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.

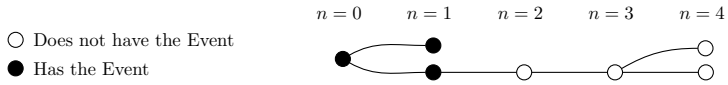
Distribution

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

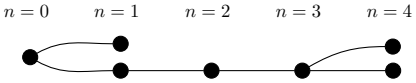
Organisation only: Only members of your organisation

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. ($n \leq 1$)



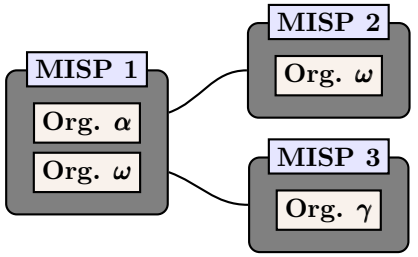
All Communities: Anyone having access. Data will be freely propagated in the network of connected MISP instances. ($n = \infty$)



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

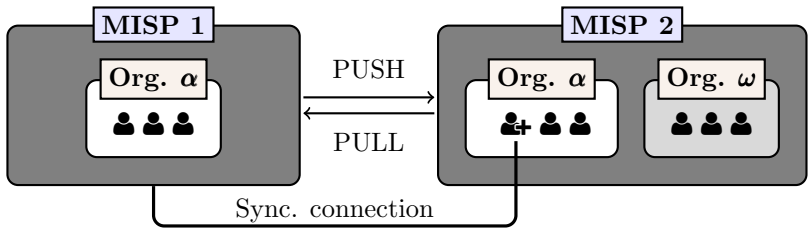
`Sharing Group` configuration	
Organisations	Org. α Org. ω Org. γ
Instances*	MISP 1 MISP 2 MISP 3

*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.

𝒯 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`.

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`