An Introduction to Cybersecurity Information Sharing

MISP - Threat Sharing

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

Agenda

- Agenda and details available https://hdoc.csirt-tooling.org/tq-qyvTQTLeZOwy-OPXjiw?view

MISP and starting from a practical use-case

- 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**.

ABOUT CIRCL

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.

MISP and CIRCL

- CIRCL is mandated by the Ministry of Economy and acting as the Luxembourg National CERT for private sector.
- CIRCL leads the development of the Open Source MISP threat intelligence platform which is used by many military or intelligence communities, private companies, financial sector, National CERTs and LEAs globally.
- **CIRCL runs multiple large MISP communities performing active daily threat-intelligence sharing**.

Co-financed by the European Union
Connecting Europe Facility

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

Development based on practical user feedback

- 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, likelyhood and occurences.
  - **Fraud analysts** willing to share financial indicators to detect financial frauds.

MISP MODEL OF GOVERNANCE

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?'
- $\rightarrow$ 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 restriction[1]
  - "Our legal framework doesn't allow us to share information."
  - "Risk of information-leak is too high and it's too risky for our organization or partners."
- Practical restriction
  - "We don't have information to share."
  - "We don't have time to process or contribute indicators."
  - "Our model of classification doesn't fit your model."
  - "Tools for sharing information are tied to a specific format, we use a different one."

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

MISP Project Overview

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

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**

Conclusion

- **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.

An Introduction to Cybersecurity Information Sharing

MISP - Threat Sharing

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

Agenda

- Agenda and details available https://hdoc.csirt-tooling.org/tq-qyvTQTLeZOwy-OPXjiw?view

MISP and starting from a practical use-case

- 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**.

ABOUT CIRCL

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.

MISP and CIRCL

- CIRCL is mandated by the Ministry of Economy and acting as the Luxembourg National CERT for private sector.
- CIRCL leads the development of the Open Source MISP threat intelligence platform which is used by many military or intelligence communities, private companies, financial sector, National CERTs and LEAs globally.
- **CIRCL runs multiple large MISP communities performing active daily threat-intelligence sharing**.

Co-financed by the European Union

Connecting Europe Facility

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

Development based on practical user feedback

- 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, likelyhood and occurences.
  - **Fraud analysts** willing to share financial indicators to detect financial frauds.

MISP MODEL OF GOVERNANCE

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?'
- $\rightarrow$ 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 restriction[1]
  - "Our legal framework doesn't allow us to share information."
  - "Risk of information-leak is too high and it's too risky for our organization or partners."
- Practical restriction
  - "We don't have information to share."
  - "We don't have time to process or contribute indicators."
  - "Our model of classification doesn't fit your model."
  - "Tools for sharing information are tied to a specific format, we use a different one."

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

MISP Project Overview

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 occurances 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

Terminology about Indicators

- Indicators[2]
  - 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**.

[2] IoC (Indicator of Compromise) is a subset of indicators

A RICH DATA-MODEL: TELLING STORIES VIA RELATIONSHIPS

Filters: All | File | Network | Financial | Proposal | Correlation | Warnings | Include deleted attributes | Show context fields

Date	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events
2018-09-28		Name: bank-account References: 0							
2018-09-28		Other	status-code: text	A - Active	+	Add		☐	
2018-09-28		Other	report-code: text	STR Suspicious Transaction Report	+	Add		☐	
2018-09-28		Other	personal-account-type: text	A - Business	+	Add		☐	
2018-09-28		Financial fraud	swift: bic	HASEHKHH	+	Add		☑	3849 11320 11584
2018-09-28		Financial fraud	account: bank-account-nr	788796894883	+	Add		☑	
2018-09-28		Other	account-name: text	FANY SILU CO. LIMITED	+	Add		☑	
2018-09-28		Other	currency-code: text	USD	+	Add		☐	

Contextualisation and aggregation

- MISP integrates at the event and the attribute levels MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK).

Pre Attack - Attack Pattern | **Enterprise Attack - Attack Pattern** | Mobile Attack - Attack Pattern

0 — 11 — Show all

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Exfiltration	Command and control
Spearphishing Attachment	Scripting	Screensaver	File System Permissions Weakness	Process Hollowing	Securityd Memory	Password Policy Discovery	AppleScript	Data from Information Repositories	Exfiltration Over Alternative Protocol	Standard Application Layer Protocol
Spearphishing via Service	Command-Line Interface	Login Item	AppCert DLLs	Code Signing	Input Capture	System Network Configuration Discovery	Distributed Component Object Model	Data from Removable Media	Exfiltration Over Command and Control Channel	Communication Through Removable Media
Trusted Relationship	User Execution	Trap	Application Shimming	Rootkit	Bash History	Process Discovery	Pass the Hash	Man in the Browser	Data Compressed	Custom Command and Control Protocol
Replication Through Removable Media	Regsvcs/Regasm	System Firmware	Scheduled Task	NTFS File Attributes	Exploitation for Credential Access	Network Share Discovery	Exploitation of Remote Services	Data Staged	Automated Exfiltration	Multi-Stage Channels
Exploit Public-Facing Application	Trusted Developer Utilities	Registry Run Keys / Start Folder	Startup Items	Exploitation for Defense Evasion	Private Keys	Peripheral Device Discovery	Remote Desktop Protocol	Screen Capture	Scheduled Transfer	Remote Access Tools
Spearphishing Link	Windows Management Instrumentation	LC_LOAD_DYLIB Addition	New Service	Network Share Connection Removal	Brute Force	Account Discovery	Pass the Ticket	Email Collection	Data Encrypted	Uncommonly Used Port
Valid Accounts	Service Execution	LSASS Driver	Sudo Caching	Process Doppelgänging	Password Filter DLL	System Information Discovery	Windows Remote Management	Clipboard Data	Exfiltration Over Other Network Medium	Multilayer Encryption
Supply Chain Compromise	CMSTP	Rc.common	Process Injection	Disabling Security Tools	Two-Factor Authentication Interception	System Network Connections Discovery	Windows Admin Shares	Video Capture	Exfiltration Over Physical Medium	Domain Fronting
Drive-by Compromise	Control Panel Items	Authentication Package	Bypass User Account Control	Timestomp	LLMNR/NBT-NS Poisoning	Network Service Scanning	Remote Services	Audio Capture	Data Transfer Size Limits	Data Obfuscation
Hardware Additions	Dynamic Data Exchange	Component Firmware	Extra Window Memory Injection	Modify Registry	Credentials in Files	File and Directory Discovery	Taint Shared Content	Data from Network Shared Drive		Connection Proxy
	Source	Windows Management Instrumentation Event Subscription	Setuid and Setgid	Indicator Removal from Tools	Forced Authentication	Security Software Discovery	Application Deployment Software	Data from Local System		Commonly Used Port
	Space after Filename	Change Default File	Launch Daemon	Hidden Window	Keychain	System Service Discovery	Third-party Software	Automated Collection		Data Encoding

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**

Correlation features: a tool for analysts

- 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 sigthings (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

Decaying of indicators: simulation tool

Simulate *Attributes* with different *Models*

Bootstrapping your MISP with data

- 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 data[3].

[3] A recurring challenge in information sharing

Conclusion

- **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

MISP Project

MISP
Threat Sharing

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/`

MISP - VM

- 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

MISP - General Usage

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)

MISP - Event (Discussions)

MISP - Event (Taxonomies and proposal correlations)

MISP - Event (The state of the art MISP datamodel)

MISP - Viewing the Event Index

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

MISP - Viewing an Event

- Event View
  - Event context
  - Attributes
    - Category/type, IDS, Correlations
  - Objects
  - Galaxies
  - Proposals
  - Discussions
- Tools to find what you are looking for
- Correlation graphs

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

MISP - Various features while adding data

- What happens automatically when adding data?
  - Automatic correlation
  - Input modification via validation and filters (regex)
  - Tagging / Galaxy Clusters
- Various ways to publish data
  - Publish with/without e-mail
  - Publishing via the API
  - Delegation

MISP - Using the data

- Correlation graphs
- Downloading the data in various formats
- API (explained later)
- Collaborating with users (proposals, discussions, emails)

MISP - Sync explained (if no admin training)

- 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

Sharing group settings

Instances	Organisations
Misp 1	Org1
Misp 2	Org2
Misp 3 (all)	Org4
	Org6

MISP - Exports and API

- 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

MISP Project

MISP
Threat Sharing

A Common Integration

Recommended MISP Setup

- 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?

Vendors and Formats

- 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

Reporting Back from your Devices, Tools or Processes

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

Q&A

- 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/

MISP Project

MISP
Threat Sharing

Viper - Main ideas

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.*

Viper

- **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)

Viper

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

PyMISP & Viper

- 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

MISP Module

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

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

Viper & VT

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

VirusTotal Module

```
Lookup the file on VirusTotal

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

Q&A

- https://github.com/MISP/PyMISP
- https://github.com/MISP/
- https://github.com/viper-framework/viper
- We welcome new functionalities and pull requests.

MAIL_TO_MISP

Connect your mail infrastructure to MISP to

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

Context

- You receive emails with IoC's inside
- How to create an event out of it?
- Create event manually and copy paste
- $\rightarrow$ This works once or twice
- Forwarding the email would be nice
- $\rightarrow$ mail_to_misp

Features: Email handling

- Extraction of URLs and IP addresses and port numbers
- Extraction of hostnames from URLs
- Extraction of hashes (MD5, SHA1, SHA256)
- DNS expansion
- Subject filters
- Refanging of URLs ('hxxp://...')
- ... and more

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

Implementation

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

Installation

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

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

```
misp_url = 'http://127.0.0.1/'
misp_key = 's5jPWCIud36Z8XHgsiCVI7SaL1XsMTyfEsN45tTe'
misp_verifycert = True
body_config_prefix = 'm2m'
...
...
```

Exercise: mail_2_misp.py

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

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

- Bonus: Fake-SMTPD spamtrap

```
./fake_smtp.py

telnet 127.0.0.1 2526
    Trying 127.0.0.1...
    Connected to 127.0.0.1.
    Escape character is '^]'.
    220 misp Python SMTP 1.1
    helo misp
    250 misp
    mail from: mikel
    250 OK
    rcpt to: 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

MISP Project

MISP - VM

- 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

MISP - Administration

- Plan for this part of the training
  - User and Organisaton 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

MISP - Templates

- Why templating?
- Create a basic template
- Text fields
- Attribute fields
- Attachment fields
- Automatic tagging

MISP - Tags and Taxonomies

- git submodule init && git submodule update
- Loading taxonomies
- Enabling taxonomies and associated tags
- Tag management
- Exportable tags

MISP - Object Templates

- git submodule init && git submodule update
- Enabling objects (and what about versioning)

MISP - Whitelisting, Regexp entries, Warninglists

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

MISP - Setting up the synchronisation

- Requirements - versions
- Pull/Push
- One way vs Two way synchronisation
- Exchanging sync users
- Certificates
- Filtering
- Connection test tool
- Previewing an instance
- Cherry picking and keeping the list updated

MISP - Scheduled tasks

- How to schedule the next execution
- Frequency, next execution
- What happens if a job fails?

MISP - Setting up the synchronisation

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

MISP - Settings and diagnostics

- Settings
  - Settings interface
  - The tabs explained at a glance
  - Issues and their severity
  - Setting guidance and how to best use it

MISP - Settings and diagnostics continued

- 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

MISP - Settings and diagnostics continued

- Plugins
  - Enrichment Modules
  - RPZ
  - ZeroMQ

MISP - Settings and diagnostics continued

- Diagnostics
  - Updating MISP
  - Writeable Directories
  - PHP settings
  - Dependency diagnostics

MISP - Settings and diagnostics continued

- 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

MISP - Settings and diagnostics continued

- 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 us-

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

From Tagging to Flexible Taxonomies

OSINT - Fancy Bear Source Code

Event ID	5703
Uuid	58724cbf-5508-4425-ab89-4f61950d210f
Org	CIRCL
Owner org	CIRCL
Contributors	
Email	alexandre.dulaunoy@circl.lu
Tags	tlp:white x osint:certainty="75" x osint:source-type="source-code-repository" x circl:osint-feed x ms-caro-malware:malware-platform="Python" x +
Date	2017-01-08
Threat Level	Medium
Analysis	Initial
Distribution	All communities
Info	OSINT - Fancy Bear Source Code
Published	Yes
Sightings	0 (0)
Activity	

- 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.

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 way that allows systems to parse and interpret it.
- Still have a human-readable version:
  - admiralty-scale:source-reliability="Fairly reliable"

MISP Taxonomies

- 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

```
{
  "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"
    }
  ],
....
```

Want to write your own taxonomy? 2/2

```
1  {
2    "values": [
3      {
4        "predicate": "source-reliability",
5        "entry": [
6          {
7            "value": "a",
8            "expanded": "Completely reliable"
9          },
10 ....
```

- Publishing your taxonomy is as easy as a simple git pull request on misp-taxonomies[2].

[2] https://github.com/MISP/misp-taxonomies

How are taxonomies integrated in MISP?

18	✔	✖	admiralty-scale:information-credibility="1"	admiralty-scale	4	0			
19	✔	✖	admiralty-scale:information-credibility="2"	admiralty-scale	15	1			
20	✔	✖	admiralty-scale:information-credibility="3"	admiralty-scale	12	4			
21	✔	✖	admiralty-scale:information-credibility="4"	admiralty-scale	1	0			
22	✔	✖	admiralty-scale:information-credibility="5"	admiralty-scale	1	0			
23	✔	✖	admiralty-scale:information-credibility="6"	admiralty-scale	2	0			
12	✔	✖	admiralty-scale:source-reliability="a"	admiralty-scale	0	0			
13	✔	✖	admiralty-scale:source-reliability="b"	admiralty-scale	15	53			
14	✔	✖	admiralty-scale:source-reliability="c"	admiralty-scale	5	2			
15	✔	✖	admiralty-scale:source-reliability="d"	admiralty-scale	1	0			
16	✔	✖	admiralty-scale:source-reliability="e"	admiralty-scale	0	0			
17	✔	✖	admiralty-scale:source-reliability="f"	admiralty-scale	4	2			
1203	✔	✖	adversary:Infrastructure-action="monitoring-active"	adversary	1	0			
1201	✔	✖	adversary:Infrastructure-action="passive-only"	adversary	0	0			

- 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.

