Carnegie Mellon University
Software Engineering Institute

Cyber DEM Python

Release 0.0.1

Austin Whisnant, Carnegie Mellon Universitty

September 25, 2020

CONTENTS:

CHAPTER ONE

CYBERDEM PYTHON LICENSE

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

CHAPTER TWO

README

2.1 Overview

CyberDEM Python provides a Python API for the Cyber Data Exchange Model (CyberDEM). CyberDEM Python provides methods to instantiate CyberDEM objects and events, serialize and deserialize objects and events, and manipulate instantiated objects. It also provides basic searching and file management methods.

2.2 Status

CyberDEM Python is based on the CyberDEM standard that is currently in DRAFT format, and therefore subject to change.

2.3 Getting Started

These instructions will help you install a copy of the package on your local machine.

2.3.1 Installing

1. Download CyberDEM Python and unzip the download folder
2. From within the top-level cyberdem folder (where setup.py is located) run

```
$ pip3 install .
```

3. To test that cyberdem is installed properly run

```
$ python3
>>> from cyberdem import base
>>> print(base.System())
System(
   id: 3bb3512e-dc75-4b86-b234-25040a79b9b9
)
```

You may also try running the example.py file downloaded with the zip file.

```
$ python3 example.py

Creating new FileSystem path ./test-fs.
```

(continues on next page)

(continued from previous page)

```
QUERY 1
--------
['description', 'id', 'name', 'version', '_type']
('Rapid SCADA software', '0f717dfa-...', 'Rapid SCADA', '5', 'Application')
('PfSense Firewall', '2720359e-...', 'PfSense', '2.4.2', 'Application')
(None, '2f6ac399-...', None, None, 'Application')
(None, 'd36e99ce-...', None, None, 'Application')
('Firefox browser', 'df8478d6-...', 'Firefox', '60', 'Application')
(None, '0eaacdbc-...', None, None, 'Application')
(None, '6265eb88-...', None, None, 'Application')

QUERY 2
--------
['description', 'name']
(None, None)
...
(None, None)
(None, None)
('PfSense Firewall', 'PfSense')
(None, None)
('Firefox browser', 'Firefox')
('Rapid SCADA software', 'Rapid SCADA')
...

QUERY 3: SELECT id FROM Application,OperatingSystem WHERE name='PfSense' OR␣
↪os_type='LinuxRedHat'
--------
['id']
('19a6f4b3-89ce-4aa5-8a94-a065833a3a53',)
('f177a5e0-f56a-4c11-b655-39e6c0cac873',)

Updating app versions...
Application(
   id: 19a6f4b3-89ce-4aa5-8a94-a065833a3a53
   name: PfSense
   version: 2.5.0
)
OperatingSystem(
   id: f177a5e0-f56a-4c11-b655-39e6c0cac873
   os_type: LinuxRedHat
   version: 8.0
)
```

You will see a folder called “test-fs” in the directory in which you ran the example.py script. This folder has subfolders containing each type of Cyber DEM object/event.

2.4 License

2.5 Acknowledgements

- SISO Cyber DEM Product Development Group

CHAPTER

THREE

BASE CLASSES

class `cyberdem.base.`**`Application`**(*version=None*, ***kwargs*)

Representation of an Application object.

Inherits *`_CyberObject`*.

Parameters

- **`version`** (*`string, optional`*) – Version of the application.
- **`kwargs`** (*`dictionary, optional`*) – Arguments to pass to the *`_CyberObject`* class

Example

```
>>> from cyberdem.base import Application
>>> kwargs = {
...     'version': '2.4.2',
...     'name': 'PfSense',
...     'description': 'PfSense Firewall'
...     }
>>> my_app = Application(**kwargs)
```

class `cyberdem.base.`**`BlockTrafficEffect`**(*is_random=None*, *percentage=None*, ***kwargs*)

Completely block all traffic over a communication channel.

Inherits *`Disrupt`*. No additional attributes.

Parameters **`kwargs`** (*`dictionary, optional`*) – Arguments to pass to the *`Disrupt`* class

Example

```
>>> from cyberdem.base import BlockTrafficEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'is_random': False,
...     'percentage': .7,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Continue',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> blocktraffic_effect = BlockTrafficEffect(**kwargs)
```

class `cyberdem.base.`**`CPULoadEffect`**(*percentage=None*, ***kwargs*)

Artificial increase in CPU load.

