correlation
MISP - EVENT (TAXONOMIES AND PROPOSAL CORRELATIONS)

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

**Tag**
- Name
- Colour

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

**Correlation**

**Discussion**

**Proposal**
- Value
- Category
- Type
- IDS
- Comment
- Org
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
MISP - Creating and populating events in various ways (demo)

- 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
Correlation graphs
Downloading the data in various formats
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
MISP - DISTRIBUTION AND TOPOLOGY

<table>
<thead>
<tr>
<th>Sharing group settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instances</strong></td>
</tr>
<tr>
<td>Misp 1</td>
</tr>
<tr>
<td>Misp 2</td>
</tr>
<tr>
<td>Misp 3 (all)</td>
</tr>
<tr>
<td>Misp 3 (all)</td>
</tr>
</tbody>
</table>
Download an event
Quick glance at the APIs
Download search results
ReST API and query builder
MISP - SHORTHAND ADMIN (IF NO ADMIN TRAINING)

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

CIRCL / Team MISP Project

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

FIRST workshop
A Common Integration

[Diagram showing integration of MISP, CSV feed, TheHive, graylog, and Cuckoo]
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).
Connecting Devices and Tools to MISP

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.
SIEM and MISP Integration

- 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.
ZMQ INTEGRATION: MISP-DASHBOARD

- 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

(https://securingtomorrow.mcafee.com/business/optimize-operations/expanding-automated-threat-hunting-response-open-
The Hive Integration
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)

- https://github.com/MISP/-
  http://www.misp-project.org/

- We welcome any contributions to the project, be it pull requests, ideas, github issues,...
Viper - Using MISP from your terminal

MISP - Threat Sharing

CIRCL / Team MISP Project

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

FIRST workshop
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)
<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>debup</td>
<td>Parse McAfee BUP Files</td>
</tr>
<tr>
<td>editdistance</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)
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
-d0 DOWNLOADED_OPEN, --download_open DOWNLOADED_OPEN
                         Open a file from the list of the DL files (ID)
-don DOWNLOADED_OPEN_NAME, --download_open_name DOWNLOADED_OPEN_NAME
                         Open a file by name from the list of the DL files (NAME)
-dd DOWNLOADED_DELETE, --download_delete DOWNLOADED_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
PASSIVE SSL

viper > pssl -h
        [-f FETCH] [-v] [-m {ips}]

Query a Passive SSL server

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
PASSIVE DNS

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 events based on the information contained within mails

CIRCL / Team MISP Project

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

FIRST workshop
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 (‘hxpx://...’)
- ... and more
Features: Support MISP features

- 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
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 = 's5jPWC1ud36Z8XHgsiCVl7SaL1XsMTyfEsN45tTe'
     misp_verifystem = True
     body_config_prefix = 'm2m'
     ...
     ```
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_sms.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: m2m
    250 OK
    data
    354 End data with <CR><LF><CR><LF>
MISP User Training - Administration of MISP 2.4

MISP Threat Sharing

CIRCL / Team MISP Project

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

FIRST workshop
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
MISP - Creating Users

- 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, Regular expressions, 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
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

CIRCL / Team MISP Project

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

FIRST workshop
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 built a new scheme using the concept of machine tags.
Triple tag, or machine tag, format was introduced in 2004 to extend geotagging on images.

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

Still have a human-readable version:

```plaintext
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 repository\(^1\).
These can be freely reused and integrated into other threat intel tools.
Taxonomies are licensed under Creative Commons (public domain) except if the taxonomy author decided to use another license.

\(^1\)https://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
{
    "namespace": "admiralty-scale",
    "description": "The Admiralty Scale (also called the NATO System) is used to rank the reliability of a source and the credibility of an information.",
    "version": 1,
    "predicates": [
        {
            "value": "source-reliability",
            "expanded": "Source Reliability"
        },
        {
            "value": "information-credibility",
            "expanded": "Information Credibility"
        }
    ]
}
```
Publishing your taxonomy is as easy as a simple git pull request on misp-taxonomies².

²https://github.com/MISP/misp-taxonomies
How are taxonomies integrated in MISP?

- MISP administrator can just import (or even cherry pick) the namespace or predicates they want to use as tags.
- Tags can be exported to other instances.
- Tags are also accessible via the MISP REST API.
FILTERING THE DISTRIBUTION OF EVENTS AMONG MISP INSTANCES

- Applying rules for distribution based on tags:
Other use cases using MISP taxonomies

- 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 independent 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"
The dilemma of false-positives

False-positives are a common issue in threat intelligence sharing.

It’s often a contextual issue:

- False-positives might be different per community of users sharing information.
- Organizations might have their own view on false-positives.

Based on the success of the MISP taxonomy model, we built misp-warninglists.
MISP WARNING LISTS

- misp-warninglists 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...
Q&A

- 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

MISP - Threat Sharing

CIRCL / Team MISP Project

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

FIRST workshop
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
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 functionnality introduced in MISP 2.4.28.

MISP import/export modules introduced in MISP 2.4.50.
MISP modules - installation

- 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

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

```
{
    "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"
    ]
}
```
### 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 sourcecache</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_sourcecache_archivepath</td>
<td></td>
<td>Set this required module specific</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_passivetotal_enabled</td>
<td>true</td>
<td>Enable or disable the passivetotal</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 specific</td>
</tr>
<tr>
<td>Recommended</td>
<td>Plugin.Enrichment_passivetotal_password</td>
<td></td>
<td>Set this required module specific</td>
</tr>
</tbody>
</table>
MISP MODULES - HOW IT’S INTEGRATED IN THE UI?
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
**Querying a Module**

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

  body.json

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

- and the response of the dns module:

  ```json
  1  { "results": [ { "values": [ "149.13.33.14" ],
        "types": [ "ip-src", "ip-dst" ] } ] }
  ```
import json
import dns.resolver
misperrors = {'error': 'Error'}
mispattributest = {'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": mispattributest["output"], "values": [str(answer[0])]}]}
    return r

def introspection():
    return mispattributest

def version():
    return moduleinfo
Testing your module

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

```bash
[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 = PySSL.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
Similar to expansion modules
Input is a file upload or a text paste
Output is a list of parsed attributes to be editend and verified by the user
Some examples
- Cuckoo JSON import
- email import
- OCR module
- Open IoC import
Export modules

- Not the preferred way to export data from MISP
- Input is currently only a single event
- Output is a file in the export format served back to the user
- Will be moved / merged with MISP built-in export modules
  - Allows export of event / attribute collections
New expansion & import modules format

- Backward compatible - an additional field to extend the format
  
  ```
  misp_attributes = {
      'input': [...],
      'output': [...],
      'format': 'misp_standard'
  }
  ```

- Takes a standard MISP attribute as input
- Returns MISP format
  - Attributes
  - Objects (with their references)
  - Tags

  ```
  results = {
      'Attribute': [...],
      'Object': [...],
      'Tag': [...]
  }
  ```

- First modules supporting this new export format
  - urlhaus expansion module
  - Joe Sandbox import & query module
New Expansion & Import Modules View (MISP 2.4.110)
Future of the modules system

- Enrichment on full events
- Move the modules to background processes with a messaging system
- Have a way to skip the results preview
  - Preview can be very heavy
  - Difficulty is dealing with uncertain results (without the user having final say)
We welcome new modules and pull requests.

MISP modules can be designed as standalone application.
MISP Galaxy

CIRCL / Team MISP Project

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

FIRST workshop
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.
Solution

- 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)
**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™)
### 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 Schrecks
- Timo Steffens
- Various

**Country**
- RU

**Refs**

[Add new cluster]
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
Use a searchable synonym database to find what you’re after

- **Sofacy**
  - Synonyms: APT 28, APT28, Pawn Storm, PawnStorm, Fancy Bear, Sednit, SNAKEMACKEREL, TsarTeam, Tsar Team, TG-4127, Group 4127, STRONTIUM, TAG_0700, Swallowtail, IRON TWILIGHT, Group 74
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 warninglists
- 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

```json
{
    "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": [{},]
    }
  ]
}
```
Cluster JSON value example

```json
{
  "meta": {
    "synonyms": [
      "APT 28", "APT28", "Pawn Storm", "Fancy Bear",
      "Sednit", "TsarTeam", "TG−4127", "Group−4127",
      "STRONTIUM", "Grey−Cloud"
    ],
    "country": "RU",
    "refs": [
      "https://en.wikipedia.org/wiki/Sofacy_Group"
    ]
  },
  "description": "The Sofacy Group (also known as APT28, 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.",
  "value": "Sofacy"
}
```

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://cdnO.vox−cdn.com/assets/4589853/crowdstrike−intelligence−report−putter−panda.original.pdf",
    "https://www.cfr.org/interactive/cyber−operations/putter−panda"
# Galaxy JSON Matrix-Like

<table>
<thead>
<tr>
<th>Propose Attribute</th>
<th>Analysis</th>
<th>Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>example-of-threats</strong></td>
<td>**Setup</td>
<td>party/candidate registration (3 items)**</td>
</tr>
<tr>
<td>DoS or overload of party/campaign registration, causing them to miss the deadline</td>
<td>Deleting or tampering with voter data</td>
<td>Hacking campaign websites (defacement, DoS)</td>
</tr>
<tr>
<td>Fabricated signatures from sponsor</td>
<td>DoS or overload of voter registration system, suppressing voters</td>
<td>Hacking candidate laptops or email accounts</td>
</tr>
<tr>
<td>Tampering with registrations</td>
<td>Identity fraud during voter registration</td>
<td>Leak of confidential information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Misconfiguration of a website</td>
</tr>
</tbody>
</table>
{  
  "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
1 {
   "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.

```json
"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"
```
from pymispgalaxies import Clusters

c = Clusters()
l = list(g.keys())

# ['threat-actor', 'ransomware', 'exploit-kit', 'tds', 'tool', 'rat', 'mitre-attack-pattern',
# 'mitre-tool', 'microsoft-activity-group', 'mitre-course-of-action', 'mitre-malware',
# '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

CIRCL / Team MISP Project

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

FIRST workshop
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.
MISP Object Templates

- Using a similar JSON format as the taxonomies, galaxies, warninglists.
- You can find the default set of object templates in the git repository\(^1\).
- 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.

\(^1\)https://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

1 {
2   "requiredOneOf": [],
3   "required": [],
4   "attributes": {},
5   "version": 1,
6   "description": "My description",
7   "meta-category": "Chosen meta category",
8   "uuid": "Object template uuid",
9   "name": "Object template name"
10 }
Adding elements to an object template

```
"regexp-type": {
  "description": "Type of the regular expression syntax.",
  "disable_correlation": true,
  "ui-priority": 0,
  "misp-attribute": "text",
  "values_list": [
    "PCRE",
    "PCRE2",
    "POSIX BRE",
    "POSIX ERE"
  ]
},
```
**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
### Add File Object

<table>
<thead>
<tr>
<th>Object Template</th>
<th>Description</th>
<th>Requirements</th>
<th>Meta category</th>
<th>Distribution</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>File v10</td>
<td>File object describing a file with meta-information</td>
<td>Required one of: 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>
<td>File</td>
<td>Inherit event</td>
<td></td>
</tr>
</tbody>
</table>

### Save Name :: Type | Description | Category | Value |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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>
</tbody>
</table>
### Templates as Rendered in the UI

<table>
<thead>
<tr>
<th>Date</th>
<th>Reference</th>
<th>Payload Delivery</th>
<th>File Name</th>
<th>File Size</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-03-27</td>
<td></td>
<td><code>payload.exe</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Other</td>
<td><code>size-in-bytes</code></td>
<td>774200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Other</td>
<td><code>entropy</code></td>
<td>6.728597226</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MD5** and **SHA1** values are provided for each payload:

<table>
<thead>
<tr>
<th>Date</th>
<th>Reference</th>
<th>Payload Delivery</th>
<th>MD5</th>
<th>SHA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-03-27</td>
<td></td>
<td><code>payload.exe</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-03-27</td>
<td></td>
<td><code>sha256</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018-03-27</td>
<td></td>
<td><code>sha512</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Malware Sample**:

- `payload.exe`
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 C002 4E06 BA70 BE89 EAAD CFFC 22BD 4CD5
MISP Dashboard

Real-time overview of threat intelligence from MISP instances

CIRCL / Team MISP Project

info@circl.lu

May 19, 2021
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
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:
   $ ./zmqDispatcher.py
6. Start the Flask server:
   $ ./server.py
7. Access the interface at http://localhost:8081/
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)
Recent changes in the MISP-dashboards

- MISP authentication can now be used in the misp-dashboard
- Improved TLS/SSL support in the default misp-dashboard
- Self-test tool to debug and test ZMQ connectivity
Future development

- 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
  - ACL enabled version
MISP-Dashboard can provide real-time 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 proposes 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

CIRCL / Team MISP Project

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

FIRST workshop
The MISP project has a Contributor Covenant Code of Conduct¹.

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.

¹https://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: CA57 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 via Github Actions.

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, the CI suite is automatically called\(^2\).

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

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

\(^2\)https://github.com/MISP/MISP/actions
JSON validation for MISP libraries

- All JSON format (galaxy, taxonomies, objects or warning-lists) are described in a JSON Schema\(^3\).
- The TravisCI tests are including JSON validation (via `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 `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\(^4\).

\(^3\)schema_name.json
\(^4\)example https://github.com/MISP/misp-galaxy/blob/master/tools/asciidoctor.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-rfc⁸ 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 IETF⁹.
- 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

CIRCL / Team MISP Project

MISP Threat Sharing

FIRST workshop
Some things to know in advance...

- MISP is based on PHP 7.3+
- 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
Controllers - Scope

- 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
- IOCIImport component (should be moved)
CONTROLLERS - ADDITIONAL FUNCTIONALITIES

- Handling API responses (RestResponseComponent)
- Handling API requests (IndexFilterComponent)
- auth/session management
- ACL management
- CRUD Component
- 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
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)
- could be interesting if we ever validated without saving
- beforeSave() (purge existing correlations for an attribute)
- afterSave() (create new correlations for an attribute / zmq)
Models - hooking pipeline (delete/read)

- 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)
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
Mostly in genericElements

Preparing the move to Cake4

Important ones

- Form - generate forms in a standardised way (/add, /edit, etc)
- IndexTable - index lists using Field templates (/index, etc)
- SingleViews - key-value lists with child elements (/view, etc)
- Menues - to be refactored, see Cerebrate
General reusable libraries

- Located in app/Lib
- Code that is to be reused across several layers
- Important ones
  - Dashboard - Dashboard widget backend code
  - EventReport - Report generation
  - Export - MISP -> external format converter modules
  - Tools - List of generic helper libraries - examples:
    - Attachment, JSON conversion, random generation, emailing, sync request generation
    - Kafka, ZMQ, AWS S3, Elastic integration, PGP encryption, CIDR operations
Distribution

- 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

- funtional testing
- Github actions
- 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

CIRCL / Team MISP Project

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

FIRST workshop
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.
Common queries: Search things

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).
COMMON QUERIES: CREATE THINGS

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)
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)
- 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 ENTRYPONTS

- add_tag(tag=None, **kwargs)
- delete()
- malware_binary (if relevant)
- tags[]
- edited, all other parameters of the MISPObject element (value, comment, ...)
- to_json()
**PYMISP - TOOLS**

- 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
**PyMISP - Logging / Debugging**

- 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)
```
We welcome new functionalities and pull requests.