PyTaxonomies

- **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**

PyTaxonomies

```
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

```
1  {
2    "name": "List of known public DNS resolvers",
3    "version": 2,
4    "description": "Event contains one or more public DNS resolvers
         as attribute with an IDS flag set",
5    "matching_attributes": [
6      "ip-src",
7      "ip-dst"
8    ],
9    "list": [
10     "8.8.8.8",
11     "8.8.4.4",...]
12 }
```

MISP warning lists

- 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

MISP Project

MISP
Threat Sharing

Why we want to go more modular...

- Ways to extend MISP before modules
  - APIs (PyMISP, MISP API)
    - Works really well
    - **No integration with the UI**
  - Change the core code
    - Have to change the core of MISP, diverge from upstream
    - Needs a deep understanding of MISP internals
    - Let's not beat around the bush: **Everyone hates PHP**

Goals for the module system

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

MISP modules - extending MISP with Python scripts

- Extending MISP with expansion modules with zero customization in MISP.
- A simple ReST API between the modules and MISP allowing auto-discovery of new modules with their features.
- Benefit from existing Python modules in Viper or any other tools.
- MISP modules 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 &‘

MISP modules - Simple REST API mechanism

- 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

```
1  {
2  "type": "expansion",
3  "name": "dns",
4  "meta": {
5    "module-type": [
6      "expansion",
7      "hover"
8    ],
9    "description": "Simple DNS expansion service
       to resolve IP address from MISP
       attributes",
10   "author": "Alexandre Dulaunoy",
11   "version": "0.1"
12 },
13 "mispattributes": {
14   "output": [
15     "ip-src",
16     "ip-dst"
17   ],
18   "input": [
19     "hostname",
20     "domain"
21   ]
22 }
```

MISP modules - configuration in the UI

Server settings

Overview | MISP settings (18) | GnuPG settings (3) | Proxy settings (5) | Security settings (2) | Misc settings (1) | Plugin settings (22) | Diagnostics | Workers

Enrichment

Priority	Setting	Value	Description
Critical	Plugin.Enrichment_services_enable	true	Enable/disable the enrichm
Recommended	Plugin.Enrichment_services_url	http://127.0.0.1	The url used to access the
Recommended	Plugin.Enrichment_services_port	6666	The port used to access the
Recommended	Plugin.Enrichment_cve_enabled	false	Enable or disable the cve m
Recommended	Plugin.Enrichment_dns_enabled	true	Enable or disable the dns m
Recommended	Plugin.Enrichment_sourcecache_enabled	false	Enable or disable the sourc
Recommended	Plugin.Enrichment_sourcecache_archivepath		Set this required module sp
Recommended	Plugin.Enrichment_passivetotal_enabled	true	Enable or disable the passi
Recommended	Plugin.Enrichment_passivetotal_username	alexandre.dulaunoy@circl.lu	Set this required module sp
Recommended	Plugin.Enrichment_passivetotal_password		Set this required module sp

MISP modules - How it's integrated in the UI?

Filters: All | File | Network | Financial | Proposal | Correlation

Value	Comment	Related Events	IDS	Distribution	Actions
microsoft.com			No	Inherit	
google.com		25	No	Inherit	
circl.lu			No	Inherit	

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

dns

Cancel

Filters: All | File | Network | Financial | Proposal | Correlation

Org	Category	Type	Value	Comment	Related Events	IDS
3	Network activity	domain	microsoft.com			No
3	Network activity	domain	google.com		25	No
3	Network activity	domain	circl.lu			No

Enrichment Results

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

Value	Category	Type	IDS	Comment	Actions
23.100.122.175	Network activity	ip-src		Imported via the freetext import.	

Submit

ip-src → ip-dst Change all

Update all comment fields Change all

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

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

- and the response of the dns module:

```
1 {"results": [{"values": ["149.13.33.14"],
2  "types": ["ip-src", "ip-dst"]}]}
```

Creating your module - DNS module

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

def introspection():
    return mispattributes

def version():
    return moduleinfo
```

Testing your module

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

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

- Check if your module is present in the introspection
- curl -s http://127.0.0.1:6666/modules
- If yes, test it directly with MISP or via curl

Code samples (Configuration)

```
# 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):
            misperrors['error'] = 'CIRCL Passive SSL authentication is missing'
            return misperrors

    x = pypssl.PyPSSL(basic_auth=(request['config']['username'], request['config']['password']))
```

Default expansion module set

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

Import modules

- Similar to expansion modules
- Input is a file upload or a text paste
- Output is a list of parsed attributes to be 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

Enrichment Results

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

Event ID	1229
Event UUID	5cc3042c-8bb4-4837-9564-47aca964451a
Event creator org	ORGNAME
Event info	urlhaus test
#Resolved Attributes	14 (2 Objects)

Category	Type	Value	UUID	Tags	IDS	Disable Correlation	Comment	Distribution
Name: virustotal-report **References:** 0								Inherit event
Other	**detection-ratio**: text	10 / 66	adc32dee-4651-41a1-a558-5a1b3f9e4be1		☐	☑	f2b701d43a43315105d649612b2	Inherit event
External analysis	**permalink**: link	https://www.virustotal.com/file/d3fad6911b80be1d64eb88ba23fecbcddc2faa73017b6dbcf78578eff47552ed/analysis/1554403108/	40b3d10d-5e81-48c7-9fe7-be2b2898427b		☐	☐	f2b701d43a43315105d649612b2	Inherit event
ID: 12700 **Name:** file **References:** 11								Inherit event
Payload delivery	**sha256**: sha256	d3fad6911b80be1d64eb88ba23fecbcddc2faa73017b6dbcf78578eff47552ed	5026ab08-8fcd-49e4-a485-b69e92d0295b		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Other	**size-in-bytes**: size-in-bytes	98304	9eef4454-fa6f-4210-a88a-e401599b4f71		☐	☑	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://automotivedreamteam.com/v.exe	eb97b50e-b872-405f-9be9-2dc39459d5e0		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://shopalldogspoop.com/v.exe	a3986a11-4e60-4fb5-ba40-999666402cbc		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://pooperscooperfranchise.com/v.exe	3778ddbd-f7b6-4186-a052-746a3895b9e0		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://cherryhillpooperscoopers.com/v.exe	b804db74-4a62-4cd7-abef-a4b68781411e		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://alldogspoop.net/v.exe	09d672d8-82f8-469f-9c1f-5315fd226d44		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://alldogspoop.mobi/v.exe	48aeba96-b739-47a0-94c1-d583b2b9c4ae		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://alldogspoop.info/v.exe	0f5ad15b-47ed-4772-acb8-d2240a6ed8c3		☑	☐	f2b701d43a43315105d649612b2	Inherit event
Network activity	url	http://alldogspoop.biz/v.exe	90b29df8-d778-4415-8544-5a2fcf53df47		☑	☐	f2b701d43a43315105d649612b2	Inherit event

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)

Q&A

- https://github.com/MISP/misp-modules
- https://github.com/MISP/
- 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: @*MISPProject*

MISP Project

MISP
Threat Sharing

MISP Galaxies

- 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 aproach 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)

(some) Existing galaxies

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

What a cluster looks like

Galaxies

Threat Actor

- Sofacy

Description	The Sofacy Group (also known as APT28, Pawn Storm, Fancy Bear and 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.
Synonyms	APT 28 APT28 Pawn Storm Fancy Bear Sednit TsarTeam TG-4127 Group-4127 STRONTIUM Grey-Cloud
Source	MISP Project
Authors	Alexandre Dulaunoy Florian Roth Thomas Schreck Timo Steffens Various
Country	RU
Refs	https://en.wikipedia.org/wiki/Sofacy_Group

Add new cluster

Attaching clusters to events

- Internally simply using a taxonomy-like tag to attach them to events
- Example: misp-galaxy:threat-actor="Sofacy"
- **Synchronisation works out of the box** with older instances too. They will simply see the tags until they upgrade.
- Currently, as mentioned we rely on the community's contribution of galaxies

Attaching clusters

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

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

Galaxy JSON

- If you want to create a completely new galaxy instead of enriching an existing one

```
1 {
2    "name" : "Threat Actor",
3    "type" : "threat-actor",
4    "description": "Threat actors are characteristics of malicious
         actors (or adversaries) representing a cyber attack threat
         including presumed intent and historically observed
         behaviour.",
5    "version": 1,
6    "uuid": "698774c7-8022-42c4-917f-8d6e4f06ada3"
7 }
```

Cluster JSON

- Clusters contain the meat of the data
- Skeleton structure as follows

```
1  {
2    "values": [
3      {
4        "meta": {},
5        "description": "",
6        "value": "",
7        "related_clusters": [{}],
8      }
9    ]
10 }
```

Cluster JSON value example

```
1   {
2     "meta": {
3       "synonyms": [
4           "APT 28", "APT28", "Pawn Storm", "Fancy Bear",
5           "Sednit", "TsarTeam", "TG-4127", "Group-4127",
6           "STRONTIUM", "Grey-Cloud"
7       ],
8       "country": "RU",
9       "refs": [
10        "https://en.wikipedia.org/wiki/Sofacy_Group"
11      ]
12    },
13    "description": "The Sofacy Group (also known as APT28,
14        Pawn Storm, Fancy Bear and Sednit) is a cyber
15        espionage group believed to have ties to the
16        Russian government. Likely operating since 2007,
17        the group is known to target government, military,
18        and security organizations. It has been
19        characterized as an advanced persistent threat.",
20    "value": "Sofacy"
21  },
```

META BEST PRACTICES

- Reusing existing values such as **complexity, effectiveness, country, possible_issues, colour, motive, impact, refs, synonyms, derivated_from, status, date, encryption, extensions, ransomnotes, cfr-suspected-victims, cfr-suspected-state-sponsor, cfr-type-of-incident, cfr-target-category, kill_chain**.
- Or adding your own meta fields.

META BEST PRACTICES - A SAMPLE

```
1  {
2      "description": "Putter Panda were the subject of an
           extensive report by CrowdStrike, which stated: 'The
           CrowdStrike Intelligence team has been tracking this
           particular unit since2012, 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.'",
3      "meta": {
4        "cfr-suspected-state-sponsor": "China",
5        "cfr-suspected-victims": [
6          "U.S. satellite and aerospace sector"
7        ],
8        "cfr-target-category": [
9          "Private sector",
10         "Government"
11       ],
12       "cfr-type-of-incident": "Espionage",
13       "country": "CN",
14       "refs": [
15         "http://cdn0.vox-cdn.com/assets/4589853/crowdstrike-
               intelligence-report-putter-panda.original.pdf",
16         "https://www.cfr.org/interactive/cyber-operations/putter
               -panda"
```

Galaxy JSON matrix-like

Propose Attribute

example-of-threats

Analysis Initial

Setup | party/candidate registration *(3 items)*	Setup | electoral rolls *(3 items)*	Campaign | campaign IT *(4 items)*
DoS or overload of party/campaign registration, causing them to miss the deadline	Deleting or tampering with voter data	Hacking campaign websites (defacement, DoS)
Fabricated signatures from sponsor	DoS or overload of voter registration system, suppressing voters	Hacking candidate laptops or email accounts
Tampering with registrations	Identity fraud during voter registration	Leak of confidential information
		Misconfiguration of a website

Select Some Options

Can

Galaxy JSON matrix-like

```
1          {
2    "description": "Universal Development and Security Guidelines as
         Applicable to Election Technology.",
3    "icon": "map",
4    "kill_chain_order": {                    \\Tab in the matrix
5          "example-of-threats": [            \\Column in the matrix
6          "setup | party/candidate-registration",
7          "setup | electoral-rolls",
8          "campaign | campaign-IT",
9          "all-phases | governement-IT",
10         "voting | election-technology",
11         "campaign/public-communication | media/press"
12       ]
13   },
14   "name": "Election guidelines",
15   "namespace": "misp",
16   "type": "guidelines",
17   "uuid": "c1dc03b2-89b3-42a5-9d41-782ef726435a",
18   "version": 1
19 }
```

Cluster JSON matrix-like

```
{
    "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": [
        "https://www.ria.ee/sites/default/files/content-editors/kuberturve/cyber_security_of_election_technology.pdf"
      ]
    },
    "uuid": "154c6186-a007-4460-a029-ea23163448fe",
    "value": "DoS or overload of party/campaign registration, causing them to miss the deadline"
}
```

Expressing relation between clusters

- Cluster can be related to one or more clusters using default relationships from MISP objects and a list of tags to classify the relation.

```
        "related": [
        {
          "dest-uuid": "5ce5392a-3a6c-4e07-9df3-9b6a9159ac45",
          "tags": [
            "estimative-language:likelihood-probability=\"likely
                \""
          ],
          "type": "similar"
        }
      ],
      "uuid": "0ca45163-e223-4167-b1af-f088ed14a93d",
      "value": "Putter Panda"
```

PyMISPGalaxies

```
from pymispgalaxies import Clusters
c = Clusters()
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.
```

Q&A

- 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

MISP Project

MISP
Threat Sharing

Objects - or How We Learned to Stop Worrying and Love the Templates

- 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 attriubte 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 prio levels
- misp-attribute: The misp attribute type used as 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

Enforcement of certain keys

- The template also defines which of the added attributes are mandatory
- Requirements are pointed to via their **object relations names**
- We differentiate between two types of rule sets:
  - Required: Everything in this list has to be set in order for the object to validate
  - Required One Of: Any of the attributes in this list will satisfy the requirements

What will the template actually do?

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

Templates as rendered in the UI

Add File Object

Object Template	File v10
Description	File object describing a file with meta-information
Requirements	**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
Meta category	File
Distribution	Inherit event
Comment	

Save	Name :: type	Description	Category	Value
	Md5 :: md5	[Insecure] MD5 hash (128 bits)	Payload delivery	
	Pattern-in-file :: pattern-in-file	Pattern that can be found in the file	Payload installation	
	Sha256 :: sha256	Secure Hash Algorithm 2 (256 bits)	Payload delivery	
	Sha512 :: sha512	Secure Hash Algorithm 2 (512 bits)	Payload delivery	

Templates as rendered in the UI

2018-03-27	**Name:** file **References:** 1			
2018-03-27	Payload delivery	**filename:** filename	putty.exe	+
2018-03-27	Other	**size-in-bytes:** size-in-bytes	774200	+
2018-03-27	Other	**entropy:** float	6.7264597226	+
2018-03-27	Payload delivery	**md5:** md5	b6c12d88eeb910784d75a5e4df954001	+
2018-03-27	Payload delivery	**sha1:** sha1	5ef9515e8fd92a254dd2dcdd9c4b50afa8007b8f	+
2018-03-27	Payload delivery	**sha256:** sha256	81de4319873046761341387 05fc1c21188ad7f27edf6b77a6551aa693194485e	+
2018-03-27	Payload delivery	**sha512:** sha512	e174ecf4fffb36d30c2cc66b37f82877d421244c924d5c9f39f2e0f37d85332b7d107d5ac5bd19cb7ffdcdbdd8b506d488faa30664ef610f62f3970c163cca76	+
2018-03-27	Payload delivery	**malware-sample:**	putty.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

CIRCL / Team MISP Project

info@circl.lu

September 13, 2022

MISP
Threat Sharing

MISP ZeroMQ

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

$\rightarrow$ Operates at global level in MISP

MISP ZeroMQ

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

MISP Live Dashboard

MISP Standard ZMQ

Network activity: 62.102.148.67 | Rotation speed: 30 sec | Zoom level: 15

Attribute.category overtime (hours)

Logs

☐ INFO ☑ WARNING ☑ CRITICAL

Time	Event.id	Attribute.Tag	Attribute.category	Attribute.type	Attribute.value | Attribute.comment
15:07:46	9356	circl:topic="undefined" tlp:white	Network activity	ip-src	99.99.99.99
15:08:36	9356	circl:incident-classification="denial-of-service"	Network activity	ip-src	8.8.8.8
15:08:36	9356	circl:incident-classification="malware" circl:incident-classification="XSS"	Antivirus detection	comment	Comment
15:08:36	9356		Network activity	ip-src	9.9.9.9
15:08:36	9356		Persistence mechanism	text	Just a another test
15:08:36	9356		Internal reference	text	Another text
15:08:36	9356		Financial fraud	phone-number	+221721120220
15:08:36	9356		Network activity	ip-src	62.102.148.67
15:08:36	9356	circl:topic="undefined" tlp:white	Network activity	ip-src	99.99.99.99

MISP-Dashboard - Features

- Subscribe to multiple **ZMQ** MISP instances
- Provides historical geolocalised information
- Present an experimental **Gamification of the platform**
- Shows when and how MISP is used
- Provides real time information showing current threats and activity

MISP-Dashboard: Architecture and development

Setting up the dashboard

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

MISP-Dashboard architecture

Writing your handler

```
# 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,
}

```

```python
# 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-dashboard

- 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

Conclusion

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

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

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

Contributing to the MISP Project

Become part of the community to design, develop

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

Code of Conduct

- The MISP project has a Contributor Covenant Code of Conduct[1].
- The goal of the code of conduct is to foster an **open, fun and welcoming environment**.
- Another important aspect of the MISP projects is to welcome different areas of expertise in information sharing and analysis. The **diversity of the MISP community** is important to make the project useful for everyone.

[1] https://github.com/MISP/MISP/code_of_conduct.md

Reporting a bug, an issue or suggesting features

- 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**.

Reporting security vulnerabilities

- **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).

Automatic integration and testing

- 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 librairies are automatically generated from the JSON via asciidoctor[4].

[3] schema_name.json

[4] example `https://github.com/MISP/misp-galaxy/blob/master/tools/adoc_galaxy.py`

Documentation

- In addition to the automatic generation of documentations from JSON files, we maintain **misp-book**[5] 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[6] 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[7] and also fixing our broken english.

[5] https://github.com/MISP/misp-book
[6] https://github.com/GitbookIO
[7] Topics of interest are analysts best-practices,

Internet-Draft - IETF for MISP formats

- If you want to contribute to our IETF Internet-Draft for the MISP standard, misp-rfc[8] 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[9].
- The process is always MISP implementation → IETF I-D updates.

[8] https://github.com/MISP/misp-rfc
[9] https://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 Project

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

Controllers - components

- Components = reusable code for Controllers
  - Authentication components
  - RestResponse component
  - ACL component
  - Cidr component
  - IOCImport 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

Models - scope

- Controls anything that has to do with:
  - finding subsets of data
  - altering existing data
  - inherited model: AppModel
  - reusable code for models: Behaviours
  - regex, trim

Models - hooking system

- Versatile hooking system
  - manipulate the data at certain stages of execution
  - code can be located in 3 places: Model hook, AppModel hook, behaviour

Model - hooking pipeline (add/edit)

- 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)

Models - misc

- code to handle version upgrades contained in AppModel
- generic cleanup/data migration tools
- centralised redis/pubsub handlers
- (Show example of adding an attribute with trace)

Views - scope and structure

- templates for views
- layouts
- reusable template code: elements
  - attribute list, rows (if reused)
- reusable code: helpers
  - commandhelper (for discussion boards), highlighter for searches, tag colour helper
- views per controller

Views - Types of views and helpers

- ajax views vs normal views
- data views vs normal views vs serialisation in the controller
- sanitisation h()
- creating forms
  - sanitisation
  - CSRF

Views - Generators

- 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

MISP Project

MISP
Threat Sharing

Context

- 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)

Big picture

- 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

Common queries: Recent changes on a timeframe

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)

Administrative tasks

Assyming 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

MISPEvent - Usecase

```python
from pymisp import MISPEvent, EncodeUpdate

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

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

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

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

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

Basics

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

Examples

- **PyMISP needs to be installed (duh)**
- Usage:
  - Create examples/keys.py with the following content

```
misp_url = "https://url-to-your-misp"
misp_key = "<API_KEY>"
misp_verifycert = True
```

- Proxy support:

```
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

Usage

- Basic example

```
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.