Inherits *`Degrade`*.

Parameters

- **percentage** (*float, optional*) – Percentage of CPU usage between 0.0 and 100.0
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import CPULoadEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'percentage': 70,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> cpuload_effect = CPULoadEffect(**kwargs)
```

class cyberdem.base.**CyberAttack**(*event_time=None, targets=None, target_modifiers=None, phase=None, duration=None, actor_ids=None, source_ids=None, **kwargs*)

Representation of a CyberAttack object.

Inherits *_CyberAction*. No additional attributes.

Example

```
>>> from cyberdem.base import CyberAttack
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'target_modifiers': {"characteristic":"value"},
...     'phase': 'Continue',
...     'duration': timedelta(seconds=10),
...     'actor_ids': [attacker_1.id]
... }
>>> generic_attack = CyberAttack(**kwargs)
```

class cyberdem.base.**CyberDefend**(*event_time=None, targets=None, target_modifiers=None, phase=None, duration=None, actor_ids=None, source_ids=None, **kwargs*)

Representation of a CyberDefend object.

Inherits *_CyberAction*. No additional attributes.

Example

```
>>> from cyberdem.base import CyberDefend
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'target_modifiers': {"characteristic":"value"},
...     'phase': 'Start',
...     'source_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> generic_defend = CyberDefend(**kwargs)
```

class cyberdem.base.**CyberRecon**(*recon_type=None*, ***kwargs*)

Representation of a CyberRecon object.

Inherits *_CyberAction*.

Parameters

- **recon_type** (value from the *ReconType* enumeration, optional) – Type of reconnaissance
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberAction* class

Example

```
>>> from cyberdem.base import CyberDefend
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'recon_type': 'NetworkMap',
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'source_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> recon_event = CyberRecon(**kwargs)
```

class cyberdem.base.**Data**(*sensitivity=None*, *data_type=None*, *encrypted=None*, *status=None*, *confidentiality=None*, ***kwargs*)

Representation of a Data object

Inherits *_CyberObject*.

Parameters

- **sensitivity** (value from *SensitivityType* enumeration, optional) – [desc]
- **data_type** (value from *DataType* enumeration, optional) – [desc]
- **encrypted** (value from *EncryptionType* enumeration, optional) – [desc]
- **status** (value from *DataStatus* enumeration, optional) – [desc]
- **confidentiality** (*float, optional*) – [desc]
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberObject* class

Example

```
>>> from cyberdem.base import Data
>>> kwargs = {
...     'sensitivity': 'FOUO',
...     'data_type': 'File',
...     'confidentiality': 0.6,
...     'name': 'Foo File',
...     'description': 'Foobarred file'
...     }
>>> my_data = Data(**kwargs)
```

class cyberdem.base.**DataExfiltration**(*event_time=None*, *targets=None*, *target_modifiers=None*, *phase=None*, *duration=None*, *actor_ids=None*, *source_ids=None*, ***kwargs*)

Data exfiltration is the unauthorized copying, transfer or retrieval of data from a computer or server. Data exfiltration is a malicious activity performed through various different techniques, typically by cybercriminals over the Internet or other network.

Inherits *CyberAttack*. No additional attributes.

Parameters **kwargs** (*dictionary, optional*) – Arguments to pass to the *CyberAttack* class

Example

```
>>> from cyberdem.base import DataExfiltration
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'phase': 'End',
...     'targets': [the_target.id],
...     'duration': timedelta(hours=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> exfil = DataExfiltration(**kwargs)
```

class cyberdem.base.**Degrade**(*event_time=None, targets=None, target_modifiers=None, phase=None, duration=None, actor_ids=None, source_ids=None, **kwargs*)

To deny access to, or operation of, a target to a level represented as a percentage of capacity

Inherits *Deny*. No additional attributes.

Parameters **kwargs** (*dictionary, optional*) – Arguments to pass to the *Deny* class

Example

```
>>> from cyberdem.base import Degrade
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> degrade_effect = Degrade(**kwargs)
```

class cyberdem.base.**DelayEffect**(*seconds=None, **kwargs*)

Increased time for data to travel between two points

Inherits *Degrade*.

Parameters

- **seconds** (*float, optional*) – Number of seconds to delay delivery of data
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import DelayEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'seconds': 22.5,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
```