- https://github.com/MISP/PyMISP
- https://github.com/MISP/
- https://pymisp.readthedocs.io/
MISP feeds - A simple and secure approach to generate, select and collect intelligence

Providing ready-to-use threat intelligence in MISP standard format

CIRCL / Team MISP Project

TLP:WHITE

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

FIRST workshop
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'

definitions = {
    '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
Real-time Feed generator - Purpose

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:

Honeypot \[\rightarrow\] MISP

ip-src
payload-delivery
url
url
malware
...
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
# Adding a MISP object (cowrie) to the daily event
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 for CSIRTs

Team CIRCL
TLP:WHITE

FIRST workshop
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 **1250 organisations**
  - **3600 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-ISAC²
  - 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

²https://www.x-isac.org/
the ATT&CK EU community\(^3\)

- Work on attacker modelling
- With the assistance of MITRE themselves
- Unique opportunity to **standardise on TTPs**
- Looking for organisations that want to get involved!

\(^3\)https://www.attack-community.org/
Communities supported by CIRCL

- ISAC / specialised community MISPs
  - Topical or community specific instances hosted or co-managed by CIRCL
  - Examples, GSMA, FIRST.org, CSIRT network, etc
  - Often come with their own taxonomies and domain specific object definitions

- FIRST.org’s MISP community

- Telecom and Mobile operators’ community

- Various ad-hoc communities for exercises for example
  - The ENISA exercise for example
  - Locked Shields exercise
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 **AIL**)

---

4https://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 PROACTIVE SERVICES - MISP DASHBOARD
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
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.
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**
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 immitate over expecting adherence to rules

- **Lead by example** - the power of immitation
- 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 make sense right now, you never know when they come handy...
How to deal with organisations that only "leech"?

- 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 losing access are the ones 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 contributions 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 valueable 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 you**, so they can see why it could be **useful to them**

Massively important once an organisation has the maturity to filter the most critical **subsets of information for their own defense**
MISP has a verify 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.
Many objectives from different user-groups

- 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.

Warning: Potential false positives

List of known IPv4 public DNS resolvers
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 bootstraping 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** if needed.

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
MISP and Decaying of Indicators

An indicator scoring method and ongoing implementation in MISP

Team CIRCL

info@circl.lu

May 19, 2021
Sharing information about threats is crucial
Organisations are sharing more and more

Contribution by unique organisation (Orgc.name) on MISPPriv:

<table>
<thead>
<tr>
<th>Date</th>
<th>Unique Org</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>17</td>
</tr>
<tr>
<td>2014</td>
<td>43</td>
</tr>
<tr>
<td>2015</td>
<td>82</td>
</tr>
<tr>
<td>2016</td>
<td>105</td>
</tr>
<tr>
<td>2017</td>
<td>118</td>
</tr>
<tr>
<td>2018</td>
<td>125</td>
</tr>
<tr>
<td>2019-10</td>
<td>135</td>
</tr>
</tbody>
</table>

```json
{
  "distribution": [1, 2, 3]
}
```
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
  - Conflicting interests such as operational security, attribution,... (depends on the user)

→ Can be partially solved with *Taxonomies*
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
  - Conflicting interests such as operational security, attribution,... (depends on the user)

→ Can be partially solved with **Taxonomies**

Attributes can be shared in large quantities (more than 7.3 million on MISPPRIV)

- Partial info about their **freshness** (**Sightings**)  
- Partial info about their **validity** (last update)

→ Can be partially solved with our **Decaying model**
Starting from **MISP 2.4.116**, the decaying feature is available.

Don’t forget to update the decay models and enable the ones you want.

The decaying feature has no impact on the information in MISP; it’s just an overlay to be used in the user-interface and API.

Decay strongly relies on *Taxonomies* and *Sightings*, don’t forget to review their configuration.
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

- Users might have other contextual needs

→ Achieved thanks to *Taxonomies*
Tagging is a simple way to attach a classification to an Event or an Attribute

Classification must be globally used to be efficient
**ADMIRALTY-SCALE Taxonomy Library**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expanded</th>
<th>Numerical Value</th>
<th>Events</th>
<th>Attributes</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>admiralry-scale:information-credibility=&quot;1&quot;</td>
<td>Information Credibility: Confirmed by other sources</td>
<td>100</td>
<td>6</td>
<td>0</td>
<td>admiralry-scale:information-credibility=&quot;1&quot;</td>
</tr>
<tr>
<td>admiralry-scale:information-credibility=&quot;2&quot;</td>
<td>Information Credibility: Probably true</td>
<td>75</td>
<td>21</td>
<td>1</td>
<td>admiralry-scale:information-credibility=&quot;2&quot;</td>
</tr>
<tr>
<td>admiralry-scale:information-credibility=&quot;3&quot;</td>
<td>Information Credibility: Possibly true</td>
<td>50</td>
<td>16</td>
<td>5</td>
<td>admiralry-scale:information-credibility=&quot;3&quot;</td>
</tr>
<tr>
<td>admiralry-scale:information-credibility=&quot;4&quot;</td>
<td>Information Credibility: Doubtful</td>
<td>25</td>
<td>2</td>
<td>0</td>
<td>admiralry-scale:information-credibility=&quot;4&quot;</td>
</tr>
<tr>
<td>admiralry-scale:information-credibility=&quot;5&quot;</td>
<td>Information Credibility: Improbable</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>admiralry-scale:information-credibility=&quot;5&quot;</td>
</tr>
<tr>
<td>admiralry-scale:information-credibility=&quot;6&quot;</td>
<td>Information Credibility: Truth cannot be judged</td>
<td>50</td>
<td>9</td>
<td>2</td>
<td>admiralry-scale:information-credibility=&quot;6&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;a&quot;</td>
<td>Source Reliability: Completely reliable</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>admiralry-scale:source-reliability=&quot;a&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;b&quot;</td>
<td>Source Reliability: Usually reliable</td>
<td>75</td>
<td>21</td>
<td>76</td>
<td>admiralry-scale:source-reliability=&quot;b&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;c&quot;</td>
<td>Source Reliability: Fairly reliable</td>
<td>50</td>
<td>9</td>
<td>8</td>
<td>admiralry-scale:source-reliability=&quot;c&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;d&quot;</td>
<td>Source Reliability: Not usually reliable</td>
<td>25</td>
<td>2</td>
<td>0</td>
<td>admiralry-scale:source-reliability=&quot;d&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;e&quot;</td>
<td>Source Reliability: Unreliable</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>admiralry-scale:source-reliability=&quot;e&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;f&quot;</td>
<td>Source Reliability: Reliability cannot be judged</td>
<td>50</td>
<td>10</td>
<td>7</td>
<td>admiralry-scale:source-reliability=&quot;f&quot;</td>
</tr>
<tr>
<td>admiralry-scale:source-reliability=&quot;g&quot;</td>
<td>Source Reliability: Deliberately deceptive</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

→ Cherry-pick allowed Tags
Some taxonomies have numerical_value

→ Can be used to prioritise Attributes

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely reliable</td>
<td>100</td>
</tr>
<tr>
<td>Usually reliable</td>
<td>75</td>
</tr>
<tr>
<td>Fairly reliable</td>
<td>50</td>
</tr>
<tr>
<td>Not usually reliable</td>
<td>25</td>
</tr>
<tr>
<td>Unreliable</td>
<td>0</td>
</tr>
<tr>
<td>Reliability cannot be judged</td>
<td>50 ?</td>
</tr>
<tr>
<td>Deliberatly deceptive</td>
<td>0 ?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirmed by other sources</td>
<td>100</td>
</tr>
<tr>
<td>Probably true</td>
<td>75</td>
</tr>
<tr>
<td>Possibly true</td>
<td>50</td>
</tr>
<tr>
<td>Doubtful</td>
<td>25</td>
</tr>
<tr>
<td>Improbable</td>
<td>0</td>
</tr>
<tr>
<td>Truth cannot be judged</td>
<td>50 ?</td>
</tr>
</tbody>
</table>
Scoring Indicators: Our solution

\[
\text{score}(\text{Attribute}) = \text{base\_score}(\text{Attribute, Model}) \cdot \text{decay}(\text{Model, time})
\]

Where,

- \( \text{score} \in [0, +\infty) \)
- \( \text{base\_score} \in [0, 100] \)
- \( \text{decay} \) is a function defined by model’s parameters controlling decay speed
- \( \text{Attribute} \) Contains \( \text{Attribute} \)'s values and metadata \((\text{Taxonomies, Galaxies, ...})\)
- \( \text{Model} \) Contains the \( \text{Model} \)'s configuration
Current implementation in MISP
### IMPLEMENTATION IN MISP: Event/view

- **Decay score toggle button**
  - Shows Score for each *Models* associated to the *Attribute* type
/attributes/restSearch

```
"Attribute": [
  {
    "category": "Network activity",
    "type": "ip-src",
    "to_ids": true,
    "timestamp": "1565703507",
    [...] 
    "value": "8.8.8.8",
    "decay_score": [
      {
        "score": 54.475223849544456,
        "decayed": false,
        "DecayingModel": {
          "id": "85",
          "name": "NIDS Simple Decaying Model"
        }
      }
    ],
    [...] 
  }
]```

Implementation in MISP: Playing with Models

- **Automatic scoring** based on default values
- **User-friendly UI** to manually set *Model* configuration (lifetime, decay, etc.)
- **Simulation** tool
- Interaction through the **API**
- Opportunity to create your **own** formula or algorithm
Decaying Models in Depth
Scoring Indicators: base_score (1)

\[
\text{score(Attribute)} = \text{base\_score(Attribute, Model)} \cdot \text{decay(Model, time)}
\]

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

- Data reliability, credibility, analyst skills, custom prioritisation tags (economical-impact), etc.
- Trust in the source

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

Where,

\[
\omega_{sc} + \omega_{tg} = 1
\]

---

\(^1\)Paper available: https://arxiv.org/pdf/1803.11052
\(^2\)at a variable extent as required
Scoring Indicators: base_score (2)

Current implementation ignores source_confidence:

→ base_score = tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Computation</th>
<th>Eff. Ratio</th>
<th>numerical_value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>admiralty-scale:source-reliability=&quot;Completely reliable&quot;</td>
<td></td>
<td>0.50</td>
<td>* 100.00</td>
<td>50.00</td>
</tr>
<tr>
<td>phishing:psychological-acceptability=&quot;high&quot;</td>
<td></td>
<td>0.50</td>
<td>* 75.00</td>
<td>37.50</td>
</tr>
</tbody>
</table>

→ The base_score can be used to prioritize attributes based on their attached context and source
Scoring Indicators: decay speed (1)

\[ \text{score}(\text{Attribute}) = \text{base\_score}(\text{Attribute, Model}) \cdot \text{decay}(\text{Model, time}) \]

The decay is calculated using:

- The lifetime of the indicator
  - May vary depending on the indicator type
  - Short for an IP, long for an hash
- The decay rate, or speed at which an attribute loses score over time
- The time elapsed since the latest update or sighting
→ decay rate is **re-initialized upon sighting** addition, or said differently, the score is reset to its base score as new sightings are applied.

\[
\text{score} = \text{base\_score} \cdot \left( 1 - \left( \frac{t}{\tau} \right)^{\frac{1}{\delta}} \right)
\]

- \(\tau = \text{lifetime}\)
- \(\delta = \text{decay speed}\)
Implentation in MISP: Models definition

\[ \text{score} = \text{base}\_\text{score} \cdot \left( 1 - \left( \frac{t}{\tau} \right)^{\frac{1}{\delta}} \right) \]

Models are an instantiation of the formula where elements can be defined:

- Parameters: lifetime, decay_rate, threshold
- base_score
- default base_score
- formula
- associate Attribute types
- creator organisation
Multiple model types are available

- **Default Models**: Models created and shared by the community. Available from misp-decaying-models repository³.
  - → Not editable

- **Organisation Models**: Models created by a user belonging to an organisation
  - These models can be hidden or shared to other organisation
  - → Editable

³https://github.com/MISP/misp-decaying-models.git
Decaying Models

<table>
<thead>
<tr>
<th>ID</th>
<th>Organization</th>
<th>Usable to everyone</th>
<th>Name</th>
<th>Description</th>
<th>Parameters</th>
<th>Formula</th>
<th># Assigned</th>
<th>Version</th>
<th>Enabled</th>
<th>Actions</th>
</tr>
</thead>
</table>
| 29 | 1            | ✓                  | Phishing model | Simple model to rapidly decay phishing website. | ```
{"lifetime": 3,
"decay_speed": 2.3,
"threshold": 30,
"default_base_score": 80,
"base_score_config": {"estimative-language": 0.5,
"phishing": 0.5
}
``` | Polynomial 5 | 9 | 1 | ✓ | |

| 85 | 1            | ✗                  | NIDS Simple Decaying Model | Simple decaying model for Network Intrusion Detection System (NIDS). | ```
{"lifetime": 120,
"decay_speed": 2,
"threshold": 30,
"default_base_score": 80,
"base_score_config": {"estimative-language": 0.25,
"priority-level": 0.25,
"retention": 0.25,
"targeted-threat-index": 0.125,
"false-positive": 0.125
}
``` | Polynomial 13 | 1 | 1 | ✓ | |

Page 1 of 1, showing 2 records out of 2 total, starting on record 1, ending on 2

View, update, add, create, delete, enable, export, import
IMPLEMENTATION IN MISP: FINE TUNING TOOL

Create, modify, visualise, perform mapping
IMPLEMENTATION IN MISP: base_score TOOL

Search Taxonomy

Default basesscore 80

Taxonomies

admimality-scale

source-reliability

information-credibility

priority-level

priority-level

reitention

reitention

estimative-language

likelihood-probability

confidence-in-analytic-judgment

misx

confidence-level

threat-level

automation-level

phising

state

psychological-acceptability

Excluded

Example

Attribute | Tags | Base score
--- | --- | ---
Tag your attribute |  | 
Attribute 1 | admimality-scale-information-credibility="5" | 0.0
Attribute 2 | priority-level:baseline-minor admimality-scale:source-reliability="d" admimality-scale-information-credibility="2" | 38.2
Attribute 3 | priority-level:severe admimality-scale:information-credibility="2" | 84.6

Computation steps

Tag | Computation | Result
--- | --- | ---
priority-level:baseline-minor | 0.46 * 25.00 | 11.52
admimality-scale:source-reliability="d" | 0.27 * 25.00 | 0.80
Simulate Attributes with different Models
/attributes/restSearch

```json
{
    "includeDecayScore": 1,
    "includeFullModel": 0,
    "excludeDecayed": 0,
    "decayingModel": [85],
    "modelOverrides": {
        "threshold": 30
    }
    "score": 30,
}
```
Creating a new decay algorithm (1)

The current architecture allows users to create their own formulae.

1. Create a new file $filename in app/Model/DecayingModelsFormulas/
2. Extend the Base class as defined in DecayingModelBase
3. Implement the two mandatory functions computeScore and isDecayed using your own formula/algorithm
4. Create a Model and set the formula field to $filename

Use cases:
- Add support for more feature (expiration taxonomy)
- Query external services then influence the score
- Completely different approach (i.e streaming algorithm)
- ...
<?php
include_once 'Base.php';

class Polynomial extends DecayingModelBase {
    public const DESCRIPTION = 'The description of your new decaying algorithm';

    public function computeScore($model, $attribute, $base_score, $elapsed_time) {
        // algorithm returning a numerical score
    }

    public function isDecayed($model, $attribute, $score) {
        // algorithm returning a boolean stating if the attribute is expired or not
    }

}
Decaying Models 2.0

- Improved support of *Sightings*
  - False positive *Sightings* should somehow reduce the score
  - Expiration *Sightings* should mark the attribute as decayed

- Potential *Model* improvements
  - Instead of resetting the score to `base_score` once a *Sighting* is set, the score should be increased additively (based on a defined coefficient); thus prioritizing surges rather than infrequent *Sightings*
  - Take into account related *Tags* or *Correlations* when computing score

- Increase *Taxonomy* coverage
  - Users should be able to manually override the numerical_value of *Tags*

- For specific type, take into account data from other services
  - Could fetch data from *BGP ranking, Virus Total, Passive X* for IP/domain/... and adapt the score
MISP and Decaying of Indicators

Primer for indicator scoring in MISP

Team CIRCL

info@circl.lu

May 19, 2021
Present the components used in MISP to expire IOCs

Present the current state of Indicators life-cycle management in MISP
Expiring IOCs: Why and How?
Sharing information about threats is crucial
- Organisations are sharing more and more

Contribution by unique organisation (Orgc.name) on MISPPriv:

<table>
<thead>
<tr>
<th>Date</th>
<th>Unique Org</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>17</td>
</tr>
<tr>
<td>2014</td>
<td>43</td>
</tr>
<tr>
<td>2015</td>
<td>82</td>
</tr>
<tr>
<td>2016</td>
<td>105</td>
</tr>
<tr>
<td>2017</td>
<td>118</td>
</tr>
<tr>
<td>2018</td>
<td>125</td>
</tr>
<tr>
<td>2019-10</td>
<td>135</td>
</tr>
</tbody>
</table>