MISPEvent, MISPAttribute, MISPObject, MISPSighting...

- **Pythonic** representation of MISP elements
- **Easy manipulation**
  - Load an existing event
  - Update te 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 paramaters 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 paramaters of the MISPObject element (name, comment, ...)
- to_json()
- Can be validated against their template
- Can have default parameters applied to all attributes (i.e. distribution, category, ...)

MISPAttribute - Main entrypoints

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

PyMISP - Tools

- Libraries requiring specfic 3rd party dependencies
- Callable via PyMISP for specific usecases
- Curently implemented:
  - **OpenIOC** to MISP Event
  - MISP to **Neo4J**

PyMISP - Default objects generators

- 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

```
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)
```

Q&A

- https://github.com/MISP/PyMISP
- https://github.com/MISP/
- https://pymisp.readthedocs.io/
- We welcome new functionalities and pull requests.

MISP feeds - A simple and secure approach to generate, select and collect intelligence

Providing ready-to-use threat intelligence in

CIRCL / Team MISP Project
TLP:WHITE

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

MISP Project

MISP Feed - Basics

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)

Feed - Overview

- 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].

Feeds

Generate feed lookup caches or fetch feed data (enabled feeds only)

Cache all feeds | Cache freetext/CSV feeds | Cache MISP feeds | Fetch and store all feed data

« previous | next »

Default feeds | Custom Feeds | All Feeds | Enabled Feeds

Id	Enabled	Name	Feed Format	Provider	Input	Url	Headers	Target	Publish	Delta Merge	Override IDS	Distribution	Tag	Lookup Visible	Caching	Actions
1	✔	CIRCL OSINT Feed MISP	MISP Feed	CIRCL	network	https://www.circl.lu/doc/misp/feed-osint						All communities	CIRCL OSINT Feed	✖	Age: 3m	
2	✔	The Botvrij.eu Data MISP	MISP Feed	Botvrij.eu	network	http://www.botvrij.eu/data/feed-osint						All communities	FEED:KOEN	✖	Not cached	
18	✖	inThreat OSINT Feed MISP	MISP Feed	inThreat	network	https://feeds.inthreat.com/osint/misp/						Your organisation only	osint:source-type="block-or-filter-list"	✖	Not cached	

[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 - Creation using PyMISP `feed generator`

`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 lighweight **TAXII interface**

Feed generator - CONFIGURATION FILE

```
1  url = 'your/misp/url'
2  key = 'YourAPIKey'
3  ssl = True
4  outputdir = 'output_directory'
5
6  filters = {
7      'tag':'tlp:white|feed-export|!privint',
8      'org':'CIRCL'
9  }
10 # the above would generate a feed for all events created by CIRCL,
       tagged tlp:white and/or feed-export but exclude anything
       tagged privint
11
12 valid_attribute_distribution_levels = ['0', '1', '2', '3', '4', '5
       ']
13 # 0: Your Organisation Only
14 # 4: Sharing Group
15 # 5: Inherit Event
16
```

Real-time Feed generator - Purpose

The PyMISP feed generator is great but may be inadequate or ineficient:

- Batch import of attributes/objects
- Data producer doesn't have a MISP instance at hand and only wants to **produce a directly consumable feed**:

Real-time Feed generator - Usage

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

Example:

```
#  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)
```

Real-time Feed generator - Usage (2)

```
#  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

Q&A

- https://github.com/MISP/PyMISP
- https://github.com/MISP/
- We welcome new functionalities and pull requests.

MISP workshop

Introduction into Information Sharing using

Team CIRCL

TLP:WHITE

MISP Project

MISP
Threat Sharing

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[1]

[1] We published the complete guidelines in https://www.x-isac.org/assets/images/guidelines_to_set-up_an_ISAC.pdf

Communities operated by CIRCL

- 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-ISAC[2]
  - **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

[2] https://www.x-isac.org/

Communities operated by CIRCL

- 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**[4])

[4] https://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

CSIRT ADVANCED SERVICES

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

CSIRTs' management of sharing communities for constituent actions:

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

A quick note on compliance...

- Collaboration with Deloitte as part of a CEF project for creating compliance documents
  - Information sharing and cooperation **enabled by GDPR**
  - How MISP enables stakeholders identified by the **NISD** to perform key activities
  - **AIL** and MISP
- For more information: https://github.com/CIRCL/compliance

Bringing different sharing communities together

- 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**

Getting started with building your own sharing community

- Starting a sharing community is **both easy and difficult** at the same time
- Many moving parts and most importantly, you'll be dealing with a **diverse group of people**
- Understanding and working with your constituents to help them face their challenges is key

Getting started with building your own sharing community

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

Running a sharing community using MISP - How to get going?

- Different models for constituents
  - **Connecting to** a MISP instance hosted by a CSIRT
  - **Hosting** their own instance and connecting to CSIRT's MISP
  - **Becoming member** of a sectorial MISP community that is connected to CSIRT's community
- Planning ahead for future growth
  - Estimating requirements
  - Deciding early on common vocabularies
  - Offering services through MISP

Rely on our instincts to 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 possibily 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."

Contextualising the information

- 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**

Choice of vocabularies

- 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

Shared libraries of meta-information (Galaxies)

- 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

False-positive handling

- 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?'
- $\rightarrow$ These objectives can be conflicting (e.g. False-positives have different impacts)

False-positive handling

- **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

Managing sub-communities

- Often within a community **smaller bubbles of information sharing will form**
- For example: Within a national private sector sharing community, specific community for financial institutions
- Sharing groups serve this purpose mainly
- As a CSIRT running a national community, consider bootstraping these sharing communities
- Organisations can of course self-organise, but you are the ones with the know-how to get them started

Managing sub-communities

- 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!

Get in touch if you need some help to get started

- 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/MISPProject

MISP and Decaying of Indicators

An indicator scoring method and ongoing imple-

Team CIRCL

info@circl.lu

September 13, 2022

MISP
Threat Sharing

Expiring IOCs: Why and How?

Indicators - Problem Statement

- **Sharing information** about threats **is crucial**
- Organisations are sharing more and more

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

Date	Unique Org
2013	17
2014	43
2015	82
2016	105
2017	118
2018	125
2019-10	135

```
{
    "distribution": [1, 2, 3]
}
```

Indicators - Problem Statement

- 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)

$\rightarrow$ Can be partially solved with *Taxonomies*

Indicators - Problem Statement

- 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*

Requirements to enjoy the decaying feature in MISP

- 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 - Refresher

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**

Organisations opt-in - setting a level of confidence

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*

Taxonomies - Refresher (1)

Taxonomies

« previous | 1 | 2 | next »

Id	Namespace	Description	Version	Enabled	Required	Active Tags	Actions
181	workflow	Workflow support language is a common language to support intelligence analysts to perform their analysis on data and information.	9	Yes	☐	**27** / 26 (enable all)	
180	vocabulaire-des-probabilites-estimatives	Ce vocabulaire attribue des valeurs en pourcentage à certains énoncés de probabilité	2	Yes	☐	**5** / 5	
179	threats-to-dns	An overview of some of the known attacks related to DNS as described by Torabi, S., Boukhtouta, A., Assi, C., & Debbabi, M. (2018) in Detecting Internet Abuse by Analyzing Passive DNS Traffic: A Survey of Implemented Systems. IEEE Communications Surveys & Tutorials, 1–1. doi:10.1109/comst.2018.2849614	1	No	☐	**0** / 18	
178	targeted-threat-index	The Targeted Threat Index is a metric for assigning an overall threat ranking score to email messages that deliver malware to a victim's computer. The TTI metric was first introduced at SecTor 2013 by Seth Hardy as part of the talk "RATastrophe: Monitoring a Malware Menagerie" along with Katie Kleemola and Greg Wiseman.	2	Yes	☐	**11** / 11	

- Tagging is a simple way to attach a classification to an *Event* or an *Attribute*
- Classification must be globally used to be efficient

Taxonomies - Refresher (2)

ADMIRALTY-SCALE Taxonomy Library

Id	127
Namespace	admiralty-scale
Description	The Admiralty Scale or Ranking (also called the NATO System) is used to rank the reliability of a source and the credibility of an information. Reference based on FM 2-22.3 (FM 34-52) HUMAN INTELLIGENCE COLLECTOR OPERATIONS and NATO documents.
Version	4
Enabled	Yes (disable)

« previous next »

Filter

Tag	Expanded	Numerical value	Events	Attributes	Tags	Action
admiralty-scale:information-credibility="1"	Information Credibility: Confirmed by other sources	100	6	0	admiralty-scale:information-credibility="1"	
admiralty-scale:information-credibility="2"	Information Credibility: Probably true	75	21	1	admiralty-scale:information-credibility="2"	
admiralty-scale:information-credibility="3"	Information Credibility: Possibly true	50	16	5	admiralty-scale:information-credibility="3"	
admiralty-scale:information-credibility="4"	Information Credibility: Doubtful	25	2	0	admiralty-scale:information-credibility="4"	
admiralty-scale:information-credibility="5"	Information Credibility: Improbable	0	1	0	admiralty-scale:information-credibility="5"	
admiralty-scale:information-credibility="6"	Information Credibility: Truth cannot be judged	50	9	2	admiralty-scale:information-credibility="6"	
admiralty-scale:source-reliability="a"	Source Reliability: Completely reliable	100	1	0	admiralty-scale:source-reliability="a"	
admiralty-scale:source-reliability="b"	Source Reliability: Usually reliable	75	21	76	admiralty-scale:source-reliability="b"	
admiralty-scale:source-reliability="c"	Source Reliability: Fairly reliable	50	9	8	admiralty-scale:source-reliability="c"	
admiralty-scale:source-reliability="d"	Source Reliability: Not usually reliable	25	2	0	admiralty-scale:source-reliability="d"	
admiralty-scale:source-reliability="e"	Source Reliability: Unreliable	0	0	0	admiralty-scale:source-reliability="e"	
admiralty-scale:source-reliability="f"	Source Reliability: Reliability cannot be judged	50	10	7	admiralty-scale:source-reliability="f"	
admiralty-scale:source-reliability="g"	Source Reliability: Deliberatly deceptive	0	N/A	N/A		+

→ Cherry-pick allowed *Tags*

Taxonomies - Refresher (3)

- Some taxonomies have `numerical_value`
  - → Can be used to prioritise *Attributes*

Description	Value
Completely reliable	100
Usually reliable	75
Fairly reliable	50
Not usually reliable	25
Unreliable	0
Reliability cannot be judged	50 ?
Deliberatly deceptive	0 ?

Description	Value
Confirmed by other sources	100
Probably true	75
Possibly true	50
Doubtful	25
Improbable	0
Truth cannot be judged	50 ?

Scoring Indicators: Our solution

$$\texttt{score}(\texttt{Attribute}) = \texttt{base_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$$

Where,

- $\texttt{score} \in [0, +\infty$
- $\texttt{base_score} \in [0, 100]$
- `decay` is a function defined by model's parameters controlling decay speed
- `Attribute` Contains *Attribute*'s values and metadata (*Taxonomies*, *Galaxies*, ...)
- `Model` Contains the *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

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: 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)

$$\texttt{score}(\texttt{Attribute}) = \texttt{base_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$$

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

$$\texttt{base_score} = \omega_{tg} \cdot \mathit{tags} + \omega_{sc} \cdot \mathit{source_confidence}$$

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 implentation ignores source_confidence:

$\rightarrow$ base_score $=$ *tags*

Tag	Computation			Result
	Eff. Ratio		numerical_value	
admiralty-scale:source-reliability="Completely reliable"	0.50	*	100.00	50.00
phishing:psychological-acceptability="high"	0.50	*	75.00	37.50
				87.50

$\rightarrow$ The base_score can be use to prioritize attribute based on their attached context and source

Scoring Indicators: decay speed (1)

$$\texttt{score}(\texttt{Attribute}) = \texttt{base_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$$

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

Scoring Indicators: putting it all toghether

$\rightarrow$ `decay rate` is **re-initialized upon sighting** addition, or said differently, the `score` is reset to its base score as new *sightings* are applied.

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

- $\tau =$ `lifetime`
- $\delta =$ `decay speed`

Implementation in MISP: Models definition

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

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

- Parameters: `lifetime, decay_rate, threshold`
- `base_score`
- `default base_score`
- formula
- associate *Attribute* types
- creator organisation

Implementation in MISP: Models Types

Multiple model types are available

- **Default Models**: Models created and shared by the community. Available from `misp-decaying-models` repository[3].
  - → Not editable
- **Organisation Models**: Models created by a user belonging to an organisation
  - These models can be hidden or shared to other organisation
  - → Editable

[3] https://github.com/MISP/misp-decaying-models.git

Implementation in MISP: Index

Decaying Models

« previous next »

All Models | My Models | Shared Models | Default Models

ID	Organization	Usable to everyone	Name	Description	Parameters { }	Formula	# Assigned Types	Version	Enabled	Actions
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	9	1	✔	
85	1	✘	NIDS Simple Decaying Model MISP	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	✔	

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

« previous next »

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

3 not having numerical value

Default basescore 80

Taxonomies	Weight
admiralty-scale	
source-reliability	31
information-credibility	30
priority-level	
priority-level	53
retention	
retention	0
estimative-language	
likelihood-probability	0
confidence-in-analytic-judgment	0
misp	
confidence-level	0
threat-level	0
automation-level	0
phishing	
state	0
psychological-acceptability	0
Excluded	

Placeholder for `Organisation source confidence`

Example

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

Computation steps

Tag	Computation			Result
	Eff. Ratio		Value	
priority-level:baseline-minor	0.46	*	25.00	11.62
admiralty-scale:source-reliability="d"	0.27	*	25.00	6.80

Implementation in MISP: simulation tool

Simulate *Attributes* with different *Models*

Implementation in MISP: API query body

/attributes/restSearch

```
{
    "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)
- ...