(continues on next page)

(continued from previous page)

```
...     'duration': timedelta(minutes=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> delay_effect = DelayEffect(**kwargs)
```

class cyberdem.base.**Deny**(*event_time=None*, *targets=None*, *target_modifiers=None*, *phase=None*, *duration=None*, *actor_ids=None*, *source_ids=None*, ***kwargs*)

To prevent access to, operation of, or availability of a target function by a specified level for a specified time, by degrade, disrupt, or destroy (JP3-12)

Inherits *_CyberEffect*. No additional attributes.

Parameters **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberEffect* class

Example

```
>>> from cyberdem.base import Deny
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> deny_effect = Deny(**kwargs)
```

class cyberdem.base.**Destroy**(*event_time=None*, *targets=None*, *target_modifiers=None*, *phase=None*, *duration=None*, *actor_ids=None*, *source_ids=None*, ***kwargs*)

To completely and irreparably deny access to, or operation of, a target.

Inherits *Deny*. No additional attributes.

Parameters **kwargs** (*dictionary, optional*) – Arguments to pass to the *Deny* class

Example

```
>>> from cyberdem.base import Destroy
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> destroy_effect = Destroy(**kwargs)
```

class cyberdem.base.**Detect**(*acquired_information=None*, ***kwargs*)

To discover or discern the existence, presence, or fact of an intrusion into information systems.

Inherits *_CyberEffect*.

Parameters

- **acquired_information** (*dictionary, optional*) – information obtained during detection

- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberEffect* class

Example

```
>>> from cyberdem.base import Detect
>>> from datetime import datetime, timedelta
>>> info = {'siem log': 'file permission change on user-ws-2'}
>>> kwargs = {
...     'acquired_information': info,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> detect_effect = Detect(**kwargs)
```

class cyberdem.base.**Device**(*device_types=None*, *is_virtual=None*, *role=None*, *device_identifier=None*, *network_interfaces=None*, ***kwargs*)

Representation of a Device object.

Inherits *_CyberObject*.

Parameters

- **device_types** (value from the *DeviceType* enumeration, optional) – Type of device (ex. “Sensor”, “Printer”)
- **is_virtual** (*boolean, optional*) – whether the device is a virtual device
- **role** (*string, optional*) – [desc]
- **device_identifier** (*string, optional*) – [desc]
- **network_interfaces** (*list of tuples, optional*) – mapping of interface names to addresses
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberObject* class

Example

```
>>> from cyberdem.base import Device
>>> kwargs = {
...     'device_type': 'Generic',
...     'is_virtual': False,
...     'device_identifier':
↪'(01)12345678987654(55)120717(55)A12345B(55)4321',
...     'name': 'The Server',
...     'description': 'Generic server description'
...     }
>>> net_ints = [('eth0','204.105.24.23'), ('eth1','192.168.10.101')]
>>> my_device = Device(network_interfaces=net_ints, **kwargs)
```

class cyberdem.base.**Disrupt**(*is_random=None*, *percentage=None*, ***kwargs*)

To completely but temporarily deny access to, or operation of, a target for a period of time.

Inherits *Deny*.

Parameters

- **is_random** (*boolean, optional*) – whether or not the disruption is uniform or random

- **percentage** (*float, optional*) – Percentage of bits to drop between 0.0 and 100.0
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Deny* class

Example

```
>>> from cyberdem.base import Disrupt
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'is_random': False,
...     'percentage': .7,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> disrupt_effect = Disrupt(**kwargs)
```

class cyberdem.base.**DropEffect**(*percentage=None*, ***kwargs*)

Packet dropping.

Inherits *Degrade*.