```json
{
  "distribution": [1, 2, 3]
}
```
Various users and organisations can share data via MISP, multiple parties can be involved

- **Trust**, data quality and relevance issues
- Each user/organisation have different use-cases and interests
  - Conflicting interests: Operational security VS attribution

→ Can be partially solved with *Taxonomies*
Various users and organisations can share data via MISP, multiple parties can be involved

- Trust, data quality and relevance issues
- Each user/organisation have different use-cases and interests
  - Conflicting interests: Operational security VS attribution

→ Can be partially solved with Taxonomies

Attributes can be shared in large quantities (more than 12M on MISPPRIV - Sept. 2020)

- Partial info about their freshness (Sightings)
- Partial info about their validity (last_seen)

→ Can be partially solved with our Data model

MISP’s Decaying model combines the two
Requirements to enjoy the decaying feature in MISP

- Starting from **MISP 2.4.116**, the decaying feature is available
- **Update** decay models and **enable** some
- MISP Decaying strongly relies on *Taxonomies* and *Sightings*, don’t forget to review their configuration

Note: The decaying feature has no impact on the information stored in MISP, it’s just an **overlay** to be used in the user-interface and API
Sightings add a temporal context to indicators.

- **Sightings** can be used to represent that you saw the IoC
- **Use case:** Continuous feedback loop MISP ↔ IDS
Sightings add a **temporal context** to indicators.

- Sightings give more credibility/visibility to indicators
- This information can be used to **prioritise and decay indicators**
Taxonomies are a simple way to attach a classification to an Event or an Attribute.

Classification must be globally used to be efficient (or agreed on beforehand).
Taxonomies - Refresher (2)

ADMIRALTY-SCALE Taxonomy Library

<table>
<thead>
<tr>
<th>Tag</th>
<th>Expanded</th>
<th>Numerical Value</th>
<th>Events</th>
<th>Attributes</th>
<th>Tags</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>admiralty-scale:information-credibility=&quot;1&quot;</td>
<td>Information Credibility: Confirmed by other sources</td>
<td>100</td>
<td>6</td>
<td>0</td>
<td>admiralty-scale:information-credibility=&quot;1&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:information-credibility=&quot;2&quot;</td>
<td>Information Credibility: Probably true</td>
<td>75</td>
<td>21</td>
<td>1</td>
<td>admiralty-scale:information-credibility=&quot;2&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:information-credibility=&quot;3&quot;</td>
<td>Information Credibility: Possibly true</td>
<td>50</td>
<td>16</td>
<td>5</td>
<td>admiralty-scale:information-credibility=&quot;3&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:information-credibility=&quot;4&quot;</td>
<td>Information Credibility: Doubtful</td>
<td>25</td>
<td>2</td>
<td>0</td>
<td>admiralty-scale:information-credibility=&quot;4&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:information-credibility=&quot;5&quot;</td>
<td>Information Credibility: Improbable</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>admiralty-scale:information-credibility=&quot;5&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:information-credibility=&quot;6&quot;</td>
<td>Information Credibility: Truth cannot be judged</td>
<td>50</td>
<td>9</td>
<td>2</td>
<td>admiralty-scale:information-credibility=&quot;6&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;a&quot;</td>
<td>Source Reliability: Completely reliable</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>admiralty-scale:source-reliability=&quot;a&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;b&quot;</td>
<td>Source Reliability: Usually reliable</td>
<td>75</td>
<td>9</td>
<td>21</td>
<td>admiralty-scale:source-reliability=&quot;b&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;c&quot;</td>
<td>Source Reliability: Fairly reliable</td>
<td>75</td>
<td>9</td>
<td>8</td>
<td>admiralty-scale:source-reliability=&quot;c&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;d&quot;</td>
<td>Source Reliability: Not usually reliable</td>
<td>50</td>
<td>2</td>
<td>0</td>
<td>admiralty-scale:source-reliability=&quot;d&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;e&quot;</td>
<td>Source Reliability: Unreliable</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>admiralty-scale:source-reliability=&quot;e&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;f&quot;</td>
<td>Source Reliability: Reliability cannot be judged</td>
<td>50</td>
<td>10</td>
<td>7</td>
<td>admiralty-scale:source-reliability=&quot;f&quot;</td>
<td>-</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=&quot;g&quot;</td>
<td>Source Reliability: Deliberately deceptive</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Cherry-pick allowed Tags
Some taxonomies have a numerical value
- Allows concepts to be used in a mathematical expression
  → Can be used to prioritise IoCs

**Admiralty-scale taxonomy**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely reliable</td>
<td>100</td>
</tr>
<tr>
<td>Usually reliable</td>
<td>75</td>
</tr>
<tr>
<td>Fairly reliable</td>
<td>50</td>
</tr>
<tr>
<td>Not usually reliable</td>
<td>25</td>
</tr>
<tr>
<td>Unreliable</td>
<td>0</td>
</tr>
<tr>
<td>Reliability cannot be judged</td>
<td>50</td>
</tr>
<tr>
<td>Deliberately deceptive</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirmed by other sources</td>
<td>100</td>
</tr>
<tr>
<td>Probably true</td>
<td>75</td>
</tr>
<tr>
<td>Possibly true</td>
<td>50</td>
</tr>
<tr>
<td>Doubtful</td>
<td>25</td>
</tr>
<tr>
<td>Improbable</td>
<td>0</td>
</tr>
<tr>
<td>Truth cannot be judged</td>
<td>50</td>
</tr>
</tbody>
</table>

---

1https://github.com/MISP/misp-taxonomies/blob/master/admiralty-scale/machinetag.json
### admirality-scale taxonomy²

<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>

→ Users can override tag numerical_value

Scoring Indicators: Our solution

\[
\text{score}(\text{Attribute}) = \text{base\_score}(\text{Attribute, Model}) \cdot \text{decay}(\text{Model, time})
\]

- **base\_score(\text{Attribute, Model})**
  - Initial score of the \text{Attribute} only considering the context (\text{Attribute's type, Tags})

- **decay(\text{Model, time})**
  - Function composed of the \text{lifetime} and \text{decay speed}
  - Decreases the base\_score over time
Scoring Indicators: Our solution

\[ \text{score}(\text{Attribute}) = \text{base\_score}(\text{Attribute, Model}) \cdot \text{decay}(\text{Model, time}) \]
Current implementation in MISP
### Implementation in MISP: Event/view

- **Decay score toggle button**
  - Shows Score for each *Models* associated to the *Attribute* type
Implementation in MISP: API result

```
/attributes/restSearch

"Attribute": [
  {
    "category": "Network activity",
    "type": "ip-src",
    "to_ids": true,
    "timestamp": "1565703507",
    [...]
    "value": "8.8.8.8",
    "decay_score": [
      {
        "score": 54.475223849544456,
        "decayed": false,
        "DecayingModel": {
          "id": "85",
          "name": "NIDS Simple Decaying Model"
        }
      }
    ],
    [...]
  }
]```

Implementation in MISP: Objectives

- **Automatic scoring** based on default values
- **User-friendly UI** to manually set *Model* configuration (lifetime, decay, etc.)
- **Simulation** tool
- Interaction through the **API**
- Opportunity to create your **own** formula or algorithm
Implementation in MISP: Models definition

\[ \text{score} = \text{base_score} \cdot \left( 1 - \left( \frac{t}{\tau} \right)^\frac{1}{\delta} \right) \]

Models are an instanciation of the formula with configurable parameters:

- Parameters: lifetime, decay_rate, threshold
- base_score computation
- default base_score
- associate Attribute types
- formula
- creator organisation
Two types of model are available

- **Default Models**: Created and shared by the community. Coming from `misp-decaying-models` repository³. → Not editable

- **Organisation Models**: Created by a user on MISP  
  ▶ Can be hidden or shared to other organisation  
  → Editable

³https://github.com/MISP/misp-decaying-models.git
### Decaying Models

<table>
<thead>
<tr>
<th>ID</th>
<th>Organization</th>
<th>Usable to everyone</th>
<th>Name</th>
<th>Description</th>
<th>Parameters</th>
<th>Formula</th>
<th># Assigned Types</th>
<th>Version</th>
<th>Enabled</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>1</td>
<td>✓</td>
<td>Phishing model</td>
<td>Simple model to rapidly decay phishing website.</td>
<td>`{</td>
<td>Polynomial</td>
<td>9</td>
<td>1</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

```json
"lifetime": 3,
"decay_speed": 2.3,
"threshold": 30,
"default_base_score": 80,
"base_score_config": {
  "estimative_language": 0.5,
  "phishing": 0.5
}
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Organization</th>
<th>Usable to everyone</th>
<th>Name</th>
<th>Description</th>
<th>Parameters</th>
<th>Formula</th>
<th># Assigned Types</th>
<th>Version</th>
<th>Enabled</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>1</td>
<td>🔴</td>
<td>NIDS Simple Decaying Model</td>
<td>Simple decaying model for Network Intrusion Detection System (NIDS).</td>
<td>`{</td>
<td>Polynomial</td>
<td>13</td>
<td>1</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

```json
"lifetime": 120,
"decay_speed": 2,
"threshold": 30,
"default_base_score": 80,
"base_score_config": {
  "estimative_language": 0.25,
  "priority-level": 0.25,
  "retention": 0.25,
  "targeted-threat-index": 0.125,
  "false-positive": 0.125
}
```

Page 1 of 1, showing 2 records out of 2 total, starting on record 1, ending on 2

**Standard CRUD operations:** View, update, add, create, delete, enable, export, import
Configure models: Create, modify, visualise, perform mapping
IMPLEMENTATION IN MISP: base_score TOOL

![Implementation in MISP: base_score Tool](image)

- **Search Taxonomy**: Default base_score 80
- **Taxonomies**
  - admiralty-scale
    - source-reliability
      - information-credibility
        - priority-level
          - priority-level
            - retention
              - estimative-language
                - likelihood-probability
                - confidence-in-analytic-judgment
              - msp
                - confidence-level
                - threat-level
                - automation-level
            - phishing
              - state
              - psychological-acceptability
  - Excluded

![Example]

- **Example**
  - **Attribute**
    - Tags: admiralty-scale:information-credibility="5"
    - Base score: 0.0
  - **Attribute 2**
    - Tags: priority-level:baseline-minor admiralty-scale:source-reliability="d" admiralty-scale:information-credibility="2"
    - Base score: 38.2
  - **Attribute 3**
    - Tags: priority-level:severe admiralty-scale:information-credibility="2"
    - Base score: 84.6

- **Computation steps**
  - **Tag**
    - **Computation**
      - **Priority level**
        - **Tag**: priority-level:baseline-minor
          - **Eff. Ratio**: 0.46
          - **Value**: 25.00
          - **Result**: 11.02
        - **Tag**: admiralty-scale:source-reliability="d"
          - **Eff. Ratio**: 0.27
          - **Value**: 25.00
          - **Result**: 6.80
Simulate decay on Attributes with different Models
/attributes/restSearch

{
  "includeDecayScore": 1,
  "includeFullModel": 0,
  "excludeDecayed": 0,
  "decayingModel": [85],
  "model Overrides": {
    "threshold": 30
  }
  "score": 30,
}
Creating a new decay algorithm

```php
<?php
include_once 'Base.php';

class Polynomial extends DecayingModelBase {
    public const DESCRIPTION = 'The description of your new decaying algorithm';

    public function computeScore($model, $attribute, $base_score, $elapsed_time)
    {
        // algorithm returning a numerical score
    }

    public function isDecayed($model, $attribute, $score)
    {
        // algorithm returning a boolean stating
        // if the attribute is expired or not
    }