Creating a new decay algorithm (2)

```
1  <?php
2  include_once 'Base.php';
3
4  class Polynomial extends DecayingModelBase
5  {
6      public const DESCRIPTION = 'The description of your new
       decaying algorithm';
7
8      public function computeScore($model, $attribute, $base_score,
       $elapsed_time)
9      {
10         // algorithm returning a numerical score
11     }
12
13     public function isDecayed($model, $attribute, $score)
14     {
15         // algorithm returning a boolean stating
16         // if the attribute is expired or not
17     }
18 }
19 ?>
20
```

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

September 13, 2022

MISP
Threat Sharing

Outline of the presentation

- 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?

Indicators lifecycle - Problem Statement

- **Sharing information** about threats **is crucial**
- Organisations are sharing more and more

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

Date	Unique Org
2013	17
2014	43
2015	82
2016	105
2017	118
2018	125
2019-10	135

```
{
    "distribution": [1, 2, 3]
}
```

Indicators lifecycle - Problem Statement

- 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

$\rightarrow$ Can be partially solved with *Taxonomies*

Indicators lifecycle - Problem Statement

- 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 - Refresher (1)

Sightings add a **temporal context** to indicators.

- *Sightings* can be used to represent that you saw the IoC
- **Usecase:** Continuous feedback loop MISP $\leftrightarrow$ IDS

Sightings - Refresher (2)

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 - Refresher (1)

Taxonomies

« previous | 1 | 2 | next »

Id ↑	Namespace	Description	Version	Enabled	Required	Active Tags	Actions
181	workflow	Workflow support language is a common language to support intelligence analysts to perform their analysis on data and information.	9	Yes	☐	**27** / 26 (enable all)	− 👁 🗑
180	vocabulaire-des-probabilites-estimatives	Ce vocabulaire attribue des valeurs en pourcentage à certains énoncés de probabilité	2	Yes	☐	**5** / 5	− 👁 🗑
179	threats-to-dns	An overview of some of the known attacks related to DNS as described by Torabi, S., Boukhtouta, A., Assi, C., & Debbabi, M. (2018) in Detecting Internet Abuse by Analyzing Passive DNS Traffic: A Survey of Implemented Systems. IEEE Communications Surveys & Tutorials, 1–1. doi:10.1109/comst.2018.2849614	1	No	☐	**0** / 18	+ 👁 🗑
178	targeted-threat-index	The Targeted Threat Index is a metric for assigning an overall threat ranking score to email messages that deliver malware to a victim's computer. The TTI metric was first introduced at SecTor 2013 by Seth Hardy as part of the talk "RATastrophe: Monitoring a Malware Menagerie" along with Katie Kleemola and Greg Wiseman.	2	Yes	☐	**11** / 11	− 👁 🗑

- *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

Id	127
Namespace	admiralty-scale
Description	The Admiralty Scale or Ranking (also called the NATO System) is used to rank the reliability of a source and the credibility of an information. Reference based on FM 2-22.3 (FM 34-52) HUMAN INTELLIGENCE COLLECTOR OPERATIONS and NATO documents.
Version	4
Enabled	Yes (disable)

« previous next »

Filter

Tag	Expanded	Numerical value	Events	Attributes	Tags	Action
admiralty-scale:information-credibility="1"	Information Credibility: Confirmed by other sources	100	6	0	admiralty-scale:information-credibility="1"	
admiralty-scale:information-credibility="2"	Information Credibility: Probably true	75	21	1	admiralty-scale:information-credibility="2"	
admiralty-scale:information-credibility="3"	Information Credibility: Possibly true	50	16	5	admiralty-scale:information-credibility="3"	
admiralty-scale:information-credibility="4"	Information Credibility: Doubtful	25	2	0	admiralty-scale:information-credibility="4"	
admiralty-scale:information-credibility="5"	Information Credibility: Improbable	0	1	0	admiralty-scale:information-credibility="5"	
admiralty-scale:information-credibility="6"	Information Credibility: Truth cannot be judged	50	9	2	admiralty-scale:information-credibility="6"	
admiralty-scale:source-reliability="a"	Source Reliability: Completely reliable	100	1	0	admiralty-scale:source-reliability="a"	
admiralty-scale:source-reliability="b"	Source Reliability: Usually reliable	75	21	76	admiralty-scale:source-reliability="b"	
admiralty-scale:source-reliability="c"	Source Reliability: Fairly reliable	50	9	8	admiralty-scale:source-reliability="c"	
admiralty-scale:source-reliability="d"	Source Reliability: Not usually reliable	25	2	0	admiralty-scale:source-reliability="d"	
admiralty-scale:source-reliability="e"	Source Reliability: Unreliable	0	0	0	admiralty-scale:source-reliability="e"	
admiralty-scale:source-reliability="f"	Source Reliability: Reliability cannot be judged	50	10	7	admiralty-scale:source-reliability="f"	
admiralty-scale:source-reliability="g"	Source Reliability: Deliberatly deceptive	0	N/A	N/A		+

→ Cherry-pick allowed *Tags*

Taxonomies - Refresher (3)

- Some taxonomies have a `numerical_value`
- Allows concepts to be used in an mathematical expression
  - → Can be used to prioritise IoCs

`admirality-scale` taxonomy[1]

Description	Value
Completely reliable	100
Usually reliable	75
Fairly reliable	50
Not usually reliable	25
Unreliable	0
Reliability cannot be judged	50
Deliberatly deceptive	0

Description	Value
Confirmed by other sources	100
Probably true	75
Possibly true	50
Doubtful	25
Improbable	0
Truth cannot be judged	50

[1] https://github.com/MISP/misp-taxonomies/blob/master/admiralty-scale/machinetag.json

Taxonomies - Refresher (3)

`admirality-scale` taxonomy[2]

Description	Value
Completely reliable	100
Usually reliable	75
Fairly reliable	50
Not usually reliable	25
Unreliable	0
Reliability cannot be judged	50 ?
Deliberatly deceptive	0 ?

Description	Value
Confirmed by other sources	100
Probably true	75
Possibly true	50
Doubtful	25
Improbable	0
Truth cannot be judged	50 ?

→ Users can override tag `numerical_value`

[2] https://github.com/MISP/misp-taxonomies/blob/master/admiralty-scale/machinetag.json

Scoring Indicators: Our solution

$$\texttt{score}(\texttt{Attribute}) = \texttt{base_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$$

- $\texttt{base_score}(\texttt{Attribute, Model})$
  - Initial score of the *Attribute* only considering the context (*Attribute's type*, *Tags*)

- $\texttt{decay}(\texttt{Model, time})$
  - Function composed of the **lifetime** and **decay speed**
  - Decreases the `base_score` over time

Scoring Indicators: Our solution

$$\texttt{score}(\texttt{Attribute}) = \texttt{base_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$$

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

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

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

Implementation in MISP: Models Types

Two types of model are available

- **Default Models**: Created and shared by the community. Coming from `misp-decaying-models` repository[3].
  - → Not editable
- **Organisation Models**: Created by a user on MISP
  - ▶ Can be hidden or shared to other organisation
  - → Editable

[3] https://github.com/MISP/misp-decaying-models.git

Implementation in MISP: Index

Decaying Models

« previous next »

All Models | My Models | Shared Models | Default Models

ID	Organization	Usable to everyone	Name	Description	Parameters { }	Formula	# Assigned Types	Version	Enabled	Actions
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	9	1	✔	
85	1	✘	NIDS Simple Decaying Model MISP	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	✔	

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

« previous next »

Standard CRUD operations: View, update, add, create, delete, enable, export, import

Implementation in MISP: Fine tuning tool

Home | Event Actions | Galaxies | Input Filters | Global Actions | Sync Actions | Administration | Audit | MISP

Import Decaying Model
Add Decaying Model
Decaying Tool
List Decaying Models

Decaying Of Indicator Fine Tuning Tool

Show All Types | Show MISP Objects | Search Attribute Type

Attribute Type	Category	Model ID
aba-rtn	Financial fraud	
authentihash	Payload delivery	
bank-account-nr	Financial fraud	
bic	Financial fraud	
bin	Financial fraud	
bro	Network activity	10 11
btc	Financial fraud	11
cc-number	Financial fraud	
cdhash	Payload delivery	
community-id	Network activity	
domain	Network activity	
domain|ip	Network activity	10 84
email-attachment	Payload delivery	
email-dst	Network activity	11
email-src	Payload delivery	
filename	Payload delivery	
filename|authentihash	Payload delivery	
filename|impfuzzy	Payload delivery	
filename|imphash	Payload delivery	
filename|md5	Payload delivery	13
filename|pehash	Payload delivery	13
filename|sha1	Payload delivery	13

Polynomial

Lifetime 3 days
Decay speed 2.3
Cutoff threshold 30

Expire after (lifetime)	1 days and 7 hours
Score halved after (Half-life)	0 day and 6 hours

Adjust base score | Simulate this model

Phishing model | Simple model to rapidly decay | Edit

All available models | My models | Default models

	ID	Model Name	Org ID	Description	Formula	Parameters: Lifetime	Decay speed	Threshold	Default basescore	Basescore config	Settings	# Types	Enabled	Action
✓	29	Phishing model	1	Simple model to rapidly decay phishing website.	Polynomial	3	2.3	30	80	estimative-language 0.5 phishing 0.5		9	✓	Load model

Configure models: Create, modify, visualise, perform mapping

Implementation in MISP: base_score tool

Search Taxonomy

3 not having numerical value

Default basescore 80

Taxonomies	Weight
admiralty-scale	
source-reliability	31
information-credibility	30
priority-level	
priority-level	53
retention	
retention	0
estimative-language	
likelihood-probability	0
confidence-in-analytic-judgment	0
misp	
confidence-level	0
threat-level	0
automation-level	0
phishing	
state	0
psychological-acceptability	0
Excluded	

Placeholder for `Organisation source confidence`

Example

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

Computation steps

Tag	Computation			Result
	Eff. Ratio		Value	
priority-level:baseline-minor	0.46	*	25.00	11.62
admiralty-scale:source-reliability="d"	0.27	*	25.00	6.80

Implementation in MISP: simulation tool

Simulate decay on *Attributes* with different *Models*

Implementation in MISP: API query body

/attributes/restSearch

```
1  {
2      "includeDecayScore": 1,
3      "includeFullModel": 0,
4      "excludeDecayed": 0,
5      "decayingModel": [85],
6      "modelOverrides": {
7          "threshold": 30
8      }
9      "score": 30,
10 }
11
```

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

Team CIRCL

info@circl.lu

September 13, 2022

MISP
Threat Sharing

DFIR and MISP digital evidences

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

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

Benefits of using MISP

- LE can leverage the long-standing experience in information sharing and **bridge their use-cases** with MISP's information sharing mechanisms.
- **Accessing existing MISP information sharing communities** by receiving actionable information from CSIRT/CERT networks or security researchers.
- **Bridging LE communities with other communities**. Sharing groups can be created (and managed) cross-sectors to support specific use-cases.
- The **MISP standard** is a flexible format which can be extended by users using the MISP platform. A MISP object template can be created in under 30 minutes, allowing users to rapidly share information using their own data-models with existing communities.

Challenges and implementations

- Standard sharing mechanism for forensic cases
  - MISP allows for the efficient **collaborative** analysis of digital evidences
  - Correlation on certain attributes
- Importing disk images and file system data activity (`Mactime`)
  - Development of an adaptable import tool: From Mactime to MISP `Mactime object`
- Create, modify and visualise the timeline of events
  - Development of a flexible timeline system at the event level

Forensic import (MISP 2.4.98)

Select	Filepath	File Size	Activity Type	Time Accessed	Permissions
☐	..c.r/rrwxrwxrwx	Xxx			00
☑	/DCIM/111___06/_MG_0125.JPG(deleted)	3541836	Accessed	Sun Jun 02 2013 00:00:00	r/rrwxrwxrwx
☑	/DCIM/111___06/_MG_0125.JPG(deleted)	3541836	Created,Modified	Sun Jun 02 2013 15:42:32	r/rrwxrwxrwx
☑	/DCIM/111___06/IMG_0126.JPG	2255115	Created,Modified	Sun Jun 02 2013 15:42:46	r/rrwxrwxrwx
☐	/DCIM/CANONMSC/M0111.CTG	884	Created,Modified	Sun Jun 02 2013 15:44:08	r/rrwxrwxrwx
☐	/CANON_DC(Volume	0	Modified	Sun Jun 02 2013 16:33:04	r/rrwxrwxrwx
☑	/DCIM/111___06/IMG_0126.JPG	2255115	Accessed	Sat Feb 06 2016 00:00:00	r/rrwxrwxrwx

- 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

CIRCL / Team MISP Project

MISP Threat Sharing

MISP Project

MISP API reworked

- 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

Goals we've set for ourselves

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

Export API's reimagined

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

Export API's reimagined

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

Performance tuning

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

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

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

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 accuractely** before feeding them to our protective tools
- Different contexts (APT targeting me? Persisting techniques?) - lifecycle management
- Use several queries / boolean operators to select the slice of data most relevant for the task

CLI tools for the CLI God, automation for the automation throne

- Open up commonly used system management tasks to the CLI
  - sync servers/feeds
  - caching feeds
  - Password resets
  - Server settings
  - Bruteforce protection resets
  - Enrichment
  - Worker management
- Goal was also to move away from the often malfunctioning scheduler and have cron friendly CLI scripts

So what does all of this look like in practice?

Demo time!

Plans for the future

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

Best Practices in Threat Intelligence

Gather, document, analyse and contextualise in-

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

Objectives

- Learn how to use MISP to support common OSINT gathering use-cases often used by SOC, CSIRTs and CERTs
  - Use practical exercise examples[1]
  - 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

[1] https://gist.github.com/adulau/8c1de4806 0e259799d3397b83b0eec4f

(Threat) Intelligence

- **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

Meta information and contextualisation 1/2

- 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 available[2]
- **Review existing practices of tagging in your sharing community, reuse practices, and improve context**

[2] https://www.misp-project.org/taxonomies.html

Meta information and contextualisation 2/2

- **When information cannot be expressed in triple tags format** (*namespace:predicate=value*), MISP use Galaxies
- **Galaxies** contain a huge set of common libraries[3] 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

Estimative Probability

- **Words of Estimative Probability**[4] propose clear wording while estimating probability of occurence from an event
- A MISP taxonomy called **estimative-language**[5] proposes an applied model to tag information in accordance with the concepts of Estimative Probability

[4] https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/books-and-monographs/sherman-kent-and-the-board-of-national-estimates-collected-essa6words.html

[5] https://www.misp-project.org/taxonomies.html

Reliability, credibility, and confidence

- The **Admiralty Scale**[6] (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[7] is available
- US DoD **JP 2-0, Joint Intelligence**[8] includes an appendix to express confidence in analytic judgments
- A MISP predicate in estimative-language called confidence-in-analytic-judgment[9] is available

---

[6] https://www.ijlter.org/index.php/ijlter/article/download/494/234, *US Army Field Manual 2-22.3, 2006*

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

[8] http://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp2_0.pdf, page 114

[9] https://www.misp-project.org/taxonomies.html

Adding attributes/objects to an event

- 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

How to select the right object?

There are more than 150 MISP object[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*.

[10] https://www.misp-project.org/objects.html

MICROBLOG OBJECT

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.

FILE OBJECT

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.

Add File Object

Object Template	File v17
Description	File object describing a file with meta-information
Requirements	**Required one of**: filename, size-in-bytes, authentihash, ssdeep, md5, sha1, sha224, sha256, sha384, sha512, sha512/224, sha512/256, tlsh, pattern-in-file, x509-fingerprint-sha1, malware-sample, attachment, path, fullpath
Meta category	File
Distribution	Inherit event
Comment	

References

- 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 Project

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

User settings - a long overdue feature

- 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/iglocska/MISP
  - via ssh: git remote add my_fork gitgithub.com:iglocska/MISP.git
- Generally a good idea to work on a new branch: git checkout -b dev_exercise
- Enable debug in MISP

Implementation

- Storage:
  - Single key/value table for all settings
  - Each user should be able to set a single instance of a key
  - Values could possible 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

Checklist

- 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 / index

- 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

- 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

MISP Project

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 composited 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

Where does the module system come into play?