Parameters

- **percentage** (*float, optional*) – Percentage of packets to drop between 0.0 and 100.0
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import DropEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'percentage': 99.5,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> pdrop_effect = DropEffect(**kwargs)
```

class cyberdem.base.**HardwareDamageEffect**(*damage_type=None*, ***kwargs*)

Physical damage to a device.

Inherits *Destroy*.

Parameters

- **damage_type** (value from the *HardwareDamageType* enumeration, optional) – type of damage
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Destroy* class

Example

```
>>> from cyberdem.base import HardwareDamageEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
```

(continues on next page)

(continued from previous page)

```
...     'damage_type': 'PhysicalDestruction',
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(days=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> hwdamage_effect = HardwareDamageEffect(**kwargs)
```

class cyberdem.base.**HardwareDegradeEffect**(*degrade_type=None*, *percentage=None*, ***kwargs*)

Degradation, but not destruction of, hardware.

Inherits *Degrade*.

Parameters

- **degrade_type** (*string, optional*) – value from the *HardwareDegradeType* enumeration
- **percentage** (*float, optional*) – The effectiveness of the hardware for the duration of the effect - between 0.0 and 100.0
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import HardwareDegradeEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'degrade_type': 'BlueScreen',
...     'percentage': 90,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> hw_effect = HardwareDegradeEffect(**kwargs)
```

class cyberdem.base.**JitterEffect**(*milliseconds=None*, ***kwargs*)

Class for JitterEffect object.

Inherits *Degrade*.

Parameters

- **milliseconds** (*float, optional*) – [desc]
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import JitterEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'milliseconds': 22.5,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'duration': timedelta(minutes=5)
```

(continues on next page)

(continued from previous page)

```
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> jitter_effect = JitterEffect(**kwargs)
```

class cyberdem.base.**LoadRateEffect**(*percentage=None*, *rate_type=None*, ***kwargs*)

Impact on data upload or download rate.

Inherits *Degrade*.

Parameters

- **percentage** (*float, optional*) – Percentage of maximum achievable rate between 0.0 and 100.0
- **rate_type** (*string, optional*) – value from the *LoadRateType* enumeration
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import LoadRateEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'percentage': 22.5,
...     'rate_type': 'Upload',
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> loadrate_effect = LoadRateEffect(**kwargs)
```

class cyberdem.base.**Manipulate**(*description=None*, ***kwargs*)

The effect of controlling or changing information, information systems, and/or networks to create physical denial effects, using deception, decoying, conditioning, spoofing, falsification, and other similar techniques.

Inherits *_CyberEffect*.

Parameters

- **description** (*string, optional*) – information obtained during detection
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberEffect* class

Example

```
>>> from cyberdem.base import Manipulate
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'description': "ransomware encrypted drives",
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'duration': timedelta(hours=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> manipulate_effect = Manipulate(**kwargs)
```

class cyberdem.base.**ManipulationAttack**(*description=None*, *attack_content=None*, ***kwargs*)

Controls or changes information, information systems, and/or networks to create physical denial effects, using deception, decoying, conditioning, spoofing, falsification, and other similar techniques.

Inherits *CyberAttack*.

Parameters

- **description** (*string, optional*) – Describes the "what and how" of the manipulation attack
- **attack_content** (*string, optional*) – could contain the details of the manipulation attack itself OR the manipulated message after the attack
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *CyberAttack* class

Example

```
>>> from cyberdem.base import DataExfiltration
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'event_time': datetime.today(),
...     'phase': 'End',
...     'targets': [the_target.id],
...     'duration': timedelta(hours=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> exfil = DataExfiltration(**kwargs)
```

class cyberdem.base.**MemoryUseEffect**(*percentage=None*, ***kwargs*)

Artificial increase in memory usage.

Inherits *Degrade*.

Parameters

- **percentage** (*float, optional*) – Percentage of memory to use between 0.0 and 100.0
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Degrade* class

Example

```
>>> from cyberdem.base import MemoryUseEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'percentage': 70,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> memuse_effect = MemoryUseEffect(**kwargs)
```

class cyberdem.base.**Network**(*protocol=None*, *mask=None*, ***kwargs*)

Representation of a Network object.

Inherits *_CyberObject*.

Parameters

- **protocol** (value from the *NetworkProtocolType* enumeration, optional) – protocol used on the network
- **mask** (*string, optional*) – network mask
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberObject* class

Example

```
>>> from cyberdem.base import Network
>>> kwargs = {
...     'protocol': 'OSPF',
...     'mask': '255.255.255.0',
...     'name': 'Network 10',
...     'description': 'User network'
... }
>>> my_network = Network(**kwargs)
```

class cyberdem.base.**NetworkLink**(*is_logical=None, physical_layer=None, data_link_protocol=None, bandwidth=None, latency=None, jitter=None, network_interfaces=None, **kwargs*)

Representation of a NetworkLink object.

