Extending MISP with Python modules

MISP - Malware Information Sharing Platform & Threat Sharing

CIRCL
Computer Incident
Response Center
Luxembourg

MISP Project @MISPProject

TLP:WHITE

MISP Training - @SWITCH - 20161206

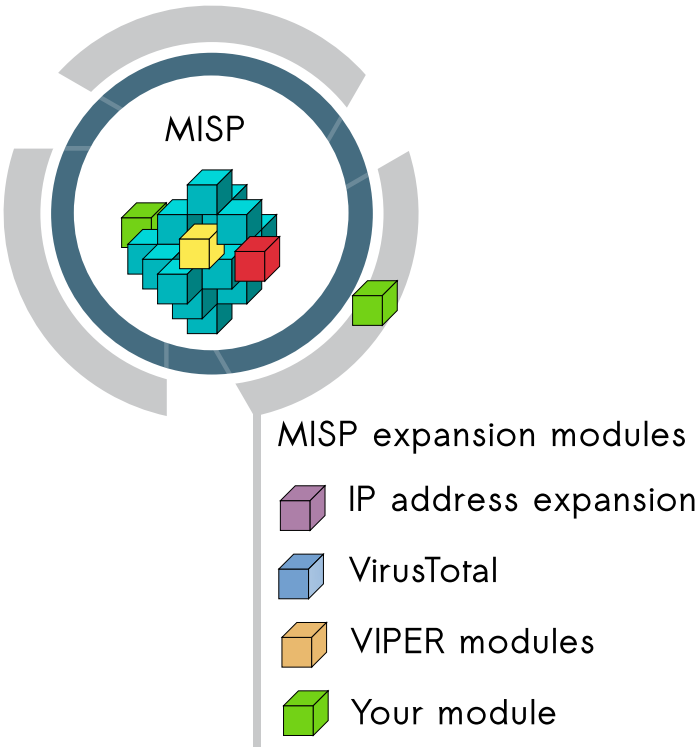
Why we want to go more modular...

- 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 –upgrade -r REQUIREMENTS
  - sudo pip3 install –upgrade .
  - 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


```
{
"type": "expansion",
"name": "dns",
"meta": {
  "module-type": [
    "expansion",
    "hover"
  ],
  "description": "Simple DNS expansion
     service to resolve IP address from
     MISP attributes",
  "author": "Alexandre Dulaunoy",
  "version": "0.1"
},
"mispattributes": {
  "output": [
    "ip-src",
    "ip-dst"
  ],
  "input": [
    "hostname",
    "domain"
  ]
}
```

Querying a module

- curl -s http://127.0.0.1:6666/query -H "Content-Type: application/json" –data @body.json -X POST