}?>
```
Decaying Models 2.0

- Improved support of *Sightings*
  - False positive *Sightings* should somehow reduce the score
  - Expiration *Sightings* should mark the attribute as decayed

- Potential *Model* improvements
  - Instead of resetting the score to `base_score` once a *Sighting* is set, the score should be increased additively (based on a defined coefficient); thus prioritizing surges rather than infrequent *Sightings*
  - Take into account related *Tags* or *Correlations* when computing score

- Increase *Taxonomy* coverage
  - Users should be able to manually override the numerical_value of *Tags*
Forensic support in MISP

Tools and visualization to support digital forensic expert

Team CIRCL

info@circl.lu

May 19, 2021
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).

¹MISP 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
FORENSIC IMPORT (MISP 2.4.98)

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

CIRCL / Team MISP Project

FIRST workshop
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:

  http://localhost:5000/events/csv/download/false/false/tag1&&tag2&&tag3/Network%20activity/domain
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

/five.osf
/one.osf/zero.osf
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
Why is the search API receiving so much focus?

- 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.
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!
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)
Best Practices in Threat Intelligence

Gather, document, analyse and contextualise intelligence using MISP

CIRCL / Team MISP Project

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

FIRST workshop
Objectives

- Learn how to use MISP to support common OSINT gathering use-cases often used by SOC, CSIRTs and CERTs
  - Use practical exercise examples
  - The exercises are based on practical recent cases to model and structure intelligence using the MISP standard
- Improve the data models available in MISP by exchanging live improvements and ideas
- Be able to share the results to the community at the end of this session

---

1https://gist.github.com/adulau/8c1de48060e259799d3397b83b0eec4f
Cyber threat intelligence (CTI) is a vast concept which includes different concepts, methods, and workflows

- Intelligence is defined differently in the military than in the financial sector than in the intelligence community

- MISP project doesn’t want to lock an organisation or a user into a specific model. Each model is useful depending on the objectives of an organisation

- A set of pre-defined knowledge base or data-models are available and organisations can select (or create) what they need

- During this session, an overview of the most used taxonomies, galaxies, and objects will be described
Overall process of collecting and analysing OSINT
Quality of indicators/attributes are important but tagging and classification are also critical to ensure actionable information.

Organizing intelligence is done in MISP by using tags, which often originate from MISP taxonomy libraries.

The scope can be classification (tlp, PAP), type (osint, type, veris), state (workflow), collaboration (collaborative-intelligence), or many other fields.

MISP taxonomy documentation is readily available²

Review existing practices of tagging in your sharing community, reuse practices, and improve context.

²https://www.misp-project.org/taxonomies.html
When information cannot be expressed in triple tags format (namespace:predicate=value), MISP use Galaxies.

Galaxies contain a huge set of common libraries such as threat actors, malicious tools, tactics, target information, mitigations, and more.

When tagging or adding a Galaxy cluster, tagging at the event level is for the whole event (including attributes and objects). Tagging at the attribute level is for a more specific context.

---

[^3]: [https://www.misp-project.org/galaxy.html](https://www.misp-project.org/galaxy.html)
Words of Estimative Probability\textsuperscript{4} propose clear wording while estimating probability of occurrence from an event. A MISP taxonomy called \texttt{estimative-language}\textsuperscript{5} proposes an applied model to tag information in accordance with the concepts of Estimative Probability.


\textsuperscript{5}https://www.misp-project.org/taxonomies.html
The **Admiralty Scale** (also called the **NATO System**) is used to rank the reliability of a source and the credibility of information.

- A MISP taxonomy called admiralty-scale is available.
- US DoD **JP 2-0, Joint Intelligence** includes an appendix to express confidence in analytic judgments.
- A MISP predicate in estimative-language called confidence-in-analytic-judgment is available.

---


7. [https://www.misp-project.org/taxonomies.html](https://www.misp-project.org/taxonomies.html)


9. [https://www.misp-project.org/taxonomies.html](https://www.misp-project.org/taxonomies.html)
If the information is a **single atomic element**, using a single attribute is preferred

- Choosing an attribute type is critical as this defines the automation/export rule (e.g. `url` versus `link` or `ip-src/ip-dst`?)
- Enabling the IDS (automation) flag is also important, but *when you are in doubt*, don’t set the IDS flag

If the information is **composite** (ip/port, filename/hash, bank account/BIC), using an object is strongly recommended
There are more than 150 MISP object\textsuperscript{10} templates. As an example, at CIRCL, we regularly use the following object templates: file, microblog, domain-ip, ip-port, coin-address, virustotal-report, paste, person, ail-leak, pe, pe-section, registry-key.

\textsuperscript{10}https://www.misp-project.org/objects.html
Use case
A series of OSINT tweets from a security researcher. To structure the thread, the information, and keep a history.

Object to use
The microblog object can be used for Tweets or any microblog post (e.g. Facebook). The object can be linked using followed-by to describe a series of post.
Use case

- A file sample was received by email or extracted from VirusTotal
- A list of file hashes were included in a report
- A hash value was mentioned in a blog post

Object to use

The file object can be used to describe file. It’s usual to have partial meta information such as a single hash and a filename.
Graphical overview of OSINT collection using MISP https://github.com/adulau/misp-osint-collection

MISP objects documentation https://www.misp-project.org/objects.html

MISP taxonomies documentation https://www.misp-project.org/taxonomies.html

MISP galaxy documentation https://www.misp-project.org/galaxy.html
MISP core development hands-on exercise

Building a small nifty feature for the MISP core

CIRCL / Team MISP Project

MISP Threat Sharing

FIRST workshop
Some practical things first...

- If you’d like to take a peak at the main files already implemented: https://github.com/iglocska/misp-dev-training-cheat-sheet
- Full implementation: https://github.com/MISP/MISP/tree/dev_session/app
Let’s try to develop a feature together

- Idea: Users should have the option to set alert filters for the publish alert e-mails
- By default receive all alerts as before
- If a filter is set, check if the alert is interesting for us or not
How to ensure that the feature is useful for the community at large?

- Always try to think in reusable systems instead of fixing a single issue
  - Much higher chance of getting a PR merged if it doesn’t just cover your specific use-case
  - Try to stay two steps ahead, see how your feature can be reused for other tasks
Allow users to set preferences for certain views
For high level users, all the technical details are sometimes wasted
Simply not being interested in certain types of data points
Non-standard MISP deployments (island only MISP instances, etc)
User pre-sets for certain settings
Objectives of the Feature

User should be able to do the following with filter rules:
- set
- get
- remove
- index

Filter rules should be flexible - we do not want to anticipate all possible settings in advance

Ensure that the system is easy to extend and reuse
Before we start with anything...

- Update our MISP instance (git pull origin 2.4)
- Fork github.com/MISP/MISP (via the github interface)
- Add a new remote to our fork:
  - via username/password auth: git remote add my_fork https://github.com/iglocsk/MISP
  - via ssh: git remote add my_fork git@github.com:iglocsk/MISP.git
- Generally a good idea to work on a new branch: git checkout -b dev_exercise
- Enable debug in MISP
### Storage:
- Single key/value table for all settings
- Each user should be able to set a single instance of a key
- Values could possibly become complex, let’s use JSON!
- Add timestamping for traceability
- Consider which fields we might want to look-up frequently for indexing
The database changes we need

- The table structure:
  - id int(11) auto increment //primary key
  - key varchar(100) //add index!
  - value text //json
  - user_id int(11) //add index!
  - timestamp int(11) //add index!

- Tie it to into the upgrade system
  (app/Model/AppModel.php)

- Test our upgrade process! Check the output in the audit logs
Outline of the changes needed:

- New Controller (UserSettingsController.php)
- New Model (UserSetting.php)
- New Views (setSetting, index)
- Add new controller actions to ACL
- Update the e-mail alert system to use the functionality
Create the new Model skeleton

- location: /var/www/MISP/app/Model/UserSetting.php
- Create basic skeleton
- Add model relationships (hasMany/BelongsTo)
- Use the hooking functionality to deal with the JSON field (beforeSave(), beforeFind())
- Add a function that can be used to check if a user should get an alert based on filters (checkPublishFilter())
- Add a function to check if a user can access/modify a setting (checkAccess())
Create the Controller skeleton

- location: /var/www/MISP/app/Model/UserSetting.php
- Create basic skeleton
- Set pagination rules
- Define CRUD functions (exceptionally, we diverge here from the norm)
  - setSetting()
  - getSetting()
  - index()
  - delete()
Start with an API only approach at first

- `setSetting()`:
  - Accepted methods: ADD / POST
  - Separate handling of API / UI
  - POST should create/update an entry
  - GET should describe the API
getSetting():

- Accepted methods: GET
- Retrieves a single setting based on either ID or setting key and user_id
- Encode the data depending on API/UI

- Accepted methods: GET
- List all settings
- Filter user scope on demand
- Filter available scopes based on role
delete():

- Accepted methods: POST / DELETE
- Deletes a single entry based on ID or setting key
- Encode the data depending on API/UI
Add the ACL functionalities

- Tie functions into checkAccess():
  - Check if user is allowed to execute actions and throw exceptions if not
  - Add it to: setSetting() / getSetting() / delete()

- Consider that:
  - Site admins have full reign
  - Org admins can manage their own users
  - Everyone else can self-manage
Test the functionalities

- Use the REST client
- Expectations
  - GET on /setSetting and /delete describing our endpoints
  - POST /setSetting with "key": "publish_filter", "value": "Event.tags":"%sofacy%" should return newly added or modified filter
  - GET on /index should list our entries, GET on /getSetting should show an individual entry
  - DELETE on /delete should delete the entry
Start adding the UI components

We now have a rudimentary CRUD, let’s add some simple UI views

- setSetting as a simple form
- index should use the parametrised generators (IndexTable)
- Add both views to the menu systems (side-menu, global menu)
- Don’t forget about sanitisation and translations!
Add the checkPublishFilter() function to the e-mailing

- Trace the code path of the e-mail sending to understand the process
- Decide on the best place to inject our check
- Don’t break the flow of the process!
- What do we have access to at this point? What format are they in?
Test if our code works correctly

- Do we see any notices / errors?
- Is our code easily accessible?
- Consider other roles! Can users/org admins do things we don’t want them to do?
- Is our code-base breaking the default behaviour?
- Is our update script working as expected?
**Push our code to our fork and create a pull request**

- git status to check what changed / got added
- git add /path/to/file to add files we want to commit
- git commit (format: is "new/fix/chg: [topic] My description"
- git push my_fork
- Create pull request from the github interface
- Wait for Travis to run, update the code if needed
MISP restSearch module development

Building a simple export module for the core

CIRCL / Team MISP Project

FIRST workshop
Building a native restSearch export

Similar in scope to an export module of the MISP modules system

Pros:

▶ Can be used for composited data coming from a filtered query
▶ Fast, native approach
▶ Can be built to support several scopes (events, attributes, sightings)

Cons...
Building a native RESTSearch export

- Similar in scope to an export module of the MISP modules system
- Pros:
  - Can be used for compositied data coming from a filtered query
  - Fast, native approach
  - Can be built to support several scopes (events, attributes, sightings)
- Cons...
So how does restSearch work?

- Standardised way of collecting **parameters**
- Using the parameters, a loop is started to **chunk and gradually build** our export data
- The chunk size depends on memory envelopes
- Each chunk is **converted piece by piece**...
- ... and subsequently are concatenated into a temporary file
- Once no more elements are left, the file is sent in the response
The export modules handle 5 tasks:

- Pass **meta-information** back to restSearch on the export format itself
- Add a **start segment** to the exported data
- Do the actual **conversion** from MISP’s internal format to the desired export format
- Provide a **separator** for data chunks
- Have a **closing segment** for the returned data, based on the format’s conventions
Our little training module: Nibbler, the ever hungry IDS/IPS
Nibbler

- Simplistic tool with its **own proprietary format**
- Meant to mimic a typical **in-house tool**
- Lightweight scope, for simplicity’s sake
- **pipe separated values**
- **VALUE | TYPE | DESCRIPTION | REFERENCE | ACTION**
Rules can be prepended by comments, each comment line starting with #

Some characters have to be escaped in some custom, crazy ways

- linebreaks: ##LINEBREAK##
- commas: ##COMMA##
- pipes: ##PIPE##
**Nibbler Format**

- **Value**: The actual indicator value
- **Type**: The format of the indicator
- **Description**: A quick description for analysts investigating the alert, why is this relevant
- **Reference**: A backreference that the analyst can use to find out more about the alert
- **Action**: What should Nibbler do if it trips over the value?
Supported types

- IP
- Domain
- Hostname
- MD5
- SHA1
- SHA256
- Filename
**Supported values**

- **ALERT** - default behaviour, create an alert.
- **BLOCK** - block the action outright. Only set if the tag `nibbler:block` is present.
Though we have types to map from MISP, in some cases several types map to a Nibbler type.

We’ve created a rough mapping (this is probably the most difficult task) in advance.

Some MISP types map to a Nibbler type directly.

**Composite** MISP types map to 2 Nibbler types each.
Mapping the types to MISP

- ip-dst :: IP
- ip-src :: IP
- domain :: Domain
- domain|ip :: Domain, IP
- hostname :: Hostname
- md5 :: MD5
- sha1 :: SHA1
- sha256 :: SHA256
- filename|md5 :: Filename, MD5
- malware-sample :: Filename, MD5
- filename|sha1 :: Filename, SHA1
- filename|sha256 :: Filename, SHA256
<?php
class NibblerExport
{
    public $additional_params = array();
    public function handler(
        $data, $options = array()
    ) {}
    public function header(
        $options = array()
    ) {}
    public function footer() {}
    public function separator() {}
}
public $additional_params = array('flatten' => 1);
private $__mapping = array(
    'ip-dst' => 'IP',
    'ip-src' => 'IP',
    'domain' => 'Domain',
    'domain|ip' => ['Domain', 'IP'],
    'hostname' => 'Hostname',
    'md5' => 'MD5',
    'sha1' => 'SHA1',
    'sha256' => 'SHA256',
    'filename|md5' => array('Filename', 'MD5'),
    'malware-sample' => array('Filename', 'MD5'),
    'filename|sha1' => array('Filename', 'SHA1'),
    'filename|sha256' => array('Filename', 'SHA256')
);
public function header($options = array())
{
    return sprintf(
        "# Nibbler rules generated by MISP at %s
        date('Y-m-d H:i:s')
    );
}

public function footer()
{
    return "\n";
}

public function separator() {
    return "\n";
}
public function handler($data, $options = array())
{
    if ($options['scope'] === 'Attribute') {
        $data['Attribute'] ['AttributeTag'] = $data['AttributeTag'];
        return $this->_convertAttribute($data['Attribute'], $data['Event']);
    }
    if ($options['scope'] === 'Event') {
        $result = array();
        foreach ($data['Attribute'] as $attribute) {
            $temp = $this->_convertAttribute($attribute, $data['Event']);
            if ($temp) $result[] = $temp;
        }
        return implode($this->separator(), $result);
    }
    return '';  
}
private function __convertAttribute($attribute, $event) {
    if (empty($this->__mapping[$attribute['type']])) {
        // mapping not found – invalid type for nibbler
        return '';;
    }
    if (is_array($this->__mapping[$attribute['type']])) {
        // handle mappings for composites – slide
    } else {
        // handle simple mappings – slide
    }
    // return 1 or 2 lines, separated by separator()
    return implode($this->separator(), $result);
}
$result[] = sprintf(
    '%s%s%s%s%s%s',
    $this->__escapeSpecialChars($attribute['value']),
    $this->__mapping[$attribute['type']],
    $event['uuid'],
    $this->__escapeSpecialChars($event['info']),
    'ALERT'
);
$attribute['value'] = explode( ' | ', $attribute['value'] )
;
foreach ( array(0,1) as $part ) {
    $result[] = sprintf(
        '%s%s%s%s%s',
        $this->__escapeSpecialChars($attribute['value'][$part]),
        $this->__mapping[$attribute['type'][$part],
        $event['uuid']],
        $this->__escapeSpecialChars($event['info']),
        'ALERT'
    );
}

HANDLING THE CASE FOR COMPOSITES
private function __convertAttribute($attribute, $event) {
    if (empty($this->__mapping[$attribute['type']])) return '';
    $result = array();
    $attributes = array();
    if (is_array($this->__mapping[$attribute['type']])) {
        $attribute['value'] = explode('|', $attribute['value']);
        foreach (array(0,1) as $part) {
            $result[] = sprintf(
                '%s%s%s%s%s',
                $this->__escapeSpecialChars($attribute['value'][$part]),
                $this->__mapping[$attribute['type']][$part],
                '/events/view/ . $event['uuid'],
                $this->__escapeSpecialChars($event['info']),
                $this->__decideOnAction($attribute['AttributeTag'])
            );
        }
    } else {
        $result[] = sprintf(
            '%s%s%s%s%s',
            $this->__escapeSpecialChars($attribute['value']),
            $this->__mapping[$attribute['type']],
            '/events/view/ . $event['uuid'],
            $this->__escapeSpecialChars($event['info']),
            $this->__decideOnAction($attribute['AttributeTag'])
        );
    }
    return implode($this->separator(), $result);
}
private function __decideOnAction($attributeTags) {
    foreach ($attributeTags as $attributeTag) {
        if (
            $attributeTag['Tag']['name'] == 'nibbler:block'
        ) {
            return 'BLOCK';
        }
    }
    return 'ALERT';
}
private function __escapeSpecialChars($value) {
    $value = preg_replace("/
", "##LINEBREAK##", $value);
    $value = preg_replace("/, /", "##COMMA##", $value);
    $value = preg_replace("/\|/", "##PIPE##", $value);
    return $value;
}
Modifying the MISP core to know about the export module

- The **models** that we are targeting by scope (Event, Attribute) need to be updated
- They are located in `/var/www/MISP/app/Model/`
- The global variable `$validFormats` houses all mappings
- Simply add a new line such as the following:
  - `'nibbler' => array('nibbler', 'NibblerExport', 'nibbler')`
Let us test the module!

- Use the **rest client** to test it conveniently
- Both the event and attribute level restSearch function should work
- Simply set the **returnFormat** to nibbler, which should also show up as a valid export format
REST CLIENT

HTTP method to use
- POST

Relative path to query
- /events/restSearch

- Use full path - disclose my apikey
- Show result

HTTP headers
- Authorization: ArSxnHf20foSapnOSyxfrljMdl9oLDnmqvHK97q
- Accept: application/json
- Content-Type: application/json

HTTP body
```
{
    "returnFormat": "nibbler",
    "page": 1,
    "limit": 4,
    "type": [
        "ip-dst",
        "ip-src",
        "domain\ip",
        "hostname",
        "domain"
    ]
}
```

Run query
MISP - Galaxy 2.0

Method for sharing threat intelligence

Team CIRCL

info@circl.lu

May 19, 2021
Present the features available for Sharing *galaxy clusters*

Look at the internals of what changed in the datamodel and MISP’s behaviors
Galaxy 2.0 introduces various new features for Galaxies and their Clusters allowing:

- Creation of **custom Clusters**
- **ACL** on Clusters
- **Connection** of Clusters via Relations
- **Synchronization** to connected instances.
- **Visualization** of forks and relationships
Default Galaxy cluster

- Coming from the misp-galaxy repository
- Cannot be edited
  - Only way to provide modification is to modify the stored JSON or to open a pull request
  - Are not synchronized
  - Source of trust

- Restrictions propagate to their children (Galaxy cluster elements, Cluster relationships)

Custom Galaxy cluster

- Can be created via the UI or API
- Belongs to an organisation
  - Fully editable
  - Are synchronized

¹https://github.com/MISP/misp-galaxy
Clusters and Relations can be edited.

- New Clusters fields
  - distribution, sharing_group_id
  - org_id, orgc_id
  - locked, published, deleted
  - default
    - Clusters coming from the misp-galaxies repository are marked as default
    - Not synchronized

- Same purpose as Event's locked field

- extends_uuid
  - Point to the Cluster that has been forked

- extends_version
  - Keep track of the Cluster version that has been forked
- **Role** `perm_galaxy_editor`
- Relations also have a distribution and can have *Tags*
- Synchronization servers have 2 new flags
  - `pull_galaxy_clusters`
  - `push_galaxy_clusters`
- Clusters blocklist
Features in depth: CRUD

- Standard CRUD
- Soft and Hard deletion
- Publishing
- Update forked cluster to keep it synchronized with its parent
- ACL on the Cluster itself, not on its tag
  - `misp-galaxy:galaxy-type="cluster UUID"`
  - `misp-galaxy:mitre-attack-pattern="e4932f21-4867-4de6-849a-1b11e48e2682"`
Features in depth: Visualization

Tree view of forked Clusters
Features in depth: Visualization

Tree and network views for Relations between Clusters
Features in depth: Visualization

Tree and network views for Relations between Clusters

Source UUID | Relationship type | Target UUID | Distribution
---|---|---|---
Bed61090-f098-4878-8b7e-2d80| dropped | | All communities

Tags

+ Add relationship
Galaxy Cluster Elements

Hasn’t been touched: Still a key-value stored. But new feature have been added²

Tabular view

- Allows you to browse **cluster elements** like before

²Will be included in next release
JSON view

- Allows you to visualisation **cluster element** in a JSON structure
- Allows you to convert any JSON into **cluster elements** enabling searches and correlations
Synchronization in depth

Has its own synchronization mechanism which can be enabled with the `pull_galaxy_cluster` and `push_galaxy_cluster` flags.

- **Pull All**: Pull all remote Clusters (similar to event’s pull all)
- **Pull Update**: Update local Clusters (similar to event’s pull update)
- **Pull Relevant**: Pull missing Clusters based on local Tags
- **Push**: Triggered whenever a Cluster is published or via standard push
An Introduction to Cybersecurity Information Sharing

MISP - Threat Sharing

CIRCL / Team MISP Project

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

FIRST workshop
Content of the presentation

- Data sharing in MISP
- Data models for the Data layer
- Data models for the Context layer
Layers of data model

Data layer
- The raw data itself as well as element to link them together
- Indicators, Observables and means to contextually link them
- MISP terminology: Event, Attributes, misp-objects, ...

Context layer
- As important as the data layer, allow triage, false-positive management, risk-assessment and prioritisation
- Latches on the data layer, usually referencing threat intelligence, concepts, knowledge base and vocabularies
- Tags, Taxonomies, Galaxies, ...
Data sharing in MISP
Sharing in MISP: Distribution

MISP offers granular distribution settings:

- Organisation only
- This community
- Connected communities
- All communities
- Distribution lists - aka **Sharing groups**

At multiple levels: **Events, Attributes, Objects** (and their **Attributes**) and **Galaxy-clusters**
Sharing in MISP: Distribution

Sharing group

This community
Connected communities
All communities
Data Layer
Data layer

- **Events** are encapsulations for contextually linked information
- **Attributes** are individual data points, which can be indicators or supporting data.
- **Objects** are custom templated Attribute compositions
- **Object references** are the relationships between individual building blocks
- **Shadow Attributes/Proposal** are suggestions made by users to modify an existing attribute
- **Sightings** are a means to convey that a data point has been seen
- **Event reports** are supporting materials for analysts to describe *events, processes, etc*
**Data layer: Events**

**Events** are encapsulations for contextually linked information.

<table>
<thead>
<tr>
<th>IoT malware - Gafgyt.Gen28 (active) - 20190220 - 20190222</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event ID</td>
</tr>
<tr>
<td>UUID</td>
</tr>
<tr>
<td>Creator org</td>
</tr>
<tr>
<td>Owner org</td>
</tr>
<tr>
<td>Creator user</td>
</tr>
<tr>
<td>Tags</td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Threat Level</td>
</tr>
<tr>
<td>Analysis</td>
</tr>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Info</td>
</tr>
<tr>
<td>Published</td>
</tr>
<tr>
<td>#Attributes</td>
</tr>
<tr>
<td>First recorded change</td>
</tr>
<tr>
<td>Last change</td>
</tr>
<tr>
<td>Modification map</td>
</tr>
<tr>
<td>Sightings</td>
</tr>
</tbody>
</table>
DATA LAYER: EVENT BUILDING BLOCKS - BASE

Event

{ Data }

seven.osf	hree.osf/eight.osf
Data layer: Events

