

PyMISP - (ab)using MISP API with PyMISP

MISP - Malware Information Sharing Platform & Threat Sharing



CIRCL

Computer Incident
Response Center
Luxembourg

MISP Project @MISPPProject -
TLP:WHITE

MISP Training
@SWITCH-CERT - 20161206

PyMISP - Basics

- Installation (v2.4.53 - Python 3 highly recommended):
 - pip3 install pymisp
- Get your auth key from:
 - <https://misppriv.circl.lu/events/automation>
- Fetch the repository to get the examples:
 - git clone <https://github.com/MISP/PyMISP.git>

PyMISP - Examples

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

```
misp_url = "https://misppriv.circl.lu"  
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, 'json', proxies=proxies)
```

PyMISP - Examples

- All the examples have help if you do **script.py -h**
- **searchall.py**: Search in the whole database for a value
- **last.py**: Returns all the most recent events (on a timeframe)
- **get.py**: Return a specific event
- **tags.py**: Returns all the tags activated on the platform
- **get_network_activity.py**: Returns network indicators
- **create_events.py**: Create an event
- **up.py**: Update an event
- **add_named_attribute.py**: Add attribute in MISP instance easily
- **upload.py**: Upload a malware sample

PyMISP - Examples

- **copy_list.py**: Copy files from one MISP instance to an other
- **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
- **test_sign.py**: Sign and verify a MISP Event
- **make_neo4j.py**: Search MISP Events matching a value and push them into neo4j

PyMISP - Usage

- Basic example

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

PyMISP - Capabilities

- Events: get, add, update, publish, delete, add/remove tag, ...
- Add file attributes: hashes, registry key, patterns, pipe, mutex
- **Update sightings**
- Add network attributes: IP dest/src, hostname, domain, url, UA, ...
- Add Email attributes: source, destination, subject, attachment, ...
- Upload/download samples
- Proposals: add, edit, accept, discard
- **Full text search** and search by attributes
- Get **STIX** event
- Export **statistics**
- And more, look at the api file

MISPEvent

- **Pythonic** representation of a MISP event
- **Easy manipulation** and **validation**
- **Signing** and **verifying** a MISP Event (GPG)
- **Dump** to JSON

MISPEvent - Usecase

```
from pymisp import MISPEvent, EncodeUpdate

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

# Load an existing JSON dump (optional)
event.load('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')

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

PyMISP - Tools

- Libraries requiring specific 3rd party dependencies
- Callable via PyMISP for specific usecases
- Currently implemented:
 - MISP Event to and from **STIX Package**
 - **OpenIOC** to MISP Event
 - MISP to **Neo4J**

PyMISP - Situational Awareness (WiP)

- High level view of the type of attributes
- Searchable over a timeframe & tag

Attributes Distribution

| Payload type | |
|--------------|----|
| comment | 14 |

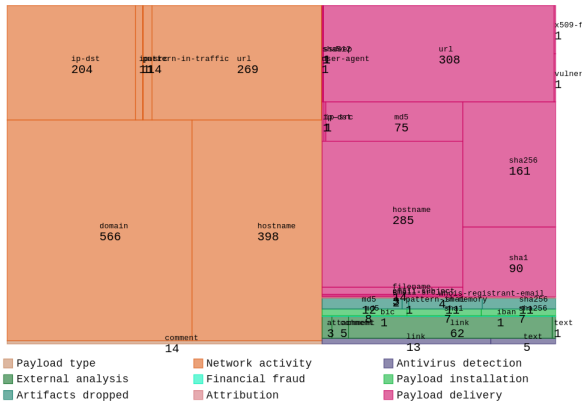
| Network activity | |
|--------------------|-----|
| domain | 566 |
| hostname | 398 |
| ip-dst | 204 |
| ip-src | 11 |
| pattern-in-traffic | 1 |
| url | 14 |
| url | 269 |
| user-agent | 1 |

| Antivirus detection | |
|---------------------|----|
| link | 13 |
| text | 5 |

| External analysis | |
|-------------------|----|
| attachment | 3 |
| comment | 5 |
| link | 62 |
| text | 1 |

| Financial fraud | |
|-----------------|---|
| bic | 1 |
| iban | 1 |

| Payload installation | |
|----------------------|---|
| md5 | 8 |



PyMISP - Feed generator

- Used to generate the **CIRCL OSINT feed**
- Export events as json based on tags, organisation, events, ...
- Automatically update the dumps and the metadata file
- Comparable to a lightweight **TAXII interface**

PyMISP - Feed generator - Config file

```
url = ''  
key = ''  
ssl = True  
outputdir = 'output'  
  
# filters = {'tag': 'tlp:white|feed-export|!privint', 'org': 'CIRCL'}  
filters = {}  
  
valid_attribute_distribution_levels = ['0', '1', '2', '3', '4', '5']
```

Q&A



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