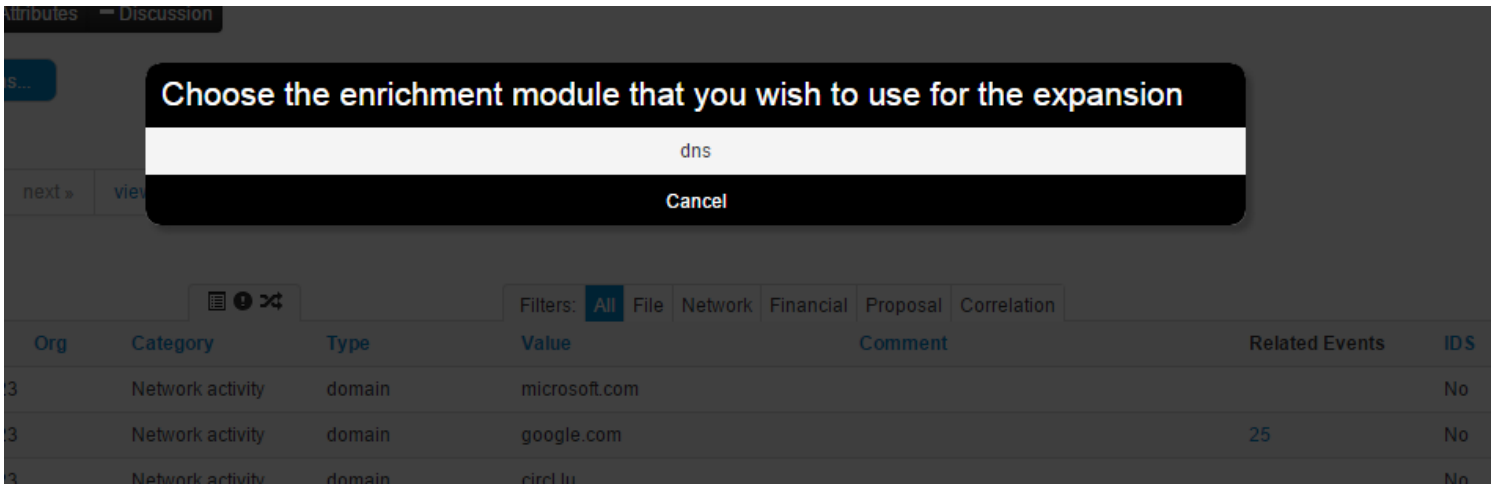
body.json

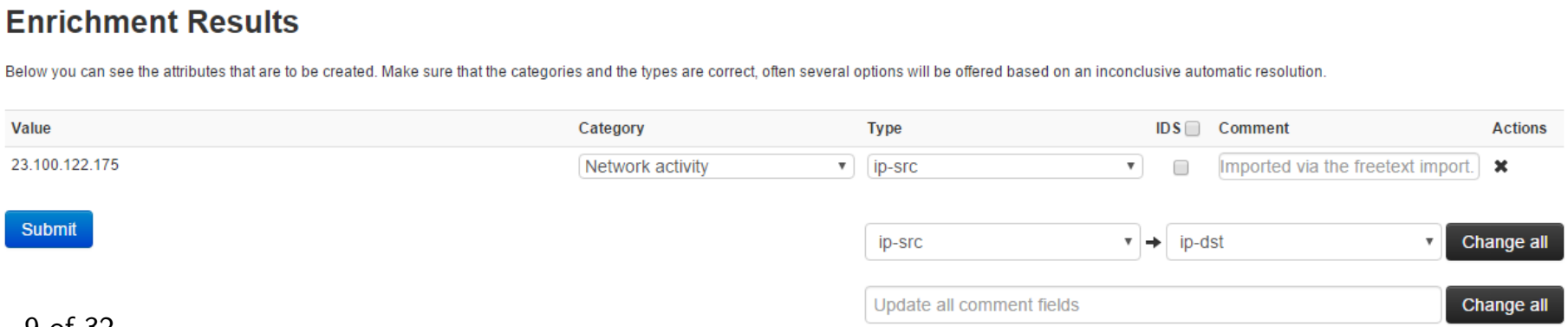
```
{"module": "dns", "hostname": "www.circl.lu"}
```

- and the response of the dns module:

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

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





MISP modules - configuration in the UI

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

Creating your Expansion module (Skeleton)

```
import json
import dns.resolver

misperrors = {'error' : 'Error'}
mispattributes = {'input': [], 'output': []}
moduleinfo = {'version': '', 'author': '',
              'description': '', 'module-type': []}

def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    r = {'results': [{'types': [], 'values':[]}]}
    return r
def introspection():
    return mispattributes
def version():
    return moduleinfo
```

Creating your Expansion module (metadata 1)

```
misperrors = {'error' : 'Error'}
mispattributes = {'input': ['hostname', 'domain'], 'output': ['ip-src', 'ip-dst']}
moduleinfo = {'version': '', 'author': '',
              'description': '', 'module-type': []}
```

Creating your Expansion module (metadata 2)

```
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']}
```

Creating your Expansion module (handler 1)

```
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    # MAGIC
    # MORE MAGIC
    r = {'results': [
        {'types': output_types, 'values':values},
        {'types': output_types2, 'values':values2}
    ]}
    return r
```

