py-attack

Release 0.0.1

Thijs van Ede

Nov 28, 2022

CONTENTS:

The *py_attack* library provides a wrapper for interacting with the MITRE ATT&CK framework.

CHAPTER

ONE

INSTALLATION

The most straightforward way of installing *py-attack* is via pip.

```
pip3 install py-attack
```

1.1 From source

If you wish to stay up to date with the latest development version, you can instead download the source code. In this case, make sure that you have all the required *dependencies* installed. You can clone the code from GitHub:

```
git clone git@github.com:Thijsvanede/py-attack.git
```

Next, you can install the latest version using pip:

```
pip install -e <path/to/py-attack/directory/containing/setup.py>
```

1.1.1 Dependencies

py-attack requires the following python packages to be installed:

- argformat: https://pypi.org/project/argformat/
- matplotlib: https://pypi.org/project/matplotlib/
- networkx: https://pypi.org/project/networkx/
- requests: https://pypi.org/project/requests/

All dependencies should be automatically downloaded if you install *py-attack* via pip. However, should you want to install these libraries manually, you can install the dependencies using the requirements.txt file

```
pip install -r requirements.txt
```

Or you can install these libraries yourself

```
pip install -U argformat matplotlib networkx requests
```

CHAPTER

TWO

USAGE

We provide example scripts for using the `py-attack` module in your own code in our `/examples/` directory on GitHub.

CHAPTER

THREE

REFERENCE

This is the reference documentation for the classes and methods objects provided by the *py-attack* module. We provide the following submodules within *py-attack*:

3.1 ATTACK

The ATTACK object provides a simple interface for loading and interacting with the ATT&CK framework.

3.1.1 Initialization

We provide two methods of loading the ATTACK object, either from a local repository through `load()`, or by downloading the ATTACK object from a remote repository using `download()`. The recommended way of initializing an ATTACK object is through `load()` as this assures that your project works with a consistent version of the MITRE ATT&CK framework and avoids repeated downloading of the CTI sources.

Example

```
# Import ATT&CK
from py_attack import ATTACK

# Load from local repository - recommended
attack = ATTACK.load(
    path    = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
    domains = ['enterprise', 'ics', 'mobile', 'pre'],
)

# Download from online source
attack = ATTACK.download(
    url     = "https://raw.githubusercontent.com/mitre/cti/master/{domain}-attack/
↪{domain}-attack.json",
    domains = ['enterprise', 'ics', 'mobile', 'pre'],
)
```

3.1.2 Domains

You can get, set and delete MITRE *ATTACKDomain* s according to its DomainTypes (see *types*).

Example

```
# Import ATT&CK
from py_attack import ATTACK, ATTACKDomain

# Load from local repository - recommended
attack = ATTACK.load(
    path    = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
    domains = ['enterprise', 'ics', 'mobile', 'pre'],
)

# Get enterprise domain
enterprise = attack['enterprise']
# Delete enterprise domain
del attack['enterprise']
# Set enterprise domain
attack['enterprise'] = ATTACKDomain.load(
    path   = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
    domain = 'enterprise',
)

# Iterate over all domains
for domain in attack:
    ...

# Show number of domains
print(len(attack))
```

3.1.3 Iterators

Similar to *ATTACKDomain*, `ATTACK` provides iterators to iterate over all `concepts` for each *ATTACKDomain* of the `ATTACK` object. The `ATTACK` object supports iterators for the following `concepts`: `matrices`, `tactics`, `techniques`, `sub_techniques`, `groups`, `software`, `procedures`, `relationships` and `mitigations`,. All of these are easily accessible via the following iterator properties:

Example

```
# Import ATT&CK
from py_attack import ATTACK

# Load from local repository - recommended
attack = ATTACK.load(
    path    = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
    domains = ['enterprise', 'ics', 'mobile', 'pre'],
)
```

(continues on next page)

(continued from previous page)

```
# Iterate over different concepts
for concept in attack.concepts:
    ...
for matrices in attack.matrices:
    ...
for tactics in attack.tactics:
    ...
for techniques in attack.techniques:
    ...
for sub_techniques in attack.sub_techniques:
    ...
for groups in attack.groups:
    ...
for software in attack.software:
    ...
for procedures in attack.procedures:
    ...
for relationships in attack.relationships:
    ...
for mitigations in attack.mitigations:
    ...
```

3.1.4 Graph

All concepts within the `ATTACK` have defined relations between them. E.g., each `domain` specifies `groups` that use `techniques` to achieve `tactics` using specific `software`. These concepts and relations can therefore be modeled in a graph provided by the `graph` property.

Because all these concepts are related, we provide a method to find concepts that are (in)directly related to a given concept:

Example

```
# Import ATT&CK
from py_attack import ATTACK

# Load from local repository - recommended
attack = ATTACK.load(
    path    = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
    domains = ['enterprise', 'ics', 'mobile', 'pre'],
)

# Get domain graph
graph = attack.graph

# Get concepts related to given ID T1087
related = attack.related_concepts('T1087')
```

3.2 ATTACKDomain

The `ATTACKDomain` object provides a simple interface for loading and interacting with a single domain within the ATT&CK framework.

3.2.1 Initialization

We provide two methods of loading the `ATTACKDomain` object, either from a local repository through `load()`, or by downloading the `ATTACKDomain` object from a remote repository using `download()`. The recommended way of initializing an `ATTACKDomain` object is through `load()` as this assures that your project works with a consistent version of the MITRE ATT&CK framework and avoids repeated downloading of the CTI sources.