- 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 formatś 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 simplicityś sake
- **pipe separated values**
- VALUE | TYPE | DESCRIPTION | REFERENCE | ACTION

Nibbler format - caveats

- 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

Mapping the types to MISP

- 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

Export module skeleton

```
<?php
class NibblerExport
{
    public $additional_params = array();
    public function handler(
        $data, $options = array()
    ) {}
    public function header(
        $options = array()
    ) {}
    public function footer() {}
    public function separator() {}
}
```

Additional parameters

```
public $additional_params = array(
    'flatten' => 1
);
```

Adding our mapping

```
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')
);
```

Writing the start of the output

```
public function header($options = array())
{
    return sprintf(
        "# Nibbler rules generated by MISP at %s\n",
        date('Y-m-d H:i:s')
    );
}
```

Footer function - how should the output end?

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

What separates the chunks?

```
public function separator()
{
    return "\n";
}
```

The actual legwork, the handler

```
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 '';
}
```

Building an optional internal converter function

```
private function __convertAttribute($attribute, $eve
{
  if (empty($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);
}
```

Handling the simple case

```
$result[] = sprintf(
  '%s|%s|%s|%s|%s',
  $this->__escapeSpecialChars($attribute['value']),
  $this->__mapping[$attribute['type']],
  $event['uuid'],
  $this->__escapeSpecialChars($event['info']),
  'ALERT'
);
```

Handling the case for composites

```
$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'
  );
}
```

Putting it together

```
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);
}
```

Adding the function that decides on the action

```
private function __decideOnAction($attributeTags)
{
  foreach($attributeTags as $attributeTag) {
    if (
      $attributeTag['Tag']['name'] ===
        'nibbler:block'
    ) {
      return 'BLOCK';
    }
  }
  return 'ALERT';
}
```

Finalising the export module… The escaping function

```
private function __escapeSpecialChars($value)
{
  $value = preg_replace(
    "/\r|\n/", "##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
- [ ] Bookmark query
- [x] Show result
- [ ] Skip SSL validation

HTTP headers

```
Authorization: ArSxnHf20foSapnOSyxfrljMdl9oLDnvmqvHK97q
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

September 13, 2022

MISP
Threat Sharing

Outline of the presentation

- Present the features available for Sharing *galaxy clusters*
- Look at the internals of what changed in the datamodel and MISP's behaviors

MISP Galaxy 2.0

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 clusters

Default *Galaxy cluster*

- Coming from the `misp-galaxy` repository[1]
- 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

[1] https://github.com/MISP/misp-galaxy

MISP Galaxy 2.0 - Comparison with prior version

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

MISP Galaxy 2.0 - Others changes

- *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

Galaxy cluster elements

Hasn't been touched: Still a key-value stored. But new feature have been added[2]

Tabular view

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

« previous | 1 | 2 | 3 | next » | last »

Tabular view | JSON view

Key ↓	Value	Actions
created	2018-10-01T00:00:00Z	
definition.access_privilege.0.privilege_action	CISAUSES	
definition.access_privilege.0.privilege_scope.permitted_nationalities.0	USA	
definition.access_privilege.0.privilege_scope.permitted_nationalities.1	AUS	
definition.access_privilege.0.privilege_scope.permitted_nationalities.2	CAN	
definition.access_privilege.0.privilege_scope.permitted_nationalities.3	GBR	
definition.access_privilege.0.privilege_scope.permitted_nationalities.4	NZL	

[2] Will be included in next release

Galaxy cluster elements

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

Tabular view | JSON view | + Add JSON as cluster's elements


```
{
    "created": [
        "2018-10-01T00:00:00Z"
    ],
    "definition": {
        "access_privilege": [
            {
                "privilege_action": [
                    "CISAUSES"
                ],
                "privilege_scope": {
                    "permitted_nationalities": [
                        [
                            "USA"
                        ],
```

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/

MISP Project

MISP
Threat Sharing

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 granulars distribution settings:

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

Sharing Group

Id	11
Uuid	5e4bf73c-05dc-4586-840f-5848a5e38e14
Name	Banking sector in Europe
Releasability	Banks located in Europe
Description	Everything banking
Selectable	✔
Created by	Training

Organisations

Name	Local	Extend
Training	✔	✔
A-FUNKY-HUNGARIAN-BANK.hu	✔	✔
AFB	✔	✖
Italian Bank	✔	✖
NCSC-NL	✖	✖

Instances

Name	Url	All orgs
Local instance	https://iglocska.eu	✖
https://iglocska.eu	https://iglocska.eu	✖

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

Sharing in MISP: Distribution

Data layer

Data layer: Naming conventions

- 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

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

Usecase: Encode incidents / events / reports / ...

IoT malware - Gafgyt.Gen28 (active) - 20190220 - 20190222

Event ID	178
UUID	5c6d21e5-bb60-47b7-b892-42e6950d2111
Creator org	CIRCL
Owner org	Training
Creator user	andras.iklody@circl.lu
Tags	tlp:white x, osint:source-type="automatic-collection" x, circl:incident-classification="malware" x, adversary:infrastructure-action="take-down" x
Date	2019-02-20
Threat Level	Low
Analysis	Completed
Distribution	All communities
Info	IoT malware - Gafgyt.Gen28 (active) - 20190220 - 20190222
Published	Yes (2020-11-28 07:53:39)
#Attributes	2601 (296 Objects)
First recorded change	2019-02-20 09:46:24
Last change	2020-10-10 07:36:28
Modification map	
Sightings	0 (0) - restricted to own organisation only.

Data layer: Event building blocks - Base

Data layer: Events

```
1  {
2      "date": "2019-02-20",
3      "info": "IoT malware - Gafgyt.Gen28 (active)",
4      "uuid": "5c6d21e5-bb60-47b7-b892-42e6950d2111",
5      "analysis": "2",
6      "timestamp": "1602315388",
7      "distribution": "3",
8      "sharing_group_id": "0",
9      "threat_level_id": "3",
10     "extends_uuid": "",
11     "Attribute": [...],
12     "Object": [...],
13     "EventReport": [...],
14     "Tag": [...],
15     "Galaxy": [...]
16 }
```

Data layer: Attributes

Attributes are individual data points, indicators or supporting data

> **Purpose**: Individual data point. Can be an indicator or supporting data.
>
> **Usecase**: Domain, IP, link, sha1, attachment, ...

« previous | next » | view all

Filters: All | File | Network | Financial | Proposal | Correlation

Date	Org	Category	Type	Value	Comment	Related Events	IDS	Distribution	Actions
2016-02-23		Network activity	domain	microsoft.com			No	Inherit	
2016-02-23		Network activity	domain	google.com		25	No	Inherit	
2016-02-23		Network activity	domain	circl.lu			No	Inherit	
2016-02-23		Network activity	ip-src	23.100.122.175	Derived from microsoft.com via the dns enrichment module.		No	Inherit	

Data layer: Event building blocks - Raw data

Data layer: Attributes

```
{
    "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

- **Purpose**: Groups Attributes that are intrinsically linked together
- **Usecase**: File, person, credit-card, x509, device, ...

2018-03-27	**Name:** file **References:** 1			
2018-03-27	Payload delivery	**filename:** filename	putty.exe	+
2018-03-27	Other	**size-in-bytes:** size-in-bytes	774200	+
2018-03-27	Other	**entropy:** float	6.7264597226	+
2018-03-27	Payload delivery	**md5:** md5	b6c12d88eeb910784d75a5e4df954001	+
2018-03-27	Payload delivery	**sha1:** sha1	5ef9515e8fd92a254dd2dcdd9c4b50afa8007b8f	+
2018-03-27	Payload delivery	**sha256:** sha256	81de4319873046761341387 05fc1c21188ad7f27edf6b77a6551aa6931944 85e	+
2018-03-27	Payload delivery	**sha512:** sha512	e174ecf4fffb36d30c2cc66b37f82877d421244c924d5c9f39f2e0f37d85332b 7d107d5ac5bd19cb7ffdcdbdd8b506d488faa30664ef610f62f3970c163cca7 6	+
2018-03-27	Payload delivery	**malware-sample:**	putty.exe	+

Data layer: Event building blocks - Data composition

Data layer: MISP Objects

```
1  {
2      "name": "elf-section",
3      "meta-category": "file",
4      "description": "Object describing a sect...",
5      "template_uuid": "ca271f32-1234-4e87-b240-6b6e882de5de",
6      "template_version": "4",
7      "uuid": "ab5f0c85-5623-424c-bc03-d79841700d74",
8      "timestamp": "1550655984",
9      "distribution": "5",
10     "sharing_group_id": "0",
11     "comment": "",
12     "first_seen": null,
13     "last_seen": null,
14     "ObjectReference": [],
15     "Attribute": [...]
16 }
```

Data layer: Object references

Object references are the relationships between individual building blocks

- **Purpose**: Allows to create relationships between entities, thus creating a graph where they are the edges and entities are the nodes.
- **Usecase**: Represent behaviours, similarities, affiliation, ...

Data layer: Object references

```
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

Data layer: Sightings

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

Purpose: Allows to add temporality to the data.

Usecase: Record activity or occurence, perform IoC expiration, ...

```
{
    "org_id": "1",
    "date_sighting": "1573722432",
    "uuid": "5dcd1940-5de8-4462-93dd-12a2a5e38e14",
    "source": "",
    "type": "0",
    "attribute_uuid": "5da97b59-9650-4be2-9443-2194a5e38e14"
}
```

Data layer: Event reports

Event reports are supporting data for analysis to describe **events**, **processes**, ect

- **Purpose**: Supporting data point to describe events or processes
- **Usecase**: Encode reports, provide more information about the Event, …

Data layer: Event building blocks - Collaboration & intelligence

Data layer: Event reports

```
1 {
2     "uuid": "076e240b-5a76-4a8b-9eab-cfff551993dd",
3     "event_id": "2127",
4     "name": "Event report (1607362986)",
5     "content": "...",
6     "distribution": "5",
7     "sharing_group_id": "0",
8     "timestamp": "1607362986"
9 }
```

Data layer: Event building blocks - Full

Context layer

Context layer: Naming conventions

- 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: Naming conventions

- Context layer
  - **Galaxies** are container copmosed 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"`

Context layer: Tags

Simple free-text labels

```
1 {
2     "name": "Android malware",
3     "colour": "#22681c",
4     "exportable": true,
5     "numerical_value": null,
6 }
```

Context layer: Taxonomies

Simple label standardised on common set of vocabularies

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

Usecase: Provide classification such as: TLP, Confidence, Source, Workflows, Event type, ...

Tag	Events	Attributes	Tags
workflow:state="complete"	11	0	workflow:state="complete"
workflow:state="draft"	0	0	workflow:state="draft"
workflow:state="incomplete"	55	10	workflow:state="incomplete"
workflow:state="ongoing"	0	0	workflow:state="ongoing"

Context layer: Taxonomies

```
{
  "Taxonomy": {
    "namespace": "admiralty-scale",
    "description": "The Admiralty Scale or Ranking (also called
        the NATO System)...",
    "version": "6",
    "exclusive": false,
  },
  "entries": [
     {
       "tag": "admiralty-scale:information-credibility=\"1\"",
       "expanded": "Information Credibility: Confirmed by other
           sources",
       "numerical_value": 100,
       "exclusive_predicate": true,
     },
     ...
  ]
}
```

Collections of **galaxy clusters**

Threat Actor galaxy

Galaxy ID	8
Name	Threat Actor
Namespace	misp
UUID	698774c7-8022-42c4-917f-8d6e4f06ada3
Description	Threat actors are characteristics of malicious actors (or adversaries) representing a cyber attack threat including presumed intent and historically observed behaviour.
Version	3

« previous next »

All | Default | Custom 0 | My Clusters | Deleted | View Fork Tree | View Galaxy Relationships

apt29 Filter

ID	Published	Value	Synonyms	Owner Org	Creator Org	Default	Activity	#Events	#Relations	Description	Distribution	Actions
7059	N/A	APT 29	Dukes, Group 100, Cozy Duke, CozyDuke, EuroAPT, CozyBear, CozyCar, Cozer, Office Monkeys	MISP	MISP	✔		0	0 0	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 since at least 2008 to collect intelligence in support of foreign and security policy decision-making. The Dukes show unusual confidence in their ability to continue successfully compromising their	All communities	

Context layer: Galaxy clusters

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

- **Purpose**: Enable description of complex high-level information for classification
- **Usecase**: Extensively describe elements such as threat actors, countries, technique used, ...

Threat Actor :: APT 29

Cluster ID	2805
Name	APT 29
Parent Galaxy	Threat Actor
Description	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...
Published	**No**
Default	Yes
Version	190
UUID	b2056ff0-00b9-482e-b11c-c771daa5f28a
Collection UUID	7cdff317-a673-4474-84ec-4f1754947823
Source	MISP Project
Authors	Alexandre Dulaunoy, Florian Roth, Thomas Schreck, Timo Steffens, Various
Distribution	**All communities**
Owner Organisation	MISP
Creator Organisation	MISP
Connector tag	misp-galaxy:threat-actor="APT 29"
Events	0
Forked From	
Forked By	

Galaxy cluster elements: Tabular view

Tabular view | JSON view

Key ↓	Value	Actions
attribution-confidence	50	
cfr-suspected-state-sponsor	Russian Federation	
cfr-suspected-victims	United States	
cfr-suspected-victims	China	
cfr-suspected-victims	New Zealand	

Galaxy cluster elements: JSON view

Tabular view | JSON view | + Add JSON as cluster's elements


```
{
    "attribution-confidence": [
        "50"
    ],
    "cfr-suspected-state-sponsor": [
        "Russian Federation"
    ],
    "cfr-suspected-victims": [
        "United States",
        "China",
        "New Zealand",
        "Ukraine",
```

Context layer: Galaxy clusters


```
{
    "uuid": "5eda0a53-1d98-4d01-ae06-40da0a00020f",
    "type": "fellowship-characters",
    "value": "Aragorn wielding Anduril",
    "tag_name": "misp-galaxy:fellowship-characters=\"c3fe907a-6a36
        -4cd1-9456-dcdf35c3f907\"",
    "description": "The Aragorn character wielding Anduril",
    "source": "Middle-earth universe by J. R. R. Tolkien",
    "authors": null,
    "version": "1591347795",
    "distribution": "0",
    "sharing_group_id": null,
    "default": false,
    "extends_uuid": "5eda0117-1e14-4b0a-9e26-34aff331dc3b",
    "extends_version": "1591345431",
    "GalaxyElement": [...],
    "GalaxyClusterRelation": [...]
}
```

Context layer: Galaxies & Galaxy clusters

- MISP integrates MITRE's Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) and similar **Galaxy Matrix**
- MISP terminology of these matrixes: **Galaxy Matrix**

Pre Attack - Attack Pattern | **Enterprise Attack - Attack Pattern** | Mobile Attack - Attack Pattern

