DEEP-DIVE INTO PYMISP

MISP - THREAT SHARING

CIRCL / TEAM MISP PROJECT

HTTP://WWW.MISP-PROJECT.ORG/ TWITTER: @MISPPROJECT

13TH ENISA-EC3 WORKSHOP



Deep-dive into PyMISP





MISP is a large project
 Your production environment is even more complex
 yrd party services are even worse
 Querying MISP via CURL is doable, but grit's painful fast
 Talking to MySIC ultercity can be dangerous
 POST a JSON blob, receive a JSON blob. You can do it

■ MISP is a large project

- Your production environment is even more complex
- **3**rd 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)

 $\mathop{\rm Deep-dive}$ into PyMISP $\mathop{\rm \resize}$

9

2024

└─Big picture

Core goal: providing stable access to APIs, respect access
 content
 content
 content
 simpling landling & automation of indicators in 3rd party
 tool
 Providing pro-code examples for commonly used
 operations
 Helping information with existing infrastructure

- 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

Deep-dive into PyMISP

- there are 4 main cases here: Metadata of the events that have been modified
- search_index => timestamp (th, td, 7d, ...), returns list of all the modified constraints
- the modified events
 Full events (metadata + attributes)
- search → timestamp (th, td, 7d, ...
 Modified attributes
- Modified attributes
 search -> controller attributes and timestame (th. td. 7d. ...
- Other use case: get last published events by using the last parameter in the search method.

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 \Rightarrow timestamp (1h, 1d, 7d, ...)
- Modified attributes
 - **search** \Rightarrow controller = attributes and timestamp (1h, 1d, 7d, ...)
- Other use case: get last **published** events by using the last parameter in the **search** method.

10-02

2024

└─Common queries: Search things

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

There are 3 main cases here:

- 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).

10-02

2024

Common queries: create things

There are 3 main cases here: ■ Add Event, eddi (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

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

Deep-dive into PyMISP

3

2024-10

Administrative tasks

Assyming you have the right to do it on the instance Managing users Managing organisations Managing sync servers

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

- Managing users
- Managing organisations
- Managing sync servers

OTHER CAPABILITIES

Deep-dive into PyMISP

8

9

2024

—Other Capabilities

OTHER CAPABILITIES

E Upload/download samples **Proposite** add, edit, accept, discard **B Sightings**. Get, set, update **B Export statistics Wanage feed Manage feed Get MSP server version**, recommended PyMSP version **B** And more, look at the api file

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

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)

Deep-dive into PyMISP

3

2024-10

MISPEvent - Usecase

MISPEVENT - USECAS

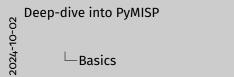
from pymisp import NISPEvent, EncodeUpdate

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

Load on existing (SON dump (optional) event.load_file('Path/to/event.json') event.info = 'Ny cool event' # Duh.

Add on attribute of type ip-dst event.add_attribute("ip-dst", "8.8.8.8")

were at attraute at defeted (From 2.4.60) vent.delete_attribute('<Attribute UUID>')



Python 3.5⁺ is recommended
 PytHon 5.5⁺ is always inline with current version (pipg install pymics)
 Dev version: pipg install givintary (given such kay from: thtps://filespit.cirel.lu/events/automation thtps://filespit.cirel.lu/events/automation your stature admin.
 Source available here: dt come

https://github.com/MISP/PyMISP.git

- 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



XAMPLES

PyMISP needs to be installed (duh)
 Usage:

Create examples/keys.py with the following content

misp_uri = "https://uri-to-your misp_key = "<API_KEY>" misp_verifycert = True ■ Proxy support:

> pression = { "http:///isp.a.a.u.tapp", "http://isp.a.a.u.tapp", } memorylation and address and sectored consistential

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:

EXAMPLES

Deep-dive into PyMISP

└─ Examples

2024-

- Lots of ideas on how to use the API
 You may also want to look at the tests directory
- You may also want to look at the tests directory
 All the examples use angoarse. Help usage is available:
- script.py-h > add_file_object.py: Attach a file (PE/ELF/Mach-O) object to
 - add_hte_object.py: Attach a file (PE/ELF/Mach-O) object 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 timefrance)
- add_named_attribute.py: Add attribute to an event
 sighting.py: Update sightings on an attribute
- signing py: optime signings on an accretione
 stats.py: Returns the stats of a MISP instance
- ► {add,edit,create}_user.py : Add, Edit, Create a user on MISI

- 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

Deep-dive into PyMISP

3

2024-10

Basic example from synthe legent synthy requests - applic verifycent-true, debugsfaltes, president if requests - applic functions if requests ("reset") if a samething went wengs else:

Basic example

<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.

Deep-dive into PyMISP

2024-

Concept behind AbstractMISP

CONCEPT BEHIND ABSTRACTMISE

- 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 ISON
- 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, MISPSIGHTING...

MISPATTRIBUTE,

MISPOBJECT,

Deep-dive into PyMISP

8

ģ

2024-

MISPEvent, MISPAttribute, MISPObject, MISPSighting... MISPEVENT, MISPATTRIBUTE, MISPOBJECT, MISPSIGHTING...

- Pythonic representation of MISP elements
 Easy manipulation
- Load an existing event
 Update te metadata, add attributes, objects, tags, mari
- attribute as deleted, ... Set relations between objects
- Load and add attachments or malware samples as pr files

Dump to JSON

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()

Deep-dive into PyMISP

3

9

2024

└─MISPEvent - Main entrypoints

MISPEVENT - MAIN ENTRYPOINTS

Intel (Morent, gam)
 Intel(gam, gam)
 Intel(gam, gam)
 Intel(gam, gam)
 Intel(gam, gam)
 Intel(gam, gam)
 Intel(gam)
 Intel(g

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, ...)

Deep-dive into PyMISP

2024-

└─MISPObject - Main entrypoints

MISPOBJECT - MAIN ENTRYPOINTS

 add_arthivate/objecc_relation," malue)
 add_arthivate/objecc_relation_", malue)
 add_arthivate/objecc_relationship.type, comment-toon, "fearage)
 Ina_arthivate_prelation(file_arthivate)
 attributed_arthivate.typ_relation(file_arthivate)
 attributed_arthivation(file_arthivation(file_arthivate)
 attributed_arthivation(file_arthiv

8

9

2024

MISPAttribute - Main entrypoints

a add_tagftag=Wone, **Wwargs) = deleta() = taggt] = taggt] = cdited, all other paramaters of the MISPObject element (value, comment, __) = to_joon()

- 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

Deep-dive into PyMISP

3

2024-10

└─PyMISP - Tools

PYMISP - TOOLS

Libraries requiring specific 3rd party dependencies
 Callable via PyMISP for specific usecases
 Currently implemented:
 PopenIOC to MISP Event
 MISP to Neos.]

- 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

Deep-dive into PyMISP

-10-02

2024

└─PyMISP - Default objects generators

File - PE/ELF/MachO - Sections
 VirusTotal
 Generic object generator

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
```

Deep-dive into PyMISP

3

9

2024-

PyMISP - Logging / Debugging

PyMISP - Logging / Debugging

 debug-True passed to the constructor enable debug to stoout
 Configurable using the standard logging module
 Show everything send to the server and received by the client dipart primp impart primp impart ingrime

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

ging.basicConfig(level-logging.DEBUG, # Enable debug to ; filename="debug.log", filemade="w", format=pymisp.FORMAT)



https://github.com/HISP/PyHISP https://github.com/HISP/ https://github.com/HISP/ https://github.com/HISP/ www.ecome.new.net.notionalities.and.pull requests.

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