Example

```
# Import ATT&CK
from py_attack import ATTACKDomain

# Load from local repository - recommended
domain = ATTACKDomain.load(
    path   = "path/to/local/cti/enterprise-attack/enterprise-attack.json",
    domain = 'enterprise',
)

# Download from online source
domain = ATTACKDomain.download(
    url    = "https://raw.githubusercontent.com/mitre/cti/master/enterprise-attack/
↪enterprise-attack.json",
    domain = 'enterprise',
)
```

3.2.2 Getters

You can retrieve a specific MITRE ATT&CK concept according to its identifier (see *format*) or `UUID`.

Example

```
# Import ATT&CK
from py_attack import ATTACKDomain

# Load from local repository - recommended
domain = ATTACKDomain.load(
    path   = "path/to/local/cti/enterprise-attack/enterprise-attack.json",
    domain = 'enterprise',
)

# Get technique using ID T1087
technique = domain['T1087']
technique = domain.get('T1087')
```

3.2.3 Iterators

Rather than retrieving a concept via one of the *DomainGetter* methods, you can also iterate over various `concepts`. A domain within the MITRE ATT&CK framework consists of the following `concepts`: `matrices`, `tactics`, `techniques`, `sub_techniques`, `groups`, `software`, `procedures`, `relationships` and `mitigations`,. All of these are easily accessible via the following iterator properties:

Example

```
# Import ATT&CK
from py_attack import ATTACKDomain

# Load from local repository - recommended
domain = ATTACKDomain.load(
    path   = "path/to/local/cti/enterprise-attack/enterprise-attack.json",
    domain = 'enterprise',
)

# Iterate over different concepts
for concept in domain.concepts:
    ...
for matrices in domain.matrices:
    ...
for tactics in domain.tactics:
    ...
for techniques in domain.techniques:
    ...
for sub_techniques in domain.sub_techniques:
    ...
for groups in domain.groups:
    ...
for software in domain.software:
    ...
for procedures in domain.procedures:
    ...
for relationships in domain.relationships:
    ...
for mitigations in domain.mitigations:
    ...
```

3.2.4 Graph

All concepts within the `ATTACKDomain` have defined relations between them. E.g., `groups` use `techniques` to achieve `tactics` using specific `software`. These concepts and relations can therefore be modeled in a graph provided by the `graph` property.

Because all these concepts are related, we provide a method to find concepts that are (in)directly related to a given concept:

Example

```
# Import ATT&CK
from py_attack import ATTACKDomain

# Load from local repository - recommended
domain = ATTACKDomain.load(
    path   = "path/to/local/cti/enterprise-attack/enterprise-attack.json",
    domain = 'enterprise',
)

# Get domain graph
graph = domain.graph

# Get concepts related to given ID T1087
related = domain.related_concepts('T1087')
```

3.3 filter

The `py_attack.filter` module provides a `Filter` object for filtering the CTI repository and a `query()` method for applying those filters.

3.3.1 Query

You can query a CTI repository (`Iterable[dict]` with a `List[Filters]`) using the following method:

3.3.2 Example

```
# Import ATT&CK
from py_attack import ATTACK
from py_attack.filter import Filter, query

# Load from local repository
attack = ATTACK.load(
    path    = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
    domains = ['enterprise', 'ics', 'mobile', 'pre'],
)

# Query for all MITRE ATT&CK techniques in store of enterprise objects
techniques = query(
    iterable = attack['enterprise'].store, # Underlying CTI datastructure of 'enterprise'␣
↪domain
    filters  = [
        Filter('type', '=', 'attack-pattern'), # Filter all CTI entries where type ==␣
↪attack-pattern, i.e., MITRE ATT&CK techniques
    ]
)
```

3.4 types

The *py_attack.types* module provides types used for type hinting.

3.5 format

Additionally, we specify the formatting of MITRE ATT&CK concepts:

Table 1: MITRE ATT&CK concept formats

Concept	Format
Tactics	`TAxxxx`
Techniques	`Txxxx`
Sub-techniques	`Txxxx.xxx`
Mitigation	`Mxxxx`
Group	`Gxxxx`
Software	`Sxxxx`
UUID	`<STIX-TYPE>--<UUID>`

3.6 utils

The `py_attack.utils` module provides functions to get `ID` and `UUID` values from ATT&CK concepts.

3.6.1 Example

```
 # Import ATT&CK
 from py_attack import ATTACK

 # Load from local repository
 attack = ATTACK.load(
     path    = "path/to/local/cti/{domain}-attack/{domain}-attack.json",
     domains = ['enterprise', 'ics', 'mobile', 'pre'],
 )

 # Get concept by identifier
 concept = attack.get(identifier='T1087')

# Get ID from concept
id = get_id(concept)
# Get UUID from concept
uuid = get_uuid(concept)
```

CHAPTER

FOUR

CONTRIBUTORS

This page lists all the contributors to this project. If you want to be involved in maintaining code or adding new features, please email t(dot)s(dot)vanede(at)utwente(dot)nl.

4.1 Code

- Thijs van Ede

4.2 Academic Contributors

- Thijs van Ede
- Marco Caselli
- Andrea Continella
- Maarten van Steen
- Andreas Peter

CHAPTER

FIVE

LICENSE

MIT License

Copyright (c) 2022 Thijs van Ede

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER

SIX

CITING

To cite *py-attack* please use the following publication:

TODO

[PDF]

6.1 Bibtex

```
TODO
```