0 — 11 — Show all

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Exfiltration	Command and control
Spearphishing Attachment	Scripting	Screensaver	File System Permissions Weakness	Process Hollowing	Securityd Memory	Password Policy Discovery	AppleScript	Data from Information Repositories	Exfiltration Over Alternative Protocol	Standard Application Layer Protocol
Spearphishing via Service	Command-Line Interface	Login Item	AppCert DLLs	Code Signing	Input Capture	System Network Configuration Discovery	Distributed Component Object Model	Data from Removable Media	Exfiltration Over Command and Control Channel	Communication Through Removable Media
Trusted Relationship	User Execution	Trap	Application Shimming	Rootkit	Bash History	Process Discovery	Pass the Hash	Man in the Browser	Data Compressed	Custom Command and Control Protocol
Replication Through Removable Media	Regsvcs/Regasm	System Firmware	Scheduled Task	NTFS File Attributes	Exploitation for Credential Access	Network Share Discovery	Exploitation of Remote Services	Data Staged	Automated Exfiltration	Multi-Stage Channels
Exploit Public-Facing Application	Trusted Developer Utilities	Registry Run Keys / Start Folder	Startup Items	Exploitation for Defense Evasion	Private Keys	Peripheral Device Discovery	Remote Desktop Protocol	Screen Capture	Scheduled Transfer	Remote Access Tools
Spearphishing Link	Windows Management Instrumentation	LC_LOAD_DYLIB Addition	New Service	Network Share Connection Removal	Brute Force	Account Discovery	Pass the Ticket	Email Collection	Data Encrypted	Uncommonly Used Port
Valid Accounts	Service Execution	LSASS Driver	Sudo Caching	Process Doppelgänging	Password Filter DLL	System Information Discovery	Windows Remote Management	Clipboard Data	Exfiltration Over Other Network Medium	Multilayer Encryption
Supply Chain Compromise	CMSTP	Rc.common	Process Injection	Disabling Security Tools	Two-Factor Authentication Interception	System Network Connections Discovery	Windows Admin Shares	Video Capture	Exfiltration Over Physical Medium	Domain Fronting
Drive-by Compromise	Control Panel Items	Authentication Package	Bypass User Account Control	Timestomp	LLMNR/NBT-NS Poisoning	Network Service Scanning	Remote Services	Audio Capture	Data Transfer Size Limits	Data Obfuscation
Hardware Additions	Dynamic Data Exchange	Component Firmware	Extra Window Memory Injection	Modify Registry	Credentials in Files	File and Directory Discovery	Taint Shared Content	Data from Network Shared Drive		Connection Proxy
	Source	Windows Management Instrumentation Event Subscription	Setuid and Setgid	Indicator Removal from Tools	Forced Authentication	Security Software Discovery	Application Deployment Software	Data from Local System		Commonly Used Port
	Space after Filename	Change Default File	Launch Daemon	Hidden Window	Keychain	System Service Discovery	Third-party Software	Automated Collection		Data Encoding

Galaxy JSON matrix-like

```
1  {
2    "description": "Universal Development and Security Guidelines as
         Applicable to Election Technology.",
3    "icon": "map",
4    "kill_chain_order": {                    \\Tab in the matrix
5        "example-of-threats": [              \\Column in the matrix
6        "setup | party/candidate-registration",
7        "setup | electoral-rolls",
8        "campaign | campaign-IT",
9        "all-phases | governement-IT",
10       "voting | election-technology",
11       "campaign/public-communication | media/press"
12     ]
13   },
14   "name": "Election guidelines",
15   "namespace": "misp",
16   "type": "guidelines",
17   "uuid": "c1dc03b2-89b3-42a5-9d41-782ef726435a",
18   "version": 1
19 }
```

Cluster JSON matrix-like

```
{
    "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": [
        "https://www.ria.ee/sites/default/files/content-editors/kuberturve/cyber_security_of_election_technology.pdf"
      ]
    },
    "uuid": "154c6186-a007-4460-a029-ea23163448fe",
    "value": "DoS or overload of party/campaign registration, causing them to miss the deadline"
}
```

Expressing relation between clusters

- Cluster can be related to one or more clusters using default relationships from MISP objects and a list of tags to classify the relation.

```
        "related": [
        {
          "dest-uuid": "5ce5392a-3a6c-4e07-9df3-9b6a9159ac45",
          "tags": [
            "estimative-language:likelihood-probability=\"likely\"
                "
          ],
          "type": "similar"
        }
      ],
      "uuid": "oca45163-e223-4167-b1af-f088ed14a93d",
      "value": "Putter Panda"
```

Acknowledgements

- 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

MISP Project

MISP
Threat Sharing

Dashboard in MISP

- User configurable simple dashboard interface
- **Visualise**, **aggregate** and **track** data important to you
- Brand new feature, still undergoing reworks

The internals of awidget

- **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/

The view layer of a widget

- 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/

Widget behaviours

- 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**

Exercise module: simple Whoami

- 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

Exercise module: simple Whoami

```php
<?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;
  }
}
```

Meta information

- **$title**: The name of the widget
- **$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)

The handler

```
1  public function handler($user, $options = array())
2  {
3    $this->Log = ClassRegistry::init('Log');
4    $entries = $this->Log->find('all', array(
5      'recursive' => -1,
6      'conditions' => array(
7        'action' => 'login', 'user_id' => $user['id']
8      ),
9      'order' => 'id desc',
10     'limit' => 5,
11     'fields' => array('created', 'ip')
12   ));
13   foreach ($entries as &$entry) {
14     $entry = $entry['Log']['created'] . ' --- ' .
15     (
16       empty($entry['Log']['ip']) ?
17       'IP not logged' :
18        $entry['Log']['ip']
19     );
20   }
21   return array(
22     array('title' => 'Email', 'value' => $user['email']),
23     array(
24       'title' => 'Role', 'value' => $user['Role']['name']
25     ),
26     array(
27       'title' => 'Organisation',
28       'value' => $user['Organisation']['name']
29     ),
30     array(
31       'title' => 'IP', 'value' => $_SERVER['REMOTE_ADDR']
32     ),
33     array('title' => 'Last logins', 'value' => $entries)
34   );
35 }
```

Result

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 Project

ABOUT CIRCL

circl.lu
Computer Incident
Response Center
LUXEMBOURG

- 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.

MISP and CIRCL

- 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**.

The aim of this presentation

- 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**

Development based on practical user feedback

- 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, likelyhood and occurences.
  - **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

Why was it so simplistic?

- 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

The growing need to contextualise data

- 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 be able to prioritise

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...

Our initial failures

- 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

TLP AMBER
TLP:AMBER
Threat tlp:Amber
tlp-amber
tlp::amber
tlp:amber

How we ended up tackling the issue more successfuly

- 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

Tag	Events	Attributes	Tags
workflow:state="complete"	11	0	workflow:state="complete"
workflow:state="draft"	0	0	workflow:state="draft"
workflow:state="incomplete"	55	10	workflow:state="incomplete"
workflow:state="ongoing"	0	0	workflow:state="ongoing"

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

Galaxy ID	373
Name	Ransomware
Namespace	misp
Uuid	3f44af2e-1480-4b6b-9aa8-f9bb21341078
Description	Ransomware galaxy based on...
Version	4

Value ↓	Synonyms
.CryptoHasYou.	
777	Sevleg
7ev3n	7ev3n-HONE$T

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, ...

LIST OF KNOWN IPV4 PUBLIC DNS RESOLVERS

Id	89
Name	List of known IPv4 public DNS resolvers
Description	Event contains one or more public IPv4 DNS resolvers as attribute with an IDS flag set
Version	20181114
Type	string
Accepted attribute types	ip-src, ip-dst, domain|ip
Enabled	**Yes** (disable)

Values

1.0.0.1
1.1.1.1
1.11.71.4

Warning: Potential false positives

List of known IPv4 public DNS resolvers
Top 1000 website from Alexa
List of known google domains

[1] https://github.com/MISP/misp-warninglists

More complex data-structures for a modern age

- 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

Filters: All | File | Network | Financial | Proposal | Correlation | Warnings | Include deleted attributes | Show context fields

Date	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events
2018-09-28		Name: bank-account References: 0							
2018-09-28		Other	status-code: text	A - Active	+	Add		☐	
2018-09-28		Other	report-code: text	STR Suspicious Transaction Report	+	Add		☐	
2018-09-28		Other	personal-account-type: text	A - Business	+	Add		☐	
2018-09-28		Financial fraud	swift: bic	HASEHKHH	+	Add		☑	3849 11320 11584
2018-09-28		Financial fraud	account: bank-account-nr	788796894883	+	Add		☑	
2018-09-28		Other	account-name: text	FANY SILU CO. LIMITED	+	Add		☑	
2018-09-28		Other	currency-code: text	USD	+	Add		☐	

Continuous feedback loop

- 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

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

/attributes/restSearch

```
{
    "returnFormat": "netfilter",
    "enforceWarninglist": 1,
    "tags": {
      "NOT": [
        "tlp:white",
        "type:OSINT"
      ],
      "OR": [
        "misp-galaxy:threat-actor=\"Sofacy\"",
        "misp-galaxy:sector=\"Chemical\""
      ],
    }
}
```

Synchronisation filters

- 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** [3] by Francesco Bigarella from ING
  - **Election guidelines** [4] by NIS Cooperation Group

[3] https://www.misp-project.org/galaxy.html#_attck4fraud
[4] https://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

Pre Attack - Attack Pattern | **Enterprise Attack - Attack Pattern** | Mobile Attack - Attack Pattern

0 — 11 — Show all

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Exfiltration	Command and control
Spearphishing Attachment	Scripting	Screensaver	File System Permissions Weakness	Process Hollowing	Securityd Memory	Password Policy Discovery	AppleScript	Data from Information Repositories	Exfiltration Over Alternative Protocol	Standard Application Layer Protocol
Spearphishing via Service	Command-Line Interface	Login Item	AppCert DLLs	Code Signing	Input Capture	System Network Configuration Discovery	Distributed Component Object Model	Data from Removable Media	Exfiltration Over Command and Control Channel	Communication Through Removable Media
Trusted Relationship	User Execution	Trap	Application Shimming	Rootkit	Bash History	Process Discovery	Pass the Hash	Man in the Browser	Data Compressed	Custom Command and Control Protocol
Replication Through Removable Media	Regsvcs/Regasm	System Firmware	Scheduled Task	NTFS File Attributes	Exploitation for Credential Access	Network Share Discovery	Exploitation of Remote Services	Data Staged	Automated Exfiltration	Multi-Stage Channels
Exploit Public-Facing Application	Trusted Developer Utilities	Registry Run Keys / Start Folder	Startup Items	Exploitation for Defense Evasion	Private Keys	Peripheral Device Discovery	Remote Desktop Protocol	Screen Capture	Scheduled Transfer	Remote Access Tools
Spearphishing Link	Windows Management Instrumentation	LC_LOAD_DYLIB Addition	New Service	Network Share Connection Removal	Brute Force	Account Discovery	Pass the Ticket	Email Collection	Data Encrypted	Uncommonly Used Port
Valid Accounts	Service Execution	LSASS Driver	Sudo Caching	Process Doppelgänging	Password Filter DLL	System Information Discovery	Windows Remote Management	Clipboard Data	Exfiltration Over Other Network Medium	Multilayer Encryption
Supply Chain Compromise	CMSTP	Rc.common	Process Injection	Disabling Security Tools	Two-Factor Authentication Interception	System Network Connections Discovery	Windows Admin Shares	Video Capture	Exfiltration Over Physical Medium	Domain Fronting
Drive-by Compromise	Control Panel Items	Authentication Package	Bypass User Account Control	Timestomp	LLMNR/NBT-NS Poisoning	Network Service Scanning	Remote Services	Audio Capture	Data Transfer Size Limits	Data Obfuscation
Hardware Additions	Dynamic Data Exchange	Component Firmware	Extra Window Memory Injection	Modify Registry	Credentials in Files	File and Directory Discovery	Taint Shared Content	Data from Network Shared Drive		Connection Proxy
	Source	Windows Management Instrumentation Event Subscription	Setuid and Setgid	Indicator Removal from Tools	Forced Authentication	Security Software Discovery	Application Deployment Software	Data from Local System		Commonly Used Port
	Space after Filename	Change Default File	Launch Daemon	Hidden Window	Keychain	System Service Discovery	Third-party Software	Automated Collection		Data Encoding

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

$$\texttt{score}(\texttt{Attribute}) = \texttt{base_score}(\texttt{Attribute, Model}) \bullet \texttt{decay}(\texttt{Model, time})$$

Where,

- $\texttt{score} \in [0, 100]$
- $\texttt{base_score} \in [0, 100]$
- `decay` is a function defined by model's parameters controlling decay speed
- `Attribute` Contains *Attribute*'s values and metadata (*Taxonomies*, *Galaxies*, ...)
- `Model` Contains the *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"
        }
```

Implementation in MISP: Index

Decaying Models

« previous next »

All Models | My Models | Shared Models | Default Models

ID	Organization	Usable to everyone	Name	Description	Parameters { }	Formula	# Assigned Types	Version	Enabled	Actions
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	9	1	✔	
85	1	✘	NIDS Simple Decaying Model MISP	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	✔	

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

« previous next »

View, update, add, create, delete, enable, export, import

Implementation in MISP: Fine tuning tool

Home | Event Actions | Galaxies | Input Filters | Global Actions | Sync Actions | Administration | Audit | MISP

Import Decaying Model
Add Decaying Model
Decaying Tool
List Decaying Models

Decaying Of Indicator Fine Tuning Tool

Show All Types | Show MISP Objects | Search Attribute Type

Attribute Type	Category	Model ID
aba-rtn	Financial fraud	
authentihash	Payload delivery	
bank-account-nr	Financial fraud	
bic	Financial fraud	
bin	Financial fraud	
bro	Network activity	10 11
btc	Financial fraud	11
cc-number	Financial fraud	
cdhash	Payload delivery	
community-id	Network activity	
domain	Network activity	
domain|ip	Network activity	10 84
email-attachment	Payload delivery	
email-dst	Network activity	11
email-src	Payload delivery	
filename	Payload delivery	
filename|authentihash	Payload delivery	
filename|impfuzzy	Payload delivery	
filename|imphash	Payload delivery	
filename|md5	Payload delivery	13
filename|pehash	Payload delivery	13
filename|sha1	Payload delivery	13

Polynomial

Lifetime	3	days
Decay speed	2.3	
Cutoff threshold	30	

Expire after (lifetime)	1 days and 7 hours
Score halved after (Half-life)	0 day and 6 hours

Adjust base score | Simulate this model

Phishing model | Simple model to rapidly decay | Edit

All available models | My models | Default models

	ID	Model Name	Org ID	Description	Formula	Parameters: Lifetime	Decay speed	Threshold	Default basescore	Basescore config	Settings	# Types	Enabled	Action
✓	29	Phishing model	1	Simple model to rapidly decay phishing website.	Polynomial	3	2.3	30	80	estimative-language 0.5 phishing 0.5		9	✓	Load model

Create, modify, visualise, perform mapping

Implementation in MISP: base_score tool

Search Taxonomy

3 not having numerical value

Default basescore 80

Taxonomies	Weight
admiralty-scale	
source-reliability	31
information-credibility	30
priority-level	
priority-level	53
retention	
retention	0
estimative-language	
likelihood-probability	0
confidence-in-analytic-judgment	0
misp	
confidence-level	0
threat-level	0
automation-level	0
phishing	
state	0
psychological-acceptability	0
Excluded	

Placeholder for `Organisation source confidence`

Example

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

Computation steps

Tag	Computation			Result
	Eff. Ratio		Value	
priority-level:baseline-minor	0.46	*	25.00	11.62
admiralty-scale:source-reliability="d"	0.27	*	25.00	6.80

Implementation in MISP: simulation tool

Simulate *Attributes* with different *Models*

Implementation in MISP: API query body

/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/iglocska
- 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

Turning data into actionable intelligence

Advanced features in MISP supporting your analysts and tools

CIRCL / Team MISP Project

MISP Project

The aim of this presentation

- Why is **contextualisation** important?
- What options do we have in MISP?
- How can we **leverage** this in the end?

The growing need to contextualise data

- Contextualisation became more and more important as we as a community matured
  - **Growth and diversification** 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 a need to be able to prioritise
- These help with filtering your TI based on your **requirements**...
- ...as highlighted by Pasquale Stirparo *Your Requirements Are Not My Requirements*

Objectives

- Some main objectives we want to achieve when producing data
  - Ensure that the information is **consumable** by everybody
  - That it is **useful** to the entire target audience
  - The data is **contextualised** for it to be understood by everyone
- What we ideally want from our data
  - We want to be able to **filter** data for different use-cases
  - We want to be able to get as much knowledge out of the data as possible
  - We want to know where the data is from, how it got there, why we should care

Different layers of context

- Context added by analysts / tools
- Data that tells a story
- Encoding analyst knowledge to automatically leverage the above

Context added by analysts / tools

Expressing why data-points matter

- An IP address by itself is barely ever interesting
- We need to tell the recipient / machine why this is relevant
- All data in MISP has a bare minimum required context
- We differentiate between indicators and supporting data

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?

Tagging and taxonomies

- Simple labels
- Standardising on vocabularies
- Different organisational/community cultures require different nomenclatures
- Triple tag system - taxonomies
- JSON libraries that can easily be defined without our intervention

Tag	Events	Attributes	Tags
workflow:state="complete"	11	0	workflow:state="complete"
workflow:state="draft"	0	0	workflow:state="draft"
workflow:state="incomplete"	55	10	workflow:state="incomplete"
workflow:state="ongoing"	0	0	workflow:state="ongoing"

Galaxies

- 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

Galaxy ID	373
Name	Ransomware
Namespace	misp
Uuid	3f44af2e-1480-4b6b-9aa8-f9bb21341078
Description	Ransomware galaxy based on...
Version	4

Value ↓	Synonyms
.CryptoHasYou.	
777	Sevleg
7ev3n	7ev3n-HONE$T

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** [1] by Francesco Bigarella from ING
  - **Election guidelines** [2] by NIS Cooperation Group

[1] https://www.misp-project.org/galaxy.html#_attck4fraud
[2] https://www.misp-project.org/galaxy.html#_election_guidelines

Data that tells a story

More complex data-structures for a modern age

- Atomic attributes were a great starting point, but lacking in many aspects
- **MISP objects**[3] 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

[3] https://github.com/MISP/misp-objects

Supporting specific datamodels

Filters: All | File | Network | Financial | Proposal | Correlation | Warnings | Include deleted attributes | Show context fields

Date	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events
2018-09-28		Name: bank-account References: 0							
2018-09-28		Other	status-code: text	A - Active	+	Add		☐	
2018-09-28		Other	report-code: text	STR Suspicious Transaction Report	+	Add		☐	
2018-09-28		Other	personal-account-type: text	A - Business	+	Add		☐	
2018-09-28		Financial fraud	swift: bic	HASEHKHH	+	Add		☑	3849 11320 11584
2018-09-28		Financial fraud	account: bank-account-nr	788796894883	+	Add		☑	
2018-09-28		Other	account-name: text	FANY SILU CO. LIMITED	+	Add		☑	
2018-09-28		Other	currency-code: text	USD	+	Add		☐	

Continuous feedback loop

- Data shared was **frozen in time**
- All we had was a creation/modification timestamp
- Improved tooling and willingness allowed us to create a **feedback loop**
- Lead to the introduction of the **Sighting system**
- Signal the fact of an indicator sighting...
- ...as well as **when** and **where** it was sighted
- Vital component for IoC **lifecycle management**

Continuous feedback loop (2)

A brief history of time - Adding temporality to our data

- As Andreas said - no time based aspect was painful
- Recently introduced **`first_seen`** and **`last_seen`** data points
- Along with a complete integration with the **UI**
- Enables the **visualisation** and **adjustment** of indicators timeframes

The various ways of encoding analyst knowledge to automatically leverage our TI

False positive handling

- Low quality / false positive prone information being shared
- Lead to **alert-fatigue**
- Exclude organisation xy out of the community?
- FPs are often obvious - **can be encoded**
- **Warninglist system**[4] aims to do that
- Lists of well-known indicators which are often false-positives like RFC1918 networks, ...

LIST OF KNOWN IPV4 PUBLIC DNS RESOLVERS

Id	89
Name	List of known IPv4 public DNS resolvers
Description	Event contains one or more public IPv4 DNS resolvers as attribute with an IDS flag set
Version	20181114
Type	string
Accepted attribute types	ip-src, ip-dst, domain|ip
Enabled	Yes (disable)

Values

1.0.0.1
1.1.1.1
1.11.71.4

Warning: Potential false positives

List of known IPv4 public DNS resolvers
Top 1000 website from Alexa
List of known google domains

[4] https://github.com/MISP/misp-warninglists

Making use of all this context

- Providing advanced ways of querying data
  - Unified export APIs
  - Incorporating all contextualisation options into **API filters**
  - Allowing for an **on-demand** way of **excluding potential false positives**
  - Allowing users to easily **build their own** export modules feed their various tools

Example query

/attributes/restSearch