Inherits *_CyberObject*.

Parameters

- **is_logical** (*boolean, optional*) – the link is logical (rather than physical)
- **physical_layer** (value from the *PhysicalLayerType* enumeration, optional) – what type is the physical layer
- **data_link_protocol** (value from the *DataLinkProtocolType* enumeration, optional) – data link protocol
- **bandwidth** (*integer, optional*) – Max data transfer rate of the link in Gb
- **latency** (*integer, optional*) – network link latency in milliseconds
- **jitter** (*integer, optional*) – variability in the latency, measured in milliseconds
- **network_interfaces** (*list of tuples, optional*) – mapping of interface names to addresses
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberObject* class

Example

```
>>> from cyberdem.base import NetworkLink
>>> kwargs = {
...     'is_logical': False,
...     'physical_layer': 'Wired',
...     'data_link_protocol': 'Ethernet',
...     'bandwidth': 5,
...     'name': 'Link 10',
...     'description': 'User network link'
... }
>>> net_ints = [('eth1','192.168.10.100'), ('eth0','192.168.10.101')]
>>> my_link = NetworkLink(network_interfaces=net_ints, **kwargs)
```

class cyberdem.base.**OperatingSystem**(*os_type=None*, ***kwargs*)

Representation of a OperatingSystem object.

Inherits `_CyberObject`.

Parameters

- **os_type** (value from the `OperatingSystemType` enumeration, optional) – Type of operating system
- **kwargs** (*dictionary, optional*) – Arguments to pass to the `_CyberObject` class

Example

```
>>> from cyberdem.base import OperatingSystem
>>> kwargs = {
...     'os_type': 'MicrosoftWindows',
...     'name': 'User machine',
...     'description': 'For employees in foo department'
... }
>>> my_os = OperatingSystem(**kwargs)
```

class cyberdem.base.**OtherDegradeEffect**(*percentage=None*, *description=None*, ***kwargs*)

Generic degradation effect.

Inherits `Degrade`.

Parameters

- **percentage** (*float, optional*) – Percentage of impacted capability's remaining availability between 0.0 and 100.0
- **description** (*string, optional*) – Human-interpretable or machine-readable description of the effect
- **kwargs** (*dictionary, optional*) – Arguments to pass to the `Degrade` class

Example

```
>>> from cyberdem.base import OtherDegradeEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'degrade_type': 'BlueScreen',
...     'percentage': 90,
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'phase': 'Start',
...     'duration': timedelta(seconds=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> other_effect = OtherDegradeEffect(**kwargs)
```

class cyberdem.base.**PacketManipulationEffect**(*manipulation_type=None*, *attack_content=None*, ***kwargs*)

Packet manipulation cyber effect: duplication, corruption, reordering, drop.

Inherits `Manipulate`.

Parameters

- **manipulation_type** (value from `PacketManipulationType` enumeration, optional) – type of manipulation

- **percentage** (*float, optional*) – Percentage of packets to affect between 0.0 and 100.0
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *Manipulate* class

Example

```
>>> from cyberdem.base import PacketManipulationEffect
>>> from datetime import datetime, timedelta
>>> kwargs = {
...     'manipulation_type': 'Dropped',
...     'percentage': 1,
...     'description': "dropping packets",
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'duration': timedelta(hours=5)
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
... }
>>> packet_effect = PacketManipulationEffect(**kwargs)
```

`class cyberdem.base.Persona(name=None, description=None, **kwargs)`

Representation of a Personna object.

Inherits *_CyberObject*. No additional attributes.

Example

```
>>> from cyberdem.base import Persona
>>> kwargs = {
...     'name': 'Attacker 1',
...     'description': 'nation-state actor'
... }
>>> attacker_1 = Persona(**kwargs)
```

`class cyberdem.base.PhishingAttack(message_type=None, header=None, body=None, **kwargs)`

The fraudulent practice of sending messages purporting to be from reputable sources in order to induce individuals to reveal sensitive information or unknowingly initiate another attack.

Inherits *CyberAttack*.