Creating your Expansion module (handler 2)

```
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
```

Creating your module - finished DNS module

```
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
- 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
- System is still new but some modules already exist
  - OCR module
  - Simple STIX import module
- Many ideas for future modules (OpenIOC import, connector to sandboxes, STIX 2.0, etc)

Creating your Import module (Skeleton)

```
import json

misperrors = {'error' : 'Error'}
userConfig = {
                'number1': {
                    'type': 'Integer',
                    'regex': '/^[0-4]$/i',
                    'errorMessage': 'Expected a number in range [0-4]',
                    'message': 'Column number used for value'
                }
            };
inputSource = ['file', 'paste']
moduleinfo = {'version': '', 'author': '',
              'description': '', 'module-type': ['import']}
moduleconfig=[]

def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    request["data"] = base64.b64decode(request["data"])
    r = {'results': [{'categories': [], 'types': [], 'values':[]}]}
    return r

def introspection():
    return {'userConfig': userConfig, 'inputSource': inputSource, 'moduleConfig': moduleConfig}

def version():
    return moduleinfo
```

Creating your import module (userConfig and inputSource)

```
userConfig = {
    'number1': {
        'type': 'Integer',
        'regex': '/^[0-4]$/i',
        'errorMessage': 'Expected a number in range [0-4]',
        'message': 'Column number used for value'
    }
};
inputSource = ['file', 'paste']
```

Creating your import module (Handler)

```
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    request["data"] = base64.b64decode(request["data"])
    r = {'results': [{'categories': [], 'types': [], 'values':[]}]}
    return r
```

Creating your import module (Introspection)

```
def introspection():
    modulesetup = {}
    try:
        userConfig
        modulesetup['userConfig'] = userConfig
    except NameError:
        pass
    try:
        moduleConfig
        modulesetup['moduleConfig'] = moduleConfig
    except NameError:
        pass
    try:
        inputSource
        modulesetup['inputSource'] = inputSource
    except NameError:
        pass
    return modulesetup
```

Export modules

- Input is currently only a single event
- Dynamic settings
- Later on to be expanded to event collections / attribute collections
- Output is a file in the export format served back to the user
- Export modules was recently introduced but a CEF export module already available
- Lots of ideas for upcoming modules

Creating your Export module (Skeleton)

```
import json
inputSource = ['event']
outputFileExtension = 'txt'
responseType = 'application/txt'
moduleinfo = {'version': '0.1', 'author': 'Andras Iklody',
              'description': 'Skeleton export module',
              'module-type': ['export']}

def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    # insert your magic here!
    output = my_magic(request["data"])
    r = {"data":base64.b64encode(output.encode('utf-8')).decode('utf-8')}
    return r

def introspection():
    return {'userConfig': userConfig, 'inputSource': inputSource, 'moduleConfig': moduleConfig, 'outputFileExtension': outputFileExtension}

def version():
    return moduleinfo
```

Creating your export module (settings)

```
inputSource = ['event']
outputFileExtension = 'txt'
responseType = 'application/txt'
```

Creating your export module (handler)

```
def handler(q=False):
    if q is False:
        return False
    request = json.loads(q)
    # insert your magic here!
    output = my_magic(request["data"])
    r = {"data":base64.b64encode(output.encode('utf-8')).decode('utf-8')}
    return r
```

Creating your export module (introspection)

```
def introspection():
    modulesetup = {}
    try:
        responseType
        modulesetup['responseType'] = responseType
    except NameError:
        pass
    try:
        userConfig
        modulesetup['userConfig'] = userConfig
    except NameError:
        pass
    try:
        moduleConfig
        modulesetup['moduleConfig'] = moduleConfig
    except NameError:
        pass
    try:
        outputFileExtension
        modulesetup['outputFileExtension'] = outputFileExtension
    except NameError:
        pass
    try:
        inputSource
```

Upcoming additions to the module system - General

- Expose the modules to the APIs
- Move the modules to background processes with a messaging system
- Difficulty is dealing with uncertain results on import (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.