```
{
    "returnFormat": "netfilter",
    "enforceWarninglist": 1,
    "tags": {
      "NOT": [
        "tlp:white",
        "type:OSINT"
      ],
      "OR": [
        "misp-galaxy:threat-actor=\"Sofacy\"",
        "misp-galaxy:sector=\"Chemical\""
      ],
    }
}
```

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

Pre Attack - Attack Pattern | **Enterprise Attack - Attack Pattern** | Mobile Attack - Attack Pattern

0 — 11 — Show all

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Exfiltration	Command and control
Spearphishing Attachment	Scripting	Screensaver	File System Permissions Weakness	Process Hollowing	Securityd Memory	Password Policy Discovery	AppleScript	Data from Information Repositories	Exfiltration Over Alternative Protocol	Standard Application Layer Protocol
Spearphishing via Service	Command-Line Interface	Login Item	AppCert DLLs	Code Signing	Input Capture	System Network Configuration Discovery	Distributed Component Object Model	Data from Removable Media	Exfiltration Over Command and Control Channel	Communication Through Removable Media
Trusted Relationship	User Execution	Trap	Application Shimming	Rootkit	Bash History	Process Discovery	Pass the Hash	Man in the Browser	Data Compressed	Custom Command and Control Protocol
Replication Through Removable Media	Regsvcs/Regasm	System Firmware	Scheduled Task	NTFS File Attributes	Exploitation for Credential Access	Network Share Discovery	Exploitation of Remote Services	Data Staged	Automated Exfiltration	Multi-Stage Channels
Exploit Public-Facing Application	Trusted Developer Utilities	Registry Run Keys / Start Folder	Startup Items	Exploitation for Defense Evasion	Private Keys	Peripheral Device Discovery	Remote Desktop Protocol	Screen Capture	Scheduled Transfer	Remote Access Tools
Spearphishing Link	Windows Management Instrumentation	LC_LOAD_DYLIB Addition	New Service	Network Share Connection Removal	Brute Force	Account Discovery	Pass the Ticket	Email Collection	Data Encrypted	Uncommonly Used Port
Valid Accounts	Service Execution	LSASS Driver	Sudo Caching	Process Doppelgänging	Password Filter DLL	System Information Discovery	Windows Remote Management	Clipboard Data	Exfiltration Over Other Network Medium	Multilayer Encryption
Supply Chain Compromise	CMSTP	Rc.common	Process Injection	Disabling Security Tools	Two-Factor Authentication Interception	System Network Connections Discovery	Windows Admin Shares	Video Capture	Exfiltration Over Physical Medium	Domain Fronting
Drive-by Compromise	Control Panel Items	Authentication Package	Bypass User Account Control	Timestomp	LLMNR/NBT-NS Poisoning	Network Service Scanning	Remote Services	Audio Capture	Data Transfer Size Limits	Data Obfuscation
Hardware Additions	Dynamic Data Exchange	Component Firmware	Extra Window Memory Injection	Modify Registry	Credentials in Files	File and Directory Discovery	Taint Shared Content	Data from Network Shared Drive		Connection Proxy
	Source	Windows Management Instrumentation Event Subscription	Setuid and Setgid	Indicator Removal from Tools	Forced Authentication	Security Software Discovery	Application Deployment Software	Data from Local System		Commonly Used Port
	Space after Filename	Change Default File	Launch Daemon	Hidden Window	Keychain	System Service Discovery	Third-party Software	Automated Collection		Data Encoding

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 available **context**
  - Taxonomies
  - Sightings
  - type of each indicator
  - Creation date
  - ...

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"
        }
```

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/iglocska
- 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 pow-

CIRCL / Team MISP Project

http://www.misp-standard.org/
Twitter: *@MISPProject*

MISP Project

MISP
Threat Sharing

MISP Standard

- 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

Standard - MISP core format

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.

MISP object template format

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.

MISP galaxy 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**.

SightingDB format

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.

Internet-Draft - IETF for MISP formats and MISP standard

- 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 → IETF I-D updates.
- Then published standards in misp-standard.org.

[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

MISP Project

MISP CLI functionalities

- 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

CLI DOCUMENTATION

- https://path.to.your.misp/events/automation

Administering the background workers via the API.

You can start/stop and view the bacground 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 Password [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

Automation via crontab

- 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

MISP Threat Sharing

MISP Project

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

Hardware considerations

- 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

Hardware considerations - continues

- 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)

Hardware considerations - continues

- 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`

An Introduction to Workflows in MISP

MISP - Threat Sharing

CIRCL / Team MISP Project

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

MISP Project

MISP
Threat Sharing

Content of the presentation

- MISP Workflows fundamentals
- Getting started
- Design of the system & how it can be extended

What problems are we trying to tackle

- Initial idea came during GeekWeek7.5[1]
- Needs:
  - Prevent default MISP behaviors
  - Hook specific actions to run callbacks
- Use-cases:
  - Prevent publication of events not meeting some criterias
  - Prevent querying thrid-party services (e.g. virustotal) with sensitive information
  - Send notifications in a chat rooms
  - And much much more..

[1] Workshop organized by the Canadian Cyber Center

Workflow - Fundamentals

Simplistic overview of a Workflow in action

1. An **action** happens in MISP
2. If there is an **enabled** Workflow for that **action**, run it
3. If all went fine, MISP **continue** to perform the action
   - The operation can potentially be cancelled by `blocking` modules

Terminology

- **workflow**: Sequence of all operations (nodes) to be executed. Basically the whole graph.
- **execution path**: A path composed of nodes
- **trigger**: Starting point of a workflow. Triggers are called when specific actions happen in MISP
  - A trigger can only have one workflow and vice-versa

Workflow execution process

Typical execution process:

1. An action happens in MISP
2. The workflow associated to the trigger is ran
3. Execution result?
   - `success`: Continue the action
   - `failure` | `blocked`: Cancel the action

Example for Event publish:

1. An Event is about to be published
2. MISP executes the workflow listening to the `event-publish` trigger
   - **success**: Continue the publishing action
   - **failure** | `blocked`: Stop publishing and log the reason

Blocking and non-blocking Workflows

Currently 2 types of workflows:

- **Blocking**: Completion of the action can be prevented
  - If a **blocking module** blocks the action
  - If a **blocking module** raises an exception
- **Non-blocking**: Workflow execution outcome has no impact
  - **Blocking modules** can still stop the execution

Execution context

- Workflows can be triggered by **any users**
- Workflows can be triggered by actions done via the **UI** or **API**
- However, the user for which the workflow executes has:
  - The `site-admin` permission
  - Is from the `MISP.host_org_id`
- Ensures data is processed regardless of ownership and access: **no ACL**

Classes of Workflow modules

3 classes of modules

- **action**: Allow to executes functions, callbacks or scripts
  - Can stop execution
  - e.g. Webhook, block the execution, perform enrichments, ...
- **logic**: Allow to redirect the execution flow.
  - IF condition, fork the blocking execution into a non-blocking one, ...
- **blueprint**: Allow to reuse composition of modules
  - Can save subworkflows and its module's configuration

Sources of Workflow modules

3 sources of action modules

- Built-in **default** modules
  - Part of the MISP codebase
  - **app/Model/**`WorkflowModules/action/[module_name].php`
- User-defined **custom** modules
  - Written in PHP
  - Can extend existing default modules
  - Can use MISP's built-in functionalities (restsearch, enrichment, push to zmq, ...)
  - Faster and easier to implement new complex behaviors
  - **app/Lib/**`WorkflowModules/action/[module_name].php`

Sources of Workflow modules

3 sources of action modules

- Modules from the **enrichment service**
  - **Default** and **custom** modules
  - From the *misp-module* misp-module
  - Written in Python
  - Can use any python libraries
  - New *misp-module* module type: `action`

→ Both the PHP and Python systems are **plug-and-play**

Triggers currently available

Currently 8 triggers can be hooked. 3 being **blocking**.

Trigger name	Scope	Trigger overhead	Description	Run counter	Blocking Workflow	MISP Core format	Workflow ID	Last Update	Enabled	Actions
Attribute After Save	attribute	high	This trigger is called after an Attribute has been saved in the database	58	×	✓	160	2022-07-29 06:58:11	✓	
Enrichment Before Query	others	low	This trigger is called just before a query against the enrichment service is done	841	✓	✓	162	2022-07-29 08:32:32	✓	
Event After Save	event	medium	This trigger is called after an Event has been saved in the database	11	×	✓	175	2022-07-29 08:37:23	✓	
Event Publish	event	low	This trigger is called just before a MISP Event starts the publishing process	1	✓	✓	180	2022-07-29 12:14:10	✓	
Object After Save	object	high	This trigger is called after an Object has been saved in the database	35	×	✓	161	2022-07-28 13:59:37	×	
Post After Save	post	low	This trigger is called after a Post has been saved in the database	36	×	×	176	2022-07-28 13:59:51	✓	
User After Save	user	low	This trigger is called after a user has been saved in the database	55	×	×	159	2022-07-28 14:00:03	✓	
User Before Save	user	low	This trigger is called just before a user is save in the database	42	✓	×	158	2022-07-28 14:00:32	✓	

Workflow - Getting started

Getting started with workflows (1)

Review MISP settings:

1. Make sure `MISP.background_jobs` is turned on
2. Make sure workers are up-and-running and healthy
3. Turn the setting `Plugin.Workflow_enable` on

4. [optional:misp-module] Turn the setting `Plugin.Action_services_enable` on

Getting started with workflows (2)

If you wish to use action modules from `misp-module`, make sure to have:

- The latest update of `misp-module`
  - There should be an `action_mod` module type in `misp-modules/misp_modules/modules`
- Restarted your `misp-module` application

```
# This command should show all ‘action‘ modules
$ curl -s http://127.0.0.1:6666/modules | \
jq ’.[] | select(.meta."module-type"[] | contains("action")) |
{name: .name, version: .meta.version}’
```

Getting started with workflows (3)

1. Go to the list of modules
   - `Administration > Workflows > List Modules`
   - or `/workflows/moduleIndex`
2. Make sure **default** modules are loaded
3. [optional:misp-module] Make sure **misp-module** modules are loaded

Creating a workflow with the editor

1. Go to the list of triggers `Administration > Workflows`
2. Enable and edit a `trigger` from the list
3. Drag an `action` module from the side panel to the canvas
4. From the `trigger` output, drag an arrow into the `action`'s input (left side)
5. Execute the action that would run the trigger and observe the effect!