Parameters

- **message_type** (value from the *MessageType* enumeration, optional) – type of message. Ex. “Email”
- **header** (*string, optional*) – Originator, From, To, Subject, Reply To
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *CyberAttack* class

Example

```
>>> from cyberdem.base import PhishingAttack
>>> from datetime import datetime
>>> kwargs = {
...     'message_type': 'Email',
...     'header': 'From: Your Name <yourname@foo.bar>, To: My Name ␣
↪       <myname@bar.foo>, Subject: foo the bar',
...     'event_time': datetime.today(),
...     'targets': [the_target.id],
...     'actor_ids': ["77545b7d-3900-4e34-a26f-eec5eb954d33"]
```

(continues on next page)

(continued from previous page)

```
... }
>>> phish = PhishingAttack(**kwargs)
```

class cyberdem.base.**Service**(*service_type=None*, *address=None*, ***kwargs*)

Representation of a Service object.

Inherits Application.

Parameters

- **service_type** – Type of service
- **address** (*string, optional*) –
- **kwargs** (*dictionary, optional*) – Arguments to pass to the Application class

Example

```
>>> from cyberdem.base import Service
>>> kwargs = {
...     'service_type': 'EmailServer',
...     'version': '15.2.595.4',
...     'name': 'Mail Server 1',
...     'description': 'external exchange server'
... }
>>> my_service = Service(**kwargs)
```

class cyberdem.base.**System**(*system_type=None*, ***kwargs*)

Representation of a System object.

Inherits _CyberObject.

Parameters

- **system_type** (value from the SystemType enumeration, optional) – Type of system
- **kwargs** (*dictionary, optional*) – Arguments to pass to the _CyberObject class

Example

```
>>> from cyberdem.base import System
>>> kwargs = {
...     'system_type': 'SCADA',
...     'name': 'MTU',
...     'description': 'Network 1 MTU'
... }
>>> my_system = System(**kwargs)
```

class cyberdem.base.**_CyberAction**(*event_time=None*, *targets=None*, *target_modifiers=None*, *phase=None*, *duration=None*, *actor_ids=None*, *source_ids=None*, ***kwargs*)

Passive superclass for all CyberDEM CyberActions.

Inherits _CyberEvent. Included for completeness of the CyberDEM standard.

class cyberdem.base.**_CyberDEMBase**(*id=None*, ***kwargs*)

Superclass for all CyberDEM Objects and Events

Will create an appropriate id if one is not given.

Parameters **id** (*string, optional*) – string formatted UUIDv4

Raises ValueError – if a given id is not a valid string representation of a UUIDv4

class cyberdem.base.**_CyberEffect**(*event_time=None*, *targets=None*, *target_modifiers=None*, *phase=None*, *duration=None*, *actor_ids=None*, *source_ids=None*, ***kwargs*)

Passive superclass for all CyberDEM CyberEffects.

Inherits *_CyberEvent*. No additional attributes.

class cyberdem.base.**_CyberEvent**(*event_time=None*, *targets=None*, *target_modifiers=None*, *phase=None*, *duration=None*, *actor_ids=None*, *source_ids=None*, ***kwargs*)

Superclass for all CyberDEM CyberEvents

CyberEvents are non-persistent cyber events, as opposed to persistent CyberObjects.

Inherits *_CyberDEMBase*. Optionally sets the event_time, targets, cyber event phase, duration, actor_ids, and/or source_ids parameters for any CyberEvent subclass.

Parameters

- **event_time** (*datetime.datetime, optional*) – Time at which the event started
- **targets** (*list, optional*) – One or more IDs identifying the CyberObject(s) targeted in the event
- **target_modifiers** (*dictionary, optional*) – mapping of target characteristics to values
- **phase** (value from *CyberEventPhaseType* enumeration, optional) – The cyber event phase of the event
- **duration** (*datetime.timedelta, optional*) – Length of time the event lasted
- **actor_ids** (*list, optional*) – Time ordered list of IDs of the perpetrators involved in this Cyber Event
- **source_ids** (*list, optional*) – Time ordered list of IDs of the simulations that this Cyber Event came from.
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberDEMBase* class

class cyberdem.base.**_CyberObject**(*name=None*, *description=None*, ***kwargs*)

Superclass for all CyberDEM CