PyMISP - (ab)using MISP API with PyMISP

MISP - Malware Information Sharing Platform & Threat Sharing

CIRCL
Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy
Andras Iklody
Raphaël Vinot
Gerard Wagener

TLP:WHITE

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

MISP Training @ Luxembourg
20171121

PyMISP - Basics

- Installation (v2.4.82 - Python 3.5+ highly recommended):
  - sudo 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, proxies=proxies)
```

PyMISP - Examples

- All the examples have help if you do **script.py -h**
- **add_file_object.py**: Attach a file (PE/ELF/Mach-O) object to an event
- **generate_file_objects.py**: Generate a json dump ready to push to MISP
- **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
- **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

- **tags.py**: Returns all the tags activted on the platform
- **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, 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
- ... generate objects.
- **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

PyMISP - Core methods

- Get a MISP event as JSON: **get**
- Create a new event: **new_event**
- Add an attribute to existing event: **add_named_attribute**
- Upload a sample: **upload_sample**
- Download a sample: **download_samples**
- Get all events matching a value: **search_all**

MISPEvent

- **Pythonic** representation of a MISP event
- **Easy manipulation** and **validation**
  - Loading an existing event
  - Updating (including mark an attribute as deleted)
  - Load and add attachments to send to the MISP instance
- **Signing** and **verifying** a MISP Event (GPG)
- **Dump** to JSON

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)
```

PyMISP - Tools

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

PyMISP - Objects

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

PyMISP - AbstractMISP

- Master class making all the MISP objects Mutable Mappings (dict)
- Makes all the MISP objects easily exportable to json
- All the public properties of the class are included in the json
- Look at the examples.

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

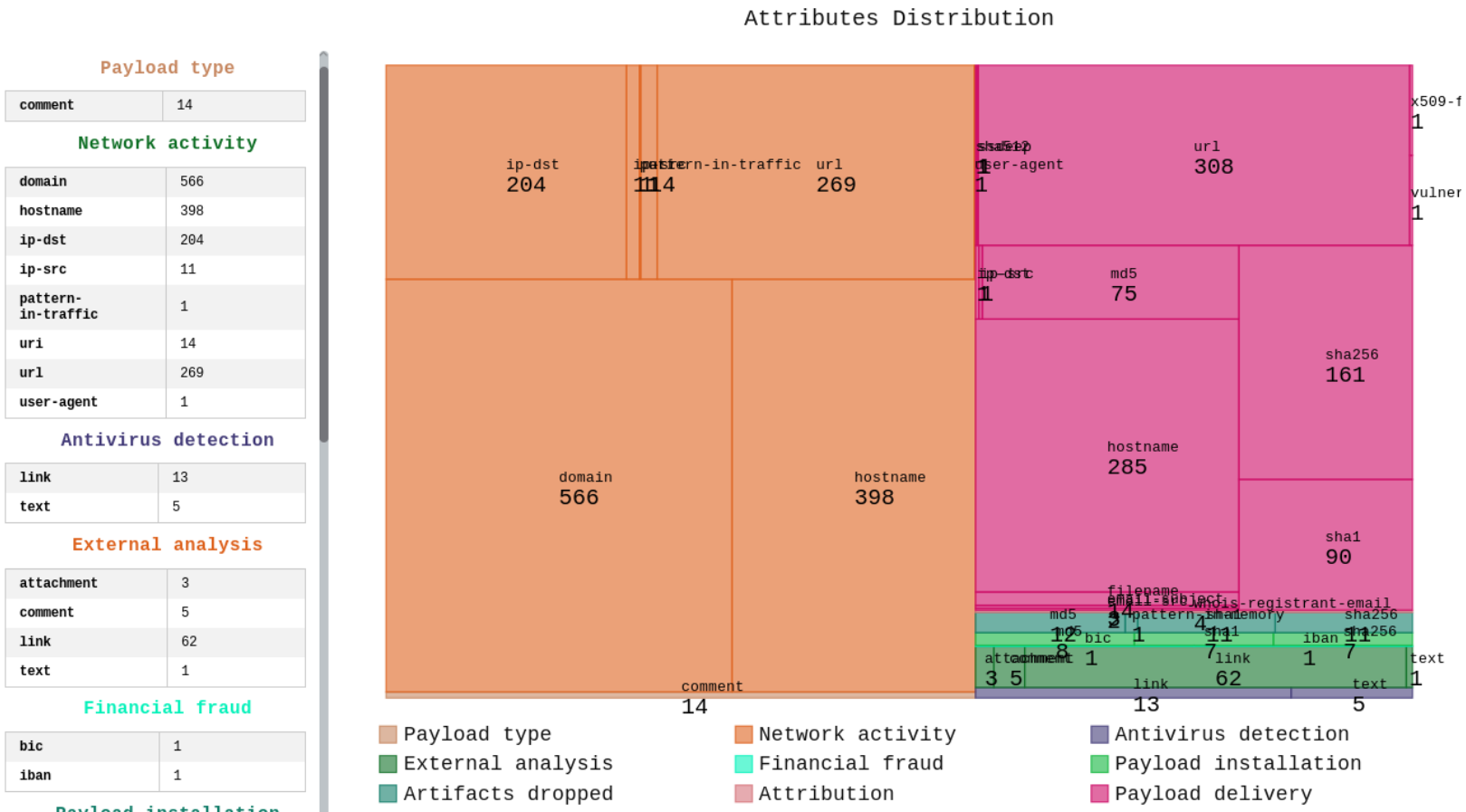
```
import pymisp
import logging

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)
```

PyMISP - Situational Awareness (WiP)

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



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 lighweight **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.