Trigger name	Scope	Trigger overhead	Description	Run counter	Blocking Workflow	MISP Core format	Workflow ID	Last Update	Enabled	Actions
Attribute After Save	attribute	high	This trigger is called after an Attribute has been saved in the database	58	×	✓	160	2022-07-29 06:58:11	✓	
Enrichment Before Query	others	low	This trigger is called just before a query against the enrichment service is done	841	✓	✓	162	2022-07-29 08:32:32	✓	
Event After Save	event	medium	This trigger is called after an Event has been saved in the database	11	×	✓	175	2022-07-29 08:37:23	✓	
Event Publish	event	low	This trigger is called just before a MISP Event starts the publishing process	1	✓	✓	180	2022-07-29 12:14:10	✓	
Object After Save	object	high	This trigger is called after an Object has been saved in the database	35	×	✓	161	2022-07-28 13:59:37	×	
Post After Save	post	low	This trigger is called after a Post has been saved in the database	36	×	×	176	2022-07-28 13:59:51	✓	
User After Save	user	low	This trigger is called after a user has been saved in the database	55	×	×	159	2022-07-28 14:00:03	✓	
User Before Save	user	low	This trigger is called just before a user is save in the database	42	✓	×	158	2022-07-28 14:00:32	✓	

Operations not allowed:

- Execution loop are not authorized
  - Current caveat: If an action re-run the workflow in any way

Operations not allowed:

- Multiple connections from the same output
  - Execution order not guaranted and confusing for users

Operations showing a warning:

- **Blocking** modules after a **concurrent tasks** module
- **Blocking** modules in a **non-blocking** workflow

Workflow blueprints

1. Blueprints allow to **re-use parts** of a workflow in another one
2. Blueprints can be saved, exported and **shared**

Blueprints origins:

1. From the "official" `misp-workflow-blueprints` repository
2. Created or imported by users

Workflow blueprints: Create

Select one or more modules to be saved as blueprint then click on the `save blueprint` button

Hash path filtering

- Some modules have the possibility to filter or check conditions using CakePHP's path expression.

```
1 $path_expression = '{n}[name=fred].id';
2 $users = [
3     {'id': 123, 'name': 'fred', 'surname': 'bloggs'},
4     {'id': 245, 'name': 'fred', 'surname': 'smith'},
5     {'id': 356, 'name': 'joe', 'surname': 'smith'},
6 ];
7 $ids = Hash::extract($users, $path_expression);
8 // => $ids will be [123, 245]
```

Module filtering

- Some action modules accept **filtering** conditions
- E.g. the `enrich-event` module will only perform the enrichment on Attributes having a `tlp:white` Tag

Data format in Workflows

- All triggers will inject data in a workflow
- In some cases, there is no format (e.g. User after-save)
- In others, the format is **compliant with the MISP Core format**
- In addition to the RFC, the passed data has **additional properties**
  - Attributes are **always encapsulated** in the Event or Object
  - Additional key **`_AttributeFlattened`**
  - Additional key **`_allTags`**
  - Additional key **`inherited`** for Tags

Logic module: Concurrent Task

- Special type of **logic** module allowing multiple connections
- Allows **breaking the execution** flow into a concurrent tasks to be executed later on by a background worker
- As a side effect, blocking modules **cannot cancel** ongoing operations

Debugging Workflows: Log Entries

- Workflow execution is logged in the application logs:
  - `/admin/logs/index`
- Or stored on disk in the following file:
  - `/app/tmp/logs/workflow-execution.log`
- Use the `webhook-listener.py` tool
  - `/app/tools/misp-workflows/webhook-listener.py`

Logs

« previous | next »

Emails | Authentication issues | MISP Update results | Setting changes | Warnings and errors

Id ↑	Email	Org	Created	Model	Model ID	Action	Title
49146	SYSTEM	SYSTEM	2022-08-01 07:34:40	Workflow	162	execute_workflow	Finished executing workflow for trigger `enrichment-before-query` (162). Outcome: success
49144	SYSTEM	SYSTEM	2022-08-01 07:34:39	Workflow	162	execute_workflow	Started executing workflow for trigger `enrichment-before-query` (162)

Debugging Workflows: Debug mode

- The

can be turned on for each workflows
- Each nodes will send data to the provided URL
  - Configure the setting: `Plugin.Workflow_debug_url`
- Result can be visualized in
  - **offline**: `tools/misp-workflows/webhook-listener.py`
  - **online**: `requestbin.com` or similar websites

Learning by examples

Workflow example 1

1. The `Event-Publish` trigger uses the MISP core format
2. The `IF::Tag` module checks if at least one of the Attribute has the `tlp:white` tag
3. If it does, the `Push-to-ZMQ` module will be executed

Workflow example 2

- If an event has the `tlp:red` tag or any of the attribute has it, the publish process will be cancelled

Extending the system

Creating a new module in PHP

app > Lib > WorkflowModules > action > Module_blueprint_action_module.php > ...

```php
<?php
include_once APP . 'Model/WorkflowModules/WorkflowBaseModule.php';

class Module_blueprint_action_module extends WorkflowBaseModule
{
    public $is_blocking = false;
    public $disabled = true;
    public $id = 'blueprint-action-module';
    public $name = 'Blueprint action module';
    public $description = 'Lorem ipsum dolor, sit amet consectetur adipisicing elit.';
    public $icon = 'shapes';
    public $inputs = 1;
    public $outputs = 1;
    public $params = [];

    public function exec(array $node, WorkflowRoamingData $roamingData, array &$errors = [])
        : bool
    {
        parent::exec($node, $roamingData, $errors);
        // If $this->is_blocking == true, returning `false` will stop the execution.
        $errors[] = __('Execution stopped');
        return false;
    }
}
```

- **app/Lib/**`WorkflowModules/action/[module_name].php`
- Module configuration are defined as public variables
- The exec function has to be implemented.
  - If it returns **true**, execution will proceed
  - If it returns **false**
    - And the module is blocking, the execution will stop and the operation will be blocked

Creating a new module in Python

```
home > sami > git > misp-modules > misp_modules > modules > action_mod > testaction.py > ...
1  import json
3
4  misperrors = {'error': 'Error'}
5
6  # config fields that your code expects from the site admin
7  moduleconfig = {
8      'foo': {
9          'type': 'string',
10         'description': 'blablabla',
11         'value': 'xyz'
12     },
13     'bar': {
14         'type': 'string',
15         'value': 'meh'
16     }
17 };
18
19 # blocking modules break the exection of the chain of actions (such as publishing)
20 blocking = False
21
22 # returns either "boolean" or "data"
23 # Boolean is used to simply signal that the execution has finished.
24 # For blocking modules the actual boolean value determines whether we break execution
25 returns = 'boolean'
26
27 moduleinfo = {'version': '0.1', 'author': 'Andras Iklody',
28                 'description': 'This module is merely a test, always returning true. Triggers on event publishing.',
29                 'module-type': ['action']}
30
31
32 def handler(q=False):
33     if q is False:
34         return False
35     result = json.loads(q)  # noqa
36     output = result  # Insert your magic here!
37     r = {"data": output}
38     return r
39
40
41 def introspection():
64
```

- Module configuration are defined in the `moduleinfo` and `moduleconfig` variables
- The `handler` function has to be implemented.
- Blocking logic is the same as other modules

MISP Concepts Cheat sheet

Glossary

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 transfering the ownership of an `Event` to another organisation while hidding 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 throught 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 transfered.

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

Proposals: Are a mechanism to propose modications 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.

Distribution

Controls who can see the data and how it should be synchronised.

Organisation only: Only members of your organisation

This community: Organisations on this MISP instance

Connected Communities: Organisations on this MISP instance and those on MISP instances synchronising with this one. Upon receiving data, the distribution will be downgraded to `This community` to avoid further propagation. ($n \leq 1$)

All Communities: Anyone having access. Data will be freely propagated in the network of connected MISP instances. ($n = \infty$)

Sharing Groups: Distribution list that exhaustively keeps track of which organisations can access the data and how it should be synchronised.

`Sharing Group` configuration	
Organisations	Org. α Org. ω Org. γ
Instances*	MISP 1 MISP 2 MISP 3

*Or enable roaming mode instead

Synchronisation

*The act of **sharing** where everyone can be a consumer and/or a producer.* A one way synchronisation link between two MISP instances. Organisation α created a *sync user* on MISP 2 and noted down the generated API Key. A synchronisation link can be created on MISP 1 using the API Key and the organisation of the *sync user*. At that point, MISP 1 can *pull* data from MISP 2 and *push* data to MISP 2.

MISP Data Model Cheat Sheet

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

Event

Encapsulations for contextually linked information.

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

Attribute

Basic building block to share information.

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

MISP Object

Advanced building block providing `Attribute` compositions via templates.

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

Object Reference

Relationships between individual building blocks.

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

Sightings

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

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

Event Report

Advanced building block containing formated text.

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

Proposals

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

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

Taxonomies

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

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

Galaxies

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

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

Galaxies Clusters

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

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

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

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

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

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

Representation of an incident in MISP

Failed spear-phishing attempt

UUID	28b1cd2e-46a7-4ee2-a364-c3d26451b089
Date	2021-12-09
Creator Org.	CIRCL.lu
Distribution	Connected Communities
Published	✓

Galaxies

- **Sector**
  - Telecoms
- **Country**
  - luxembourg
- **Attack Pattern**
  - Spearphishing Attachment - T1566.001
  - Phishing - T1566

Taxonomies

- workflow:state="draft"
- tlp:amber
- PAP:RED
- phishing:techniques="email-spoofing"
- phishing:distribution="spear-phishing"

> Intelligence Visualization Widgets

> Attributes

☐	2021-11-25	Payload delivery	ip-src	118.217.182.3
☐	2021-11-25	Payload delivery	url	https://evilprovider.com/this-is-not-malicious.exe

...

> Objects

2021-12-09 — **Object name:** file; **References:** 1; **Referenced by:** 1

☐	2021-12-09	Payload delivery	**malware-sample:** malware-sample	malicious.exe f1a3e62de12faecee82bf4599cc1fdcd
☐	2021-12-09	Payload delivery	**filename:** filename	malicious.exe
☐	2021-12-09	Payload delivery	**md5:** md5	f1a3e62de12faecee82bf4599cc1fdcd
☐	2021-12-09	Payload delivery	**sha1:** sha1	d836f2ee449b74913d1efc615eeb459b65e4f791
☐	2021-12-09	Payload delivery	**sha256:** sha256	d90401420908dbb4b3488a306467e8fffc57577ce9d5eee016578ff6a3ada12e
☐	2021-12-09	Other	**size-in-bytes:** size-in-bytes	751328

...

Event: Encapsulates contextually linked information.
Events also have basic information including ownership and access-control
Here: Contains all the information related to the spear-phishing incident.

Taxonomies: Simple label standardised on common set of vocabularies.
Here: Usage of labels to classify the current completeness of the Event, what recipient can do with the information and the category of the incident.

Galaxies & Galaxy-Clusters: Advanced label containing meta-data
Here: The sector affected by the incident as well as the country. The kill-chain of the attack can be described using the MITRE ATT&CK framework

Event Graph: Visualization of the relationships between entities contained in the Event.
Here: The whole story of the attack can be described with relationships defined between Attributes and Objects

Event Timeline: Visualization of the temporality of the data contained in the event.
Here: A timeline of the steps performed during the attack. The time data is taken directly from the Attributes and Objects belonging to the Event.

Event Report: Markdown-aware supporting text document to describe events or incidents
Here: The report describe the steps taken by the attacker and provide additional contextual information. It also contains references to Attributes and Object encoded in the Event

Attributes: Basic building block to represent information.
They can have context such as taxonomy and express if they are supportive data or meant for automation. An Event can have multiple Attributes
Here: Two Attributes representing payload delivery. One is an IP address, the other is an URL.

Objects: Advanced building block allowing Attribute composition via predefined templates.
As an Object is an instantiation of its template, it is composed of Attributes that make sense Together. They can also have relationship to other entity contained in the Event
Here: A file object composed of Attributes such as the filename, size and hashes. It also have a relationship

MISP User & Admin Cheat Sheet

- User -

API

Wildcard searches:

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

Or and Negation searches:

```
POST /attributes/restSearch
{"tags": ["tlp:white", "!tlp:green"]}
```

And and Negation searches:

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

Galaxy Cluster metadata searches:

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

Attach tags:

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

Timestamps:

`timestamp`: Time of the last modification on the data

- Usecase: 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

- Usecase: Get data that arrived in my system since t
- E.g.: New data from a feed

`event_timestamp`: Used in the Attribute scope

- Usecase: Get events modified in the last t

Usage:

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

Tips & Tricks

Get JSON Representation: Append `.json` to any URLs to get their content in JSON format. Example: `/events/view/42.json`

- Admin -

Reset Password

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

CLI: `MISP/app/Console/cake Password [email] [password]`

Reset Bruteforce login protection

CLI: `MISP/app/Console/cake Admin clearBruteforce [email]`

Upgrade to the latest version

All in 1-shot: `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: `MISP/app/Console/cake Admin restartWorkers`

Add: `MISP/app/Console/cake Admin startWorker [queue]`

Stop: `MISP/app/Console/cake Admin stopWorker [pid]`

Settings

Get: `MISP/app/Console/cake Admin getSetting [setting]`

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

Base URL: `MISP/app/Console/cake Baseurl [baseurl]`

Miscellaneous

Clean Caches: `MISP/app/Console/cake Admin cleanCaches`

Get IPs For User ID: `MISP/app/Console/cake Admin UserIP [user_id]`

Get User ID For User IP: `MISP/app/Console/cake Admin IPUser [ip]`

Documentation: `/events/automation`

Logs files location: `MISP/app/tmp/logs`

User

Check	Description	Length
☐	**Add events**	
☐	- via Standard UI	
☐	- Distribution levels and publication	
☐	- Different timestamps & publish_timestamp	
☐	**Add attributes**	
☐	- Freetext	
☐	- Standard UI	
☐	- Template	
☐	- ReST API	
☐	- via EventGraph	
☐	**Object**	
☐	- add Object	
☐	- add References	
☐	- show via EventGraph	
☐	***-lists**	
☐	- Warninglists: show warnings raised in steps above	
☐	- Noticelists: show warnings when adding data	
☐	- Import Regexp: avoid leaking private/personal data	
☐	**Correlations**	
☐	- show correlations that were added	
☐	- pivot to events via correlations	
☐	- show correlations graph	
☐	- feeds & servers correlation	
☐	**Tags and Galaxies**	
☐	- Tag from Taxonomy	
☐	- GalaxyCluster	
☐	- ATT&CK pattern & Galaxy matrix	
☐	- Tag Collection	
☐	**Sighting**	
☐	- via UI & API	
☐	**Delegation**	
☐	**Proposal**	
☐	**Delete (including soft versus hard delete)**	
☐	- Event blocklist when deleting	
☐	**Extending event** (how and when to use it)	
☐	**Exporting data**	
☐	- download from	
☐	- download from via modules	
☐	- .json routing	
☐	- RestSearch	
☐	**Searching for data**	
☐	- Attribute search	
☐	- Event index filter search	
☐	**Advanced features**	
☐	- Event graph, Event timeline, Event report	
☐	- Decaying of IoC	
☐	- Galaxy 2.0	
☐	**Enrichments**	
☐	- Hover & persistent	

Administrator (Community)

Check	Description	Length
☐	**Organisations**	10m
☐	- local and remote	
☐	- administration: Creation and merge	
☐	**User**	5m
☐	- administration and contact via standard UI	
☐	- Pasword/Auth key reset	
☐	- Disabling (never remove)	
☐	**Roles and permissions**	3m
☐	- Constraints & special sync-user	
☐	**Sharing group**	10m
☐	- administration via standard UI	
☐	**Block listing**	3m
☐	- Events & Organisations	
☐	**Synchronisation**	35m
☐	- MISP to MISP (sync_user, test & preview, flow control)	
☐	- Feeds to MISP (Options, overlap)	
☐	- Pub-Sub	
☐	**Collaboration settings**	
☐	- ‘proposal_block_attributes‘, ‘sanitise_attribute_on_delete‘, ‘Sightings_anonymise‘	
☐	**Templates**	
☐	- administration via standard UI	

Administrator (Instance)

Check	Description	Length
☐	**Advanced Auth keys**	3m
☐	- Migration from old system	
☐	- Usage	
☐	**Server settings**	5m
☐	**Maintenance**	15m
☐	- Updating & release process	
☐	- Submodules and populate DB	
☐	- Diagnostic	
☐	**Jobs and Workers**	10m
☐	- Administration via standard UI	
☐	- Scheduled Tasks and CRON jobs	
☐	**User settings & User management**	5m
☐	- User settings	
☐	- User monitoring, self-management, auto-registration	
☐	**Logging & auditing**	10m
☐	- Logs (and purge: event history)	
☐	- Paranoid, IP & Auth log, Sync audit	
☐	**Troubleshooting**	5m
☐	- Clean cache & DB Schema diagnostic	
☐	- Stuck workers	
☐	- Update in progress	
☐	- Apache logs & workers logs	

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`
- Alexandre Dulaunoy `https://github.com/adulau`
- Andras Iklody `https://github.com/iglocska`
- Sami Mokaddem `https://github.com/mokaddem`
- Sascha Rommelfangen `https://github.com/rommelfs`
- Christian Studer `https://github.com/chrisr3d`
- Raphaël Vinot `https://github.com/rafiot`
- Gerard Wagener `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*.

Co-financed by the Connecting Europe Facility of the European Union

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

CIRCL
Computer Incident
Response Center
Luxembourg