```json
{
    "date": "2019-02-20",
    "info": "IoT malware – Gafgyt.Gen28 (active)",
    "uuid": "5c6d21e5-bb60-47b7-b892-42e6950d2111",
    "analysis": "2",
    "timestamp": "1602315388",
    "distribution": "3",
    "sharing_group_id": "0",
    "threat_level_id": "3",
    "extends_uuid": "",
    "Attribute": [...],
    "Object": [...],
    "EventReport": [...],
    "Tag": [...],
    "Galaxy": [...]
}
```
**Data layer: Attributes**

**Attributes** are individual data points, indicators or supporting data.
Data layer: Event building blocks - Raw data

Event

{ Data }

Attribute

Attribute

Attribute
Data layer: Attributes

```json
1 {  
    "type": "url",
    "category": "Network activity",
    "to_ids": true,
    "uuid": "5c6d24bd-d094-4dd6-a1b6-4fa3950d2111",
    "event_id": "178",
    "distribution": "5",
    "sharing_group_id": "0",
    "timestamp": "1550656701",
    "comment": "Delivery point for the malware",
    "object_id": "0",
    "object_relation": null,
    "first_seen": null,
    "last_seen": null,
    "value": "ftp://185.135.80.163/",
    "Tag": [...]
    "Galaxy": [...]
}  
```
**Data layer: MISP Objects**

**Objects** are custom templated Attribute compositions

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>filename</td>
<td>putty.exe</td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>md5</td>
<td>b6c12d88eeb910784d75a5e4df954001</td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>sha1</td>
<td>5ef9515e8fd92a254dd2dcd9c4b50afa8007b8f</td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>sha256</td>
<td>81ded31987304676134138705fc1c21188ad7127ed6b77a6551a693194485e</td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>sha512</td>
<td>e174ecf4ffdb36d30c2cc66b37f82877d421244e924d5c9f39f2e0f37d85332bd7d107d5ac5bd19cb7ffdc6bb9c505d488faa30664ef510f52f970c163cc76</td>
</tr>
<tr>
<td>2018-03-27</td>
<td>Payload delivery</td>
<td>malware-sample</td>
<td>putty.exe</td>
</tr>
</tbody>
</table>
Data layer: Event building blocks - Data composition
```json
{
    "name": "elf-section",
    "meta-category": "file",
    "description": "Object describing a sect...",
    "template_uuid": "ca271f32-1234-4e87-b240-6b6e882de5de",
    "template_version": "4",
    "uuid": "ab5f0c85-5623-424c-bc03-d79841700d74",
    "timestamp": "1550655984",
    "distribution": "5",
    "sharing_group_id": "0",
    "comment": "",
    "first_seen": null,
    "last_seen": null,
    "ObjectReference": [],
    "Attribute": [...]
}
```
Object references are the relationships between individual building blocks.
```json
1 {
2     "uuid": "5c6d21f9-0384-4bd2-b256-40de950d2111",
3     "timestamp": "1602318569",
4     "object_id": "1024",
5     "source_uuid": "23275e05-c202-460e-aadf-819c417fb326",
6     "referenced_uuid": "ab5f0c85-5623-424c-bc03-d79841700d74",
7     "referenced_type": "1",
8     "relationship_type": "included-in",
9     "comment": "Section 0 of ELF"
10 }
```
Data layer: Event building blocks - Context

Event

{ Data }

Attribute

Attribute

Context

Object

Object Attribute

Object Attribute

Object Attribute

Reference
Data layer: Sightings

Sightings are a means to convey that a data point has been seen.

```
1 { 
2     "org_id": "1", 
3     "date_sighting": "1573722432", 
4     "uuid": "5dcd1940-5de8-4462-93dd-12a2a5e38e14", 
5     "source": "", 
6     "type": "0", 
7     "attribute_uuid": "5da97b59-9650-4be2-9443-2194a5e38e14" 
8 } 
```
Data layer: Event reports

Event reports are supporting data for analysis to describe events, processes, etc.
DATA LAYER: EVENT BUILDING BLOCKS - COLLABORATION & INTELLIGENCE
```json
{
  "uuid": "076e240b-5a76-4a8b-9eab-cfff551993dd",
  "event_id": "2127",
  "name": "Event report (1607362986)",
  "content": "...",
  "distribution": "5",
  "sharing_group_id": "0",
  "timestamp": "1607362986"
}
```
DATA LAYER: EVENT BUILDING BLOCKS - FULL
Data layer: Combining the data layer

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

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

Correlation

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

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

Proposal
- Value
- Category
- Type
- IDS
- Comment
- Org
CONTEXT LAYER
Context layer

- **Tags** are free-text labels attached to events/attributes and can come from **Taxonomies**
  - Android Malware, C2, ...
- **Taxonomies** are a set of common classification allowing to express the same vocabulary among a distributed set of users and organisations
  - tlp:green, false-positive: risk="high", admiralty-scale: information-credibility="2"
Context layer

- **Galaxies** are container composed of **Galaxy-clusters** that belongs to the same family
  - Similar to what **Events** are to **Attributes**
  - Country, Threat actors, Botnet, ...

- **Galaxy-clusters** are knowledge base items coming from **Galaxies**.
  - Basically a taxonomy with additional meta-information
  - `misp-galaxy:threat-actor="APT 29"`, `misp-galaxy:country="luxembourg"`
Simple free-text labels

```json
1 {  
2     "name": "Android malware",  
3     "colour":="#22681c",  
4     "exportable": true,  
5     "numerical_value": null,  
6 }
```
Simple label standardised on common set of vocabularies

<table>
<thead>
<tr>
<th>Tag</th>
<th>Events</th>
<th>Attributes</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflow:state=&quot;complete&quot;</td>
<td>11</td>
<td>0</td>
<td>workflow:state=&quot;complete&quot;</td>
</tr>
<tr>
<td>workflow:state=&quot;draft&quot;</td>
<td>0</td>
<td>0</td>
<td>workflow:state=&quot;draft&quot;</td>
</tr>
<tr>
<td>workflow:state=&quot;incomplete&quot;</td>
<td>55</td>
<td>10</td>
<td>workflow:state=&quot;incomplete&quot;</td>
</tr>
<tr>
<td>workflow:state=&quot;ongoing&quot;</td>
<td>0</td>
<td>0</td>
<td>workflow:state=&quot;ongoing&quot;</td>
</tr>
</tbody>
</table>
```
1 {
2   "Taxonomy": {
3     "namespace": "admiralty-scale",
4     "description": "The Admiralty Scale or Ranking (also called the NATO System)...",
5     "version": "6",
6     "exclusive": false,
7   },
8   "entries": [
9     {
10    "tag": "admiralty-scale:information-credibility=\"1\"",
11    "expanded": "Information Credibility: Confirmed by other sources",
12    "numerical_value": 100,
13    "exclusive_predicate": true,
14     },
15     ...
16   ]
17 }
```
Collections of **galaxy clusters**

**Threat Actor galaxy**

- **Galaxy ID**: 8
- **Name**: Threat Actor
- **Namespace**: misp
- **UUID**: 698774c7-8222-4204-9178-8d6e406a3a3
- **Description**: Threat actors are characteristics of malicious actors (or adversaries) representing a cyber attack threat including presumed intent and historically observed behaviour.
- **Version**: 3

**Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>Published</th>
<th>Value</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>APT 29</td>
<td></td>
<td>Dukes, Group 100, Cozy Duke, CozyDuke, CozyBear, CozyCob, CozyC, Cozy, CozyCar, Cozy, Office favourite</td>
</tr>
</tbody>
</table>
**CONTEXT LAYER: GALAXY CLUSTERS**

Knowledge base items including a description, links, synonyms, meta-information and relationships

---

**Threat Actor :: APT 29**

<table>
<thead>
<tr>
<th>Cluster ID</th>
<th>2805</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>APT 29</td>
</tr>
<tr>
<td>Parent Galaxy</td>
<td>Threat Actor</td>
</tr>
<tr>
<td>Description</td>
<td>A 2015 report by F-Secure describe APT29 as: 'The Dukes are a well-resourced, highly dedicated and organized cyberespionage group that we believe has been working for the Russian Federation...'</td>
</tr>
</tbody>
</table>

**Published No**

<table>
<thead>
<tr>
<th>Default</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>100</td>
</tr>
<tr>
<td>UUID</td>
<td>b20567c0-00b9-482e-bb1c-c771daa3f28a</td>
</tr>
<tr>
<td>Collection UUID</td>
<td>7cdef3f317-a673-4474-84ec-4f1754947823</td>
</tr>
<tr>
<td>Source</td>
<td>MISP Project</td>
</tr>
<tr>
<td>Authors</td>
<td>Alexandre Dulaunoy, Florian Roth, Thomas Schreck, Timo Steffens, Various</td>
</tr>
<tr>
<td>Distribution</td>
<td>All communities</td>
</tr>
<tr>
<td>Owner Organisation</td>
<td>MISP</td>
</tr>
<tr>
<td>Creator Organisation</td>
<td>MISP</td>
</tr>
<tr>
<td>Connector tag</td>
<td>misp-galaxy:threat-actor=&quot;APT 29&quot;</td>
</tr>
<tr>
<td>Events</td>
<td>0</td>
</tr>
<tr>
<td>Forked From</td>
<td></td>
</tr>
<tr>
<td>Forked By</td>
<td></td>
</tr>
</tbody>
</table>
CONTEXT LAYER: GALAXY CLUSTERS

Galaxy cluster elements: Tabular view

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribution-confidence</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>cfr-suspected-state-sponsor</td>
<td>Russian Federation</td>
<td></td>
</tr>
<tr>
<td>cfr-suspected-victims</td>
<td>United States</td>
<td></td>
</tr>
<tr>
<td>cfr-suspected-victims</td>
<td>China</td>
<td></td>
</tr>
<tr>
<td>cfr-suspected-victims</td>
<td>New Zealand</td>
<td></td>
</tr>
</tbody>
</table>

Galaxy cluster elements: JSON view

```
{
  "attribution-confidence": [
    "50"
  ],
  "cfr-suspected-state-sponsor": [
    "Russian Federation"
  ],
  "cfr-suspected-victims": [
    "United States",
    "China",
    "New Zealand",
    "Ukraine"
  ]
}
```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{</td>
</tr>
<tr>
<td>2</td>
<td>&quot;uuid&quot;: &quot;5eda0a53-1d98-4d01-ae06-40da0a00020f&quot;,</td>
</tr>
<tr>
<td>3</td>
<td>&quot;type&quot;: &quot;fellowship-characters&quot;,</td>
</tr>
<tr>
<td>4</td>
<td>&quot;value&quot;: &quot;Aragorn wielding Anduril&quot;,</td>
</tr>
<tr>
<td>5</td>
<td>&quot;tag_name&quot;: &quot;misp-galaxy:fellowship-characters=&quot;c3fe907a-6a36-4cd1-9456-dcdf35c3f907&quot;&quot;,</td>
</tr>
<tr>
<td>6</td>
<td>&quot;description&quot;: &quot;The Aragorn character wielding Anduril&quot;,</td>
</tr>
<tr>
<td>7</td>
<td>&quot;source&quot;: &quot;Middle-earth universe by J. R. R. Tolkien&quot;,</td>
</tr>
<tr>
<td>8</td>
<td>&quot;authors&quot;: null,</td>
</tr>
<tr>
<td>9</td>
<td>&quot;version&quot;: &quot;1591347795&quot;,</td>
</tr>
<tr>
<td>10</td>
<td>&quot;distribution&quot;: &quot;0&quot;,</td>
</tr>
<tr>
<td>11</td>
<td>&quot;sharing_group_id&quot;: null,</td>
</tr>
<tr>
<td>12</td>
<td>&quot;default&quot;: false,</td>
</tr>
<tr>
<td>13</td>
<td>&quot;extends_uuid&quot;: &quot;5eda0117-1e14-4b0a-9e26-34aff331dc3b&quot;,</td>
</tr>
<tr>
<td>14</td>
<td>&quot;extends_version&quot;: &quot;1591345431&quot;,</td>
</tr>
<tr>
<td>15</td>
<td>&quot;GalaxyElement&quot;: [...],</td>
</tr>
<tr>
<td>16</td>
<td>&quot;GalaxyClusterRelation&quot;: [...]</td>
</tr>
<tr>
<td>17</td>
<td>}</td>
</tr>
</tbody>
</table>
MISP integrates MITRE’s Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) and similar Galaxy Matrix.

MISP terminology of these matrixes: Galaxy Matrix.
{ "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 }
{  
  "description": "DoS or overload of party/campaign registration, causing them to miss the deadline",
  "meta": {  
    "date": "March 2018.",
    "kill_chain": [ "Define in which column the cluster should be placed
        "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.

```json
"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"
]
```
Both layers: Combining everything
Supported by the grant 2018-LU-IA-0148

Co-financed by the European Union
Connecting Europe Facility
Visualise all the things

Building dashboard widgets for MISP

CIRCL / Team MISP Project

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

FIRST workshop
■ User configurable simple dashboard interface
■ **Visualise, aggregate** and **track** data important to you
■ Brand new feature, still undergoing reworks
The internals of a widget

- **Backend** for the widget, full access to all MISP internals
- **Load, convert, format** to be represented via view widgets
- **Widget metadata** - size, name, description, behaviours
- Only main function required to be implemented: `handler()`
- Optional: `checkPermissions()` for ACL
- Accepts **user configuration** for which a template can be provided

- Located in `/var/www/MISP/app/Lib/Dashboard/`
- Custom widgets can be placed in `/var/www/MISP/app/Lib/Dashboard/Custom/`
View files are included by default and reusable
Currently we have a small but growing list of views
  ▶ BarChart
  ▶ SimpleList
  ▶ WorldMap
Converts the data passed by the Widget logic to HTML
Located in
/var/www/MISP/view/Elements/dashboard/Widgets/
Widgets can additionally be tied to certain behaviours:

- **Caching**
  - Executions of the widget logic are cached
  - Separate caches for each organisation in addition to site admins
  - Cache duration is controlled by the widget logic

- **Refresh**
  - Widgets can be set to refresh after x seconds

- Both of these should be used with special care in regards to the use of system resources
Let’s start with a skeleton

Create `/var/www/MISP/app/Lib/Dashboard/Custom/WhoamiWidget.php`

MISP will parse anything ending with `Widget.php` in this directory
<?php

class MispWhoamiWidget
{
    public $title = 'Whoami';
    public $render = 'SimpleList';
    public $width = 2;
    public $height = 2;
    public $params = array();
    public $description = 'Shows information about the currently logged in user.';
    public $cacheLifetime = false;
    public $autoRefreshDelay = 3;

    public function handler($user, $options = array())
    {
        $data = array();
        return $data;
    }
}
**$title**: The name of the widget

**$description**: A description of the widget

**$render**: The view element to use in rendering the widget

**$width & $height**: Default relative dimensions

**$params**: Configuration array with explanations for each key

**$cacheLifetime**: The lifetime of the caches in seconds (false disables it)

**$autoRefreshDelay**: The time in seconds between each refresh (false disables it)
```php
public function handler($user, $options = array())
{
    $this->Log = ClassRegistry::init('Log');
    $entries = $this->Log->find('all', array(
        'recursive' => -1,
        'conditions' => array(
            'action' => 'login', 'user_id' => $user['id']
        ),
        'order' => 'id desc',
        'limit' => 5,
        'fields' => array('created', 'ip')
    ));
    foreach ($entries as &$entry) {
        $entry = $entry['Log']['created'] . ' --- ' .
            (empty($entry['Log']['ip']) ? 'IP not logged' : $entry['Log']['ip']);
    }
    return array(
        array('title' => 'Email', 'value' => $user['email']),
        array(
            'title' => 'Role', 'value' => $user['Role']['name']
        ),
        array(
            'title' => 'Organisation',
            'value' => $user['Organisation']['name']
        ),
        array(
            'title' => 'IP', 'value' => $this->SERVER['REMOTE_ADDR']
        ),
        array('title' => 'Last logins', 'value' => $entries)
    );
}
```
Whoami

Email: admin@admin.test
Role: admin
Organisation: ORGNAME
IP: ::1

Last logins:
2020-03-05 06:50:46 --- ::1
2020-03-04 21:35:15 --- IP not logged
2020-03-04 09:34:44 --- IP not logged
2020-03-03 16:58:35 --- IP not logged
2020-03-03 06:49:10 --- IP not logged
Turning data into actionable intelligence
advanced features in MISP supporting your analysts and tools

CIRCL / Team MISP Project

MISP Threat Sharing

FIRST workshop
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.
To give some insight into what sort of an evolution of our various communities’ have gone through as observed over the past 8 years

Show the importance of strong contextualisation...

...and how that can be leveraged when trying to make our data actionable
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.
The initial scope of MISP

- **Extract information** during the analysis process
- Store and **correlate** these datapoints
- **Share** the data with partners
- Focus on technical indicators: IP, domain, hostname, hashes, filename, pattern in file/memory/traffic
- Generate protective signatures out of the data: snort, suricata, OpenIOC
Initial workflow

MISP Threat Sharing

Raw data → MISP Threat Sharing → Signatures
This was both a reflection of our maturity as a community

- Capabilities for **extracting** information
- Capabilities for **utilising** the information
- Lack of **willingness** to share context
- Lack of **co-operation** between teams doing technical analysis/monitoring and threat-intel

The more growth we saw in maturity, the more we tried to match it with our data-model, often against pushback
There were separate factors that made our data-sets less and less useful for detection/defense in general

- **Growth of our communities**
- Distinguish between information of interest and raw data
- **False-positive** management
- TTPs and aggregate information may be prevalent compared to raw data (risk assessment)
- **Increased data volumes** leads to being able to prioritize
Our initial solution

- Allow users to **tag any information** created in MISP
- We wanted to be **lax with what we accept** in terms of data, but be **strict on what we fed to our tools**, with strong filter options
- We had some ideas on how to potentially move forward...
Try to capture different aspects of contextualisation into **normalised values** (threat level, source reliability, etc)

▶ Didn’t scale with needs other than our own
▶ Incorporating new types of contextualisation would mean **the modification of the software**
▶ Getting communities with **established naming conventions** to use anything but their go-to vocabularies was a pipe-dream
▶ Heated arguments over numeric conversions
**Human Creativity**

- We tried an alternate approach instead: Free tagging
  - Result was spectacularly painful, at least 7 different ways to spell tlp:amber
  - No canonisation for common terms lead to tagging ultimately becoming a highly flawed tool for filtering within a sharing community

<table>
<thead>
<tr>
<th>TLP AMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLP:AMBER</td>
</tr>
<tr>
<td>Threat tlp:Amber</td>
</tr>
<tr>
<td>tlp-amber</td>
</tr>
<tr>
<td>tlp::amber</td>
</tr>
<tr>
<td>tlp:amber</td>
</tr>
</tbody>
</table>
How we ended up tackling the issue more successfully

We ended up with a mixed approach, currently implemented by the MISP-taxonomy system

- Taxonomies are *vocabularies* of known tags
- Tags would be in a *triple tag format*
  
  namespace: predicate="value"

- Create your own taxonomies, recipients should be able to use data you tag with them without knowing it at the first place
- Avoid any coding, stick to *JSON*

Massive success, approaching 100 taxonomies

Organisations can solve their own issues without having to rely on us
We were still missing something...

- Taxonomy tags often non self-explanatory
- Example: universal understanding of tlp:green vs APT 28
- For the latter, a single string was ill-suited
- So we needed something new in addition to taxonomies - Galaxies
  - Community driven knowledge-base libraries used as tags
  - Including descriptions, links, synonyms, meta information, etc.
  - Goal was to keep it simple and make it reusable
  - Internally it works the exact same way as taxonomies (stick to JSON)

![Ransomware galaxy](image)
Broadening the scope of what sort of context we are interested in

- **Who** can receive our data? **What** can they do with it?
- Data accuracy, source reliability
- **Why** is this data relevant to us?
- **Who** do we think is behind it, **what tools** were used?
- What sort of **motivations** are we dealing with? Who are the **targets**?
- How can we **block/detect/remediate** the attack?
- What sort of **impact** are we dealing with?
Parallel to the contextualisation efforts: False positive handling

- Low quality / false positive prone information being shared
- Lead to alert-fatigue
- Exclude organisation xy out of the community?
- False positives are often obvious - can be encoded
- **Warninglist system**\(^1\) aims to do that
- Lists of well-known indicators which are often false-positives like RFC1918 networks, ...

\(^1\)https://github.com/MISP/misp-warninglists
Atomic attributes were a great starting point, but lacking in many aspects

- **MISP objects**\(^2\) system
  - Simple templating approach
  - Use templating to build more complex structures
  - Decouple it from the core, allow users to define their own structures
  - MISP should understand the data without knowing the templates
  - Massive caveat: **Building blocks have to be MISP attribute types**
  - Allow relationships to be built between objects

\(^2\)https://github.com/MISP/misp-objects
Supporting specific datamodel
Data ingested by MISP was in a sense frozen in time

We had a creation data, but lacked a way to use the output of our detection

Lead to the introduction of the **Sighting system**

The community could sight indicators and convey the time of sighting

Potentially powerful tool for IoC lifecycle management, clumsy query implementation default
Supporting specific datamodel

Events

- Sightings
  - CIRCL: 2 (2017-03-19 16:17:59)

- Inherit
  - (2/0/0)

- Inherit
  - (0/0/0)

Tags
- Date: 2016-02-24
- Threat Level: High
- Analysis: Initial
- Distribution: Connected communities
  - freetext test

Sighting Details
- No
- MISP: 2
- CIRCL: 2
Making use of all this context

Most obvious goal: Improve the way we query data
- Unified all export APIs
- Incorporate all contextualisation options into API filters
- Allow for an on-demand way of excluding potential false positives
- Allow users to easily build their own export modules feed their various tools
Example Query

```json
/attributes/restSearch
{
    "returnFormat": "netfilter",
    "enforceWarninglist": 1,
    "tags": {
        "NOT": [
            "tlp:white",
            "type:OSINT"
        ],
        "OR": [
            "misp-galaxy:threat-actor="Sofacy",
            "misp-galaxy:sector="Chemical"
        ]
    }
}
```
Make decisions on whom to share data with based on context

- MISP by default decides based on the information creator’s decision who data gets shared with
- Community hosts should be able to act as a safety net for sharing
  - Push filters - what can I push?
  - Pull filters - what am I interested in?
  - Local tags allow for information flow control
The emergence of ATT&CK and similar galaxies

- Standardising on high-level **TTPs** was a solution to a long list of issues
- Adoption was rapid, tools producing ATT&CK data, familiar interface for users
- A much better take on kill-chain phases in general
- Feeds into our **filtering** and **situational awareness** needs extremely well
- Gave rise to other, ATT&CK-like systems tackling other concerns
  - **attck4fraud** by Francesco Bigarella from ING
  - **Election guidelines** by NIS Cooperation Group

---

3https://www.misp-project.org/galaxy.html#attck4fraud

4https://www.misp-project.org/galaxy.html#election_guidelines
Example query to generate ATT&CK heatmaps

/events/restSearch
{
    "returnFormat": "attack",
    "tags": [
        "misp-galaxy:sector="Chemical"
    ],
    "timestamp": "365d"
}
**A SAMPLE RESULT FOR THE ABOVE QUERY**

<table>
<thead>
<tr>
<th>Initial access</th>
<th>Execution</th>
<th>Persistence</th>
<th>Privilege escalation</th>
<th>Defense evasion</th>
<th>Credential access</th>
<th>Discovery</th>
<th>Lateral movement</th>
<th>Collection</th>
<th>Exfiltration</th>
<th>Command and control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spearphishing via Service</td>
<td>Command-Line Interface</td>
<td>Login Item</td>
<td>AosCert DLLs</td>
<td>Code Signing</td>
<td>Input Capture</td>
<td>System Network Configuration Discovery</td>
<td>Distributed Component Object Model</td>
<td>Data from Removable Media</td>
<td>Exfiltration Over Command and Control Channel</td>
<td>Communication Through Removable Media</td>
</tr>
<tr>
<td>Trusted Relationship</td>
<td>User Execution</td>
<td>Trap</td>
<td>Application Shimming</td>
<td>Rootkit</td>
<td>Each History</td>
<td>Process Discovery</td>
<td>Pass the Hash</td>
<td>Man in the Browser</td>
<td>Data Compressed</td>
<td>Custom Command and Control Protocol</td>
</tr>
<tr>
<td>Replication Through Removable Media</td>
<td>RegsOvs/Ragsam</td>
<td>System Firmware</td>
<td>Scheduled Task</td>
<td>NTFS File Attributes</td>
<td>Exploitation for Credential Access</td>
<td>Network Share Discovery</td>
<td>Exploitation of Remote Services</td>
<td>Data Staged</td>
<td>Automated Exfiltration</td>
<td>Multi-Stage Channels</td>
</tr>
<tr>
<td>Exploit Public-Facing Application</td>
<td>Trusted Developer Utilities</td>
<td>Registry Run Keys / Start Folder</td>
<td>Startup Items</td>
<td>Exploitation for Defense Evasion</td>
<td>Private Keys</td>
<td>Peripheral Device Discovery</td>
<td>Remote Desktop Protocol</td>
<td>Data Migration</td>
<td>Automated Exfiltration</td>
<td>Multi-Stage Channels</td>
</tr>
<tr>
<td>Spearphishing Link</td>
<td>Windows Management Instrumentation</td>
<td>LC_LOAD_DYLIB Addition</td>
<td>New Service</td>
<td>Network Share Discovery</td>
<td>Node Force</td>
<td>Account Discovery</td>
<td>Pass the Ticket</td>
<td>Email Collection</td>
<td>Data Encrypted</td>
<td>Uncommonly Used Port</td>
</tr>
<tr>
<td>Valid Accounts</td>
<td>Service Execution</td>
<td>LSASS Driver</td>
<td>Sudo Caching</td>
<td>Process Doppelganging</td>
<td>Password Filter DLL</td>
<td>System Information Discovery</td>
<td>Windows Remote Management</td>
<td>Clipboard Data</td>
<td>Exfiltration Over Other Network Medium</td>
<td>Multilayer Encryption</td>
</tr>
<tr>
<td>Supply Chain Compromise</td>
<td>CMSTP</td>
<td>Rcommon</td>
<td>Process Injection</td>
<td>Disabling Security Tools</td>
<td>Two-Factor Authentication Interception</td>
<td>System Network Connections Discovery</td>
<td>Windows Admin Shares</td>
<td>Video Capture</td>
<td>Exfiltration Over Physical Medium</td>
<td>Domain Framing</td>
</tr>
<tr>
<td>Drive-by Compromise</td>
<td>Control Panel Items</td>
<td>Authentication Package</td>
<td>Bypass User Account Control</td>
<td>Timestamp</td>
<td>LLMNR/NBT-NS Polishing</td>
<td>Network Service Scanning</td>
<td>Remote Services</td>
<td>Audio Capture</td>
<td>Data Transfer Size Limits</td>
<td>Data Obfuscation</td>
</tr>
<tr>
<td>Hardware Additions</td>
<td>Dynamic Data Exchange</td>
<td>Component Firmware</td>
<td>Extra Window Memory Injection</td>
<td>Modify Registry</td>
<td>Credentials in Files</td>
<td>File and Directory Discovery</td>
<td>Taint Shared Content</td>
<td>Data from Network Shared Drive</td>
<td>Connection Proxy</td>
<td>Commonly Used Port</td>
</tr>
<tr>
<td>Source</td>
<td>Windows Management Instrumentation Event Subscription</td>
<td>Suid and Suidgid</td>
<td>Indicator Removal from Tools</td>
<td>Forced Authentication</td>
<td>Security Software Discovery</td>
<td>Application Deployment Software</td>
<td>Data from Local System</td>
<td>Automated Collection</td>
<td>Data Encoding</td>
<td></td>
</tr>
<tr>
<td>Space after Filename</td>
<td>Change Default File</td>
<td>Launch Daemon</td>
<td>Hidden Window</td>
<td>Keychain</td>
<td>System Service Discovery</td>
<td>Third-party Software</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Monitor trends outside of MISP (example: dashboard)
Decaying of indicators

- We were still missing a way to use all of these systems in combination to decay indicators
- Move the decision making from complex filter options to complex decay models
- Decay models would take into account various taxonomies, sightings, the type of each indicator Sightings and Creation date
- The first iteration of what we have in MISP now took:
  - 2 years of research
  - 3 published research papers
  - A lot of prototyping
**Scoring Indicators: Our solution**

\[ \text{score}(\text{Attribute}) = \text{base\_score}(\text{Attribute, Model}) \cdot \text{decay}(\text{Model, time}) \]

Where,

- \( \text{score} \in [0, 100] \)
- \( \text{base\_score} \in [0, 100] \)
- \( \text{decay} \) is a function defined by model’s parameters controlling decay speed
- \( \text{Attribute} \) Contains \( \text{Attribute} \)’s values and metadata (\text{Taxonomies, Galaxies, ...})
- \( \text{Model} \) Contains the \( \text{Model} \)’s configuration
**Implementation in MISP: Event/view**

- **Decay score toggle button**
  - Shows Score for each *Models* associated to the *Attribute* type
Implementation in MISP: API result

/attributes/restSearch

"Attribute": [
  {
    "category": "Network activity",
    "type": "ip-src",
    "to_ids": true,
    "timestamp": "1565703507",
    [...]
    "value": "8.8.8.8",
    "decay_score": [
      {
        "score": 54.475223849544456,
        "decayed": false,
        "DecayingModel": {
          "id": "85",
          "name": "NIDS Simple Decaying Model"
        }
      },
### Decaying Models

<table>
<thead>
<tr>
<th>ID</th>
<th>Organization</th>
<th>Usable to everyone</th>
<th>Name</th>
<th>Description</th>
<th>Parameters {}</th>
<th>Formula</th>
<th># Assigned Types</th>
<th>Version</th>
<th>Enabled</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>1</td>
<td>✔</td>
<td>Phishing model</td>
<td>Simple model to rapidly decay phishing website.</td>
<td>{ &quot;lifetime&quot;: 3, &quot;decay_speed&quot;: 2.3, &quot;threshold&quot;: 30, &quot;default_base_score&quot;: 80, &quot;base_score_config&quot;: { &quot;estimative_language&quot;: 0.5, &quot;phishing&quot;: 0.5 } }</td>
<td>Polynomial</td>
<td>9</td>
<td>1</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>1</td>
<td>✗</td>
<td>NIDS Simple Decaying Model</td>
<td>Simple decaying model for Network Intrusion Detection System (NIDS).</td>
<td>{ &quot;lifetime&quot;: 120, &quot;decay_speed&quot;: 2, &quot;threshold&quot;: 30, &quot;default_base_score&quot;: 80, &quot;base_score_config&quot;: { &quot;estimative_language&quot;: 0.25, &quot;priority_level&quot;: 0.25, &quot;retention&quot;: 0.25, &quot;targeted_threat_index&quot;: 0.125, &quot;false_positive&quot;: 0.125 } }</td>
<td>Polynomial</td>
<td>13</td>
<td>1</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

Page 1 of 1, showing 2 records out of 2 total, starting on record 1, ending on 2

View, update, add, create, delete, enable, export, import
IMPLEMENTATION IN MISP: FINE TUNING TOOL

Create, modify, visualise, perform mapping
IMPLEMENTATION IN MISP: base_score TOOL

<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>admiralty-scale</td>
<td>31</td>
</tr>
<tr>
<td>source-reliability</td>
<td></td>
</tr>
<tr>
<td>information-credibility</td>
<td>30</td>
</tr>
<tr>
<td>priority-level</td>
<td></td>
</tr>
<tr>
<td>priority-level</td>
<td>53</td>
</tr>
<tr>
<td>retention</td>
<td></td>
</tr>
<tr>
<td>retention</td>
<td>0</td>
</tr>
<tr>
<td>estimative-language</td>
<td></td>
</tr>
<tr>
<td>likelihood-probability</td>
<td></td>
</tr>
<tr>
<td>confidence-in-analytic-judgment</td>
<td>0</td>
</tr>
<tr>
<td>misp</td>
<td></td>
</tr>
<tr>
<td>confidence-level</td>
<td>0</td>
</tr>
<tr>
<td>threat-level</td>
<td>0</td>
</tr>
<tr>
<td>automation-level</td>
<td>0</td>
</tr>
<tr>
<td>phishing</td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>0</td>
</tr>
<tr>
<td>psychological-acceptability</td>
<td>0</td>
</tr>
</tbody>
</table>

Example:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Tags</th>
<th>Base score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag your attribute</td>
<td><img src="#" alt="Admiralty Scale Information Credibility" /></td>
<td>5.0</td>
</tr>
<tr>
<td>Attribute 1</td>
<td>admiralty-scale:information-credibility=5</td>
<td>0.0</td>
</tr>
<tr>
<td>Attribute 2</td>
<td>priority-level:baseline-minor</td>
<td>38.2</td>
</tr>
<tr>
<td></td>
<td>admiralty-scale:source-reliability=d</td>
<td></td>
</tr>
<tr>
<td></td>
<td>admiralty-scale:information-credibility=2</td>
<td></td>
</tr>
<tr>
<td>Attribute 3</td>
<td>priority-level:severe</td>
<td>84.6</td>
</tr>
<tr>
<td></td>
<td>admiralty-scale:source-reliability=d</td>
<td></td>
</tr>
</tbody>
</table>

Computation steps:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Computation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority-level:baseline-minor</td>
<td>0.46 * 25.00</td>
<td>11.62</td>
</tr>
<tr>
<td>admiralty-scale:source-reliability=d</td>
<td>0.27 * 25.00</td>
<td>6.80</td>
</tr>
</tbody>
</table>
Simulate Attributes with different Models
/attributes/restSearch
{
   "includeDecayScore": 1,
   "includeFullModel": 0,
   "excludeDecayed": 0,
   "decayingModel": [85],
   "modelOverrides": {
      "threshold": 30
   }
   "score": 30,
}
To sum it all up...

- Massive rise in **user capabilities**
- Growing need for truly **actionable threat intel**
- Lessons learned:
  - **Context is king** - Enables better decision making
  - **Intelligence and situational awareness** are natural by-products of context
  - Don’t lock users into your **workflows**, build tools that enable theirs
Get in touch if you have any questions

Contact us
- https://twitter.com/mokaddem_sami
- https://twitter.com/iglocskka

Contact CIRCL
- info@circl.lu
- https://twitter.com/circl_lu
- https://www.circl.lu/

Contact MISPProject
- https://github.com/MISP
- https://gitter.im/MISP/MISP
- https://twitter.com/MISPProject
MISP Standard

The collaborative intelligence standard powering intelligence and information exchange, sharing and modeling.

CIRCL / Team MISP Project

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

FIRST workshop
Following the grow of organisations relying on MISP, the JSON format used by MISP are standardised under the misp-standard.org umbrella.

The goal is to provide a flexible set of standards to support information exchange and data modeling in the following field:

- Cybersecurity intelligence
- Threat intelligence
- Financial fraud
- Vulnerability information
- Border control information
- Digital Forensic and Incident Response
- and intelligence at large
This standard describes the **MISP core format** used to exchange indicators and threat information between MISP instances. The **JSON format includes the overall structure along with the semantics associated for each respective key**. The format is described to support other implementations, aiming to reuse the format and ensuring the interoperability with the existing MISP software and other Threat Intelligence Platforms.
This standard describes the **MISP object** template format which describes a simple JSON format to represent the various templates used to construct MISP objects. A public directory of common **MISP object templates and relationships** is available and relies on the MISP object reference format.
This standard describes the **MISP galaxy format** which describes a simple **JSON format to represent galaxies and clusters** that can be attached to MISP events or attributes. A public directory of MISP galaxies is available and relies on the MISP galaxy format. MISP galaxies are used to attach additional information structures such as MISP events or attributes. **MISP galaxy is a public repository of known malware, threats actors and various other collections of data that can be used to mark, classify or label data in threat information sharing.**
This standard describes the format used by SightingDB to give automated context to a given Attribute by counting occurrences and tracking times of observability. SightingDB was designed to provide to MISP and other tools an interoperable, scalable and fast way to store and retrieve attributes sightings.
If you want to contribute to our IETF Internet-Draft for the MISP standard, misp-rfc\(^1\) 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\(^2\).

The process is always MISP implementation $\rightarrow$ IETF I-D updates.


\(^1\)https://github.com/MISP/misp-rfc
\(^2\)https://datatracker.ietf.org/doc/search/?name=misp&activedrafts=on&rfcs=on
MISP CLI

Automate all the things

CIRCL / Team MISP Project

MISP
Threat Sharing

FIRST workshop
The MISP API is great for remotely executing administrative tasks.

But sometimes we want to simplify the process / avoid having to deal with authentication.

MISP also has an extensive CLI sub-system for this reason.
Types of objectives for the scripts

- Automating recurring tasks
- Recovery from loss of access
- Updates / initialisation
- Background worker management
Administering the background workers via the API.

You can start/stop and view the background workers via the API.
- **Add worker**: `http://localhost:5001/servers/startWorker[/queue_name]`
- **Stop worker**: `http://localhost:5001/servers/stopWorker[/worker_pid]`
- **Get worker info**: `http://localhost:5001/servers/getWorkers`

Administering MISP via the CLI

Certain administrative tasks are exposed to the API, these help with maintaining and configuring MISP in an automated way / via external tools:
- **Get Setting**: `MISP/app/Console/cake Admin getSetting [setting]`
- **Set Setting**: `MISP/app/Console/cake Admin setSetting [setting] [value]`
- **Get Authkey**: `MISP/app/Console/cake Admin getAuthkey [email]`
- **Set Baseurl**: `MISP/app/Console/cake Baseurl [baseurl]`
- **Change Password**: `MISP/app/Console/cake Admin setPassword [email] [new_password] [--override_password_change]`
- **Clear Bruteforce Entries**: `MISP/app/Console/cake Admin clearBruteForce [user_email]`
- **Run Database Update**: `MISP/app/Console/cake Admin updateDatabase`
- **Update All JSON Structures**: `MISP/app/Console/cake Admin updateJSON`
- **Update Galaxy Definitions**: `MISP/app/Console/cake Admin updateGalaxies`
- **Update Taxonomy Definitions**: `MISP/app/Console/cake Admin updateTaxonomies`
- **Update Object Templates**: `MISP/app/Console/cake Admin updateObjectTemplates`
- **Update Warninglists**: `MISP/app/Console/cake Admin updateWarningLists`
Usage

/var/www/MISP/app/Console/cake [Shell] [Command] [parameters]

- Example:
  - /var/www/MISP/app/Console/cake Password "andras.iklody@gmail.com" "Nutella"
  - Change password to "Nutella" for my user
  - Some shells are single use and don’t need a command parameter

- Also used by the background processing
- Automation is meant to be used via cron jobs
Edit crontab of www-data user

```
crontab -u www-data -e
0 3,9,15,21 * * * /var/www/MISP/app/Console/cake Server pull 1 30 full
Pull server ID #30 as user #1 every 6 hours
@hourly /var/www/MISP/app/Console/cake Server cacheFeed 1 csv full
Cache all csv feeds as user #1 every hour
```
MISP Deployment

Some basic guidelines

CIRCL / Team MISP Project

FIRST workshop
MISP DEPLOYMENT CONSIDERATIONS

- Deployment types
- Distro choice
- Hardware specs
- Authentication
- Other considerations - settings, gotchas
Deployment types

- Native install
  - Manual
  - One liner script - INSTALL.sh
    https://github.com/MISP/MISP/tree/2.4/INSTALL
- MISP VM
  https://www.circl.lu/misp-images/latest/
- Docker
- RPM maintained by SWITCH
  https://github.com/amuehlem/MISP-RPM
- Cloud provider images
  https://github.com/MISP/misp-cloud
Docker options

- CoolAcid’s MISP images
  https://github.com/coolacid/docker-misp

- MISP-docker by XME
  https://github.com/MISP/misp-docker

- docker-misp by Harvard security
  https://github.com/MISP/docker-misp
DISTRO OPTIONS

- Ubuntu 20.04 (18.04 will also work)
  - Our target platform
  - Our CI target
  - Use this unless you are absolutely forced not to
  - This is the platform we can support you with!

- CentOS 7
  - Annoying to operate
  - Less tested, though used by many
  - CentOS is going away. Consider other options

- RHEL 7
  - Same annoyance as CentOS in general
  - We test against CentOS in general, some assembly may be required
Hardware Specs

- No firm recommendations, it’s highly usage dependent
- It’s better to go a bit over what you need than under
- **SSDs** are massively beneficial
- Let’s look at what affects specs and some sample configurations
What are the factors that can impact my performance?

- Clustering of the data (how many datapoints / event?) (RAM, disk speed)
- Correlation (RAM, disk speed, disk space)
  - Consider blocking overtly correlating values from doing so
  - Feed ingestion strategy is crucial
- Over-contextualisation (RAM, disk speed)
  - Tag/attach galaxies to the event instead of each attribute when possible
What are the factors that can impact my performance?

- Number of users that are active at any given time (RAM, CPU, disk speed)
- Logging strategy (Disk space)
- API users especially with heavy searches (substring searches for example) (RAM, CPU, Disk speed)
What are the factors that generally do **NOT** impact my performance as much as expected?

- Warninglist usage
- Number of raw attributes on the instance
- Number of sync connections / recurring syncs (with measure)
- Tools feeding off the automation channels (ZMQ, kafka, syslog)
Authentication options

- Username/password is the default
- Some built in modules by 3rd parties (LDAP, Shibboleth, x509, OpenID, Azure Active Directory)
- CustomAuth system for more flexibility
- Additionally, consider Email OTP
Other considerations - tuning

- PHP tuning
  - Maximum memory usage (per process)
  - Timeout settings
  - Consider setting it per role!
  - Background processes are exempt

- MySQL: key buffer size is important

- Generally, tune for few heavy requests rather than many light ones
Other Considerations - High Availability

- Clustering
  - Load balanced apache servers with MISP
  - Replicating / mirrored database backends
- Careful about session pinning
- Attachment storage can be abstracted / network attached
- An example implementation for AWS
  https://github.com/0xtf/HAMISPA
**Correlations**: Links created automatically whenever an Attribute is created or modified. They allow interconnection between Events based on their attributes.

**Correlation Engine**: Is the system used by MISP to create correlations between Attribute’s value. It currently supports strict string comparison, SSDEEP and CDIR blocks matches.

**Caching**: Is the process of fetching data from a MISP instance or feed but only storing hashes of the collected values for correlation and look-up purposes.

**Delegation**: Act of transferring the ownership of an Event to another organisation while hiding the original creator, thus providing anonymity.

**Deletion (hard/soft)**: Hard deletion is the act of removing the element from the system; it will not perform revocation on other MISP instances. Soft deletion is the act flagging an element as deleted and propagating the revocation among the network of connected MISP instances.

**Extended Event**: Event that extends an existing Event, providing a combined view of the data contained in both Events. The owner of the extending Event is the organisation that created the extension. This allows anyone to extend any Events and have total control over them.

**Galaxy Matrix**: Matrix derived from Galaxy Clusters belonging to the same Galaxy. The layout (pages and columns) is defined at the Galaxy level and its content comes from the Galaxy Clusters meta-data themselves.

**Indicators**: Attribute containing a pattern that can be used to detect suspicious or malicious activity. These Attributes usually have their to_ids flag enabled.

**Orgc / Org**: Creator Organisation (Orgc) is the organisation that created the data and the one allowed to modify it. Owner Organisation (Org) is the organisation owning the data on a given instance and is allowed to view it regardless of the distribution level. The two are not necessarily the same.

**Publishing**: Action of declaring that an Event is ready to be synchronised. It may also send e-mail notifications and makes it available to some export formats.

**Pulling**: Action of using a user on a remote instance to fetch the accessible data and storing it locally.

**Pushing**: Action of using an uplink connection via a sync user to send data to a remote instance.

**Synchronisation**: Is the exchange of data between two (or more) MISP instances through the pull or push mechanisms.

**Sync. filtering rule**: Can be applied on a synchronisation link for both the pull and push mechanisms to block or allow data to be transferred.

**Sync. User**: Special role of a user granting additional sync permissions. The recommended way to setup push synchronisation is to use sync users.

**Proposals**: Are a mechanism to propose modifications to the creating organisations (Orgc). If a path of connected MISP instances exists, the Proposal will be synchronised allowing the creator to accept or discard it.
MISP Data Model Cheat Sheet

**Event**

Encapsulations for contextually linked information.

**Purpose:** Group datapoints and context together. Acting as an envelop, it allows setting distribution and sharing rules for itself and its children.

**Use case:** Encode incidents/events/reports/...

- **Events** can contain other elements such as Attributes, MISP Objects and Event Reports.
- The distribution level and any context added on an Event (such as Taxonomies) are propagated to its underlying data.

**Attribute**

Basic building block to share information.

**Purpose:** Individual data point. Can be an indicator or supporting data.

**Use case:** Domain, IP, link, sha1, attachment, ...

- **Attributes** cannot be duplicated inside the same Event and can have Sightings.
- The difference between an indicator or supporting data is usually indicated by the state of the attribute’s to_ids flag.

**MISP Object**

Advanced building block providing Attribute compositions via templates.

**Purpose:** Groups Attributes that are intrinsically linked together.

**Use case:** File, person, credit-card, x509, device, ...

- **MISP Objects** have their attribute compositions described in their respective template.
- They are instantiated with Attributes and can Reference other Attributes or MISP Objects.
- **MISP** is not required to know the template to save and display the object. However, edits will not be possible as the template to validate against is unknown.

**Object Reference**

Relationships between individual building blocks.

**Purpose:** Allows to create relationships between entities, thus creating a graph where they are the edges and entities are the nodes.

**Use case:** Represent behaviours, similarities, affiliation, ...

- **References** can have a textual relationship which can come from MISP or be set freely.

**Sightings**

Means to convey that an Attribute has been seen.

**Purpose:** Allows to add temporality to the data.

**Use case:** Record activity or occurrence, perform IoC expiration, ...

- **Sightings** are the best way to express that something has been seen. They can also be used to mark false positives.

**Event Report**

Advanced building block containing formatted text.

**Purpose:** Supporting data point to describe events or processes.

**Use case:** Encode reports, provide more information about the Event, ...

- **Event Reports** are markdown-aware and include a special syntax to reference data points or context.

**Proposals**

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

**Purpose:** Allow the correction or the creation of Attributes for Events your organisation does not own.

**Use case:** Disable the IDS flag, Correct errors

- As Proposals are sync., if the creator organisation is connected to the MISP instance from where the Proposal has been created, it will be able to either accept or discard it.

**Taxonomies**

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

**Purpose:** Enable efficient classification globally understood, easing consumption and automation.

**Use case:** Provide classification such as: TLP, Confidence, Source, Workflows, Event type, ...

- Even though MISP allows the creation of free-text tags, it’s always preferable to use those coming from Taxonomies, if they exist.

**Galaxies**

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

**Purpose:** Bundle Galaxy Clusters by their type to avoid confusion and to ease searches.

**Use case:** Bundle types: Exploit-Kit, Preventive Measures, ATT&CK, Tools, Threat-actors, ...

**Galaxies Clusters**

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

**Purpose:** Enable description of complex high-level information for classification.

**Use case:** Extensively describe elements such as threat actors, countries, technique used, ...

- Galaxy Clusters can be seen as an enhanced Taxonomy as they can have meta-data and relationships with other Galaxy Clusters.
- Any Galaxy Clusters can contain the following:
  - **Cluster Elements:** Key-Value pair forming the meta-data.
  - **Cluster Relations (🔗🔗🔗):** Enable the creation of relationships between one or more Galaxy Clusters.

Example: Threat actor X is similar to threat actor Y with high-likelihood.
**MISP User & Admin Cheat Sheet**

### - User -

#### API

**Wildcard searches:**

```plaintext
POST /attributes/restSearch
{
    "value": "1.2.3.%"
}
```

**Or and Negation searches:**

```plaintext
POST /attributes/restSearch
{
    "tags": ["tlp:white", "tlp:yellow"]
}
```

**And and Negation searches:**

```plaintext
POST /attributes/restSearch
{
    "tags": {"AND": ["tlp:green", "Malware"], "NOT": ["%ransomware%"]}
}
```

**Galaxy Cluster metadata searches:**

```plaintext
POST /attributes/restSearch
{
    "galaxy.synonyms": "APT29",
    "galaxy.cfr-target-category": "Financial sector"
}
```

**Attach tags:**

```plaintext
POST /tags/attachTagToObject
{
    "uuid": "[Could be UUID from Event, Attribute, ...]",
    "tag": "tlp:amber"
}
```

**Timestamps:**

- **timestamp**: Time of the last modification on the data
  - Use case: Get data was modified in the last t
  - E.g.: Last updated data from a feed
- **publish_timestamp**: Time at which the event was published
  - Use case: Get data that arrived in my system since t
  - E.g.: New data from a feed
- **event_timestamp**: Used in the Attribute scope
  - Use case: Get events modified in the last t

**Usage:**

```json
{"timestamp": 1521846000}
{"timestamp": "7d"}
{"timestamp": ["2d", "1h"]}
```

### - Admin -

#### Reset Password

**API:**

```plaintext
POST /users/initiatePasswordReset/[id]  {"password": "***"}
```

**CLI:**

```plaintext
MISP/app/Console/cake Password [email] [password]
```

#### Reset Bruteforce login protection

**CLI:**

```plaintext
MISP/app/Console/cake Admin clearBruteforce [email]
```

#### Upgrade to the latest version

**All in 1-shot:**

```plaintext
MISP/app/Console/cake Admin updateMISP
```

**Manually:**

1. cd /var/www/MISP
2. git pull origin 2.4
3. git submodule update --init --recursive
4. MISP/app/Console/cake Admin updateJSON
5. Check live update progress GET /servers/updateProgress

#### Workers

**Restart All:**

```plaintext
MISP/app/Console/cake Admin restartWorkers
```

**Add:**

```plaintext
MISP/app/Console/cake Admin startWorker [queue]
```

**Stop:**

```plaintext
MISP/app/Console/cake Admin stopWorker [pid]
```

#### Settings

**Get:**

```plaintext
MISP/app/Console/cake Admin getSetting [setting]
```

**Set:**

```plaintext
MISP/app/Console/cake Admin setSetting [setting] [value]
```

**Base URL:**

```plaintext
MISP/app/Console/cake Baseurl [baseurl]
```

#### Miscellaneous

**Clean Caches:**

```plaintext
MISP/app/Console/cake Admin cleanCaches
```

**Get IPs For User ID:**

```plaintext
MISP/app/Console/cake Admin UserIP [user_id]
```

**Get User ID For User IP:**

```plaintext
MISP/app/Console/cake Admin IPUser [ip]
```

**Documentation:**

```plaintext
/events/automation
```

**Logs files location:**

```plaintext
MISP/app/tmp/logs
```

### Tips & Tricks

**Get JSON Representation:** Append .json to any URLs to get their content in JSON format. Example: /events/view/42.json
<table>
<thead>
<tr>
<th>Check Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add events</td>
<td></td>
</tr>
<tr>
<td>- via Standard UI</td>
<td></td>
</tr>
<tr>
<td>- Distribution levels and publication</td>
<td></td>
</tr>
<tr>
<td>- Different timestamps &amp; publish_timestamp</td>
<td></td>
</tr>
<tr>
<td>Add attributes</td>
<td></td>
</tr>
<tr>
<td>- Freetext</td>
<td></td>
</tr>
<tr>
<td>- Standard UI</td>
<td></td>
</tr>
<tr>
<td>- Template</td>
<td></td>
</tr>
<tr>
<td>- ReST API</td>
<td></td>
</tr>
<tr>
<td>- via EventGraph</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td></td>
</tr>
<tr>
<td>- add Object</td>
<td></td>
</tr>
<tr>
<td>- add References</td>
<td></td>
</tr>
<tr>
<td>- show via EventGraph</td>
<td></td>
</tr>
<tr>
<td>*.lists</td>
<td></td>
</tr>
<tr>
<td>- Warninglists: show warnings raised in steps above</td>
<td></td>
</tr>
<tr>
<td>- Noticelists: show warnings when adding data</td>
<td></td>
</tr>
<tr>
<td>- Import Regexp: avoid leaking private/personal data</td>
<td></td>
</tr>
<tr>
<td>Correlations</td>
<td></td>
</tr>
<tr>
<td>- show correlations that were added</td>
<td></td>
</tr>
<tr>
<td>- pivot to events via correlations</td>
<td></td>
</tr>
<tr>
<td>- show correlations graph</td>
<td></td>
</tr>
<tr>
<td>- feeds &amp; servers correlation</td>
<td></td>
</tr>
<tr>
<td>Tags and Galaxies</td>
<td></td>
</tr>
<tr>
<td>- Tag from Taxonomy</td>
<td></td>
</tr>
<tr>
<td>- GalaxyCluster</td>
<td></td>
</tr>
<tr>
<td>- ATT&amp;CK pattern &amp; Galaxy matrix</td>
<td></td>
</tr>
<tr>
<td>- Tag Collection</td>
<td></td>
</tr>
<tr>
<td>Sighting</td>
<td></td>
</tr>
<tr>
<td>- via UI &amp; API</td>
<td></td>
</tr>
<tr>
<td>Delegation</td>
<td></td>
</tr>
<tr>
<td>Proposal</td>
<td></td>
</tr>
<tr>
<td>Delete (including soft versus hard delete)</td>
<td></td>
</tr>
<tr>
<td>- Event blocklist when deleting</td>
<td></td>
</tr>
<tr>
<td>Extending event (how and when to use it)</td>
<td></td>
</tr>
<tr>
<td>Exporting data</td>
<td></td>
</tr>
<tr>
<td>- download from</td>
<td></td>
</tr>
<tr>
<td>- download from via modules</td>
<td></td>
</tr>
<tr>
<td>- .json routing</td>
<td></td>
</tr>
<tr>
<td>- RestSearch</td>
<td></td>
</tr>
<tr>
<td>Searching for data</td>
<td></td>
</tr>
<tr>
<td>- Attribute search</td>
<td></td>
</tr>
<tr>
<td>- Event index filter search</td>
<td></td>
</tr>
<tr>
<td>Advanced features</td>
<td></td>
</tr>
<tr>
<td>- Event graph, Event timeline, Event report</td>
<td></td>
</tr>
<tr>
<td>- Decaying of IoC</td>
<td></td>
</tr>
<tr>
<td>- Galaxy 2.0</td>
<td></td>
</tr>
<tr>
<td>Enrichments</td>
<td></td>
</tr>
<tr>
<td>- Hover &amp; persistent</td>
<td></td>
</tr>
<tr>
<td>Check</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Organisations</strong></td>
<td>- local and remote</td>
</tr>
<tr>
<td></td>
<td>- administration: Creation and merge</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>- administration and contact via standard UI</td>
</tr>
<tr>
<td></td>
<td>- Password/Auth key reset</td>
</tr>
<tr>
<td></td>
<td>- Disabling (never remove)</td>
</tr>
<tr>
<td><strong>Roles and permissions</strong></td>
<td>- Constraints &amp; special sync-user</td>
</tr>
<tr>
<td><strong>Sharing group</strong></td>
<td>- administration via standard UI</td>
</tr>
<tr>
<td></td>
<td>- Events &amp; Organisations</td>
</tr>
<tr>
<td><strong>Block listing</strong></td>
<td>- administration via standard UI</td>
</tr>
<tr>
<td><strong>Synchronisation</strong></td>
<td>- MISP to MISP (sync_user, test &amp; preview, flow control)</td>
</tr>
<tr>
<td></td>
<td>- Feeds to MISP (Options, overlap)</td>
</tr>
<tr>
<td></td>
<td>- Pub-Sub</td>
</tr>
<tr>
<td><strong>Collaboration settings</strong></td>
<td>- 'proposal_block_attributes', 'sanitise_attribute_on_delete', 'Sightings_anonymise'</td>
</tr>
<tr>
<td><strong>Templates</strong></td>
<td>- administration via standard UI</td>
</tr>
</tbody>
</table>
### Administrator (Instance)

<table>
<thead>
<tr>
<th>Check</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Auth keys</td>
<td>- Migration from old system</td>
<td>3m</td>
</tr>
<tr>
<td></td>
<td>- Usage</td>
<td></td>
</tr>
<tr>
<td>Server settings</td>
<td></td>
<td>5m</td>
</tr>
<tr>
<td>Maintenance</td>
<td>- Updating &amp; release process</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Submodules and populate DB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Diagnostic</td>
<td></td>
</tr>
<tr>
<td>Jobs and Workers</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>User settings &amp; User management</td>
<td></td>
<td>5m</td>
</tr>
<tr>
<td></td>
<td>- User settings</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- User monitoring, self-management, auto-registration</td>
<td></td>
</tr>
<tr>
<td>Logging &amp; auditing</td>
<td>- Logs (and purge: event history)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Paranoid, IP &amp; Auth log, Sync audit</td>
<td></td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>- Clean cache &amp; DB Schema diagnostic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Stuck workers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Update in progress</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Apache logs &amp; workers logs</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/
[^2]: https://github.com/MISP/misp-training
[^3]: https://www.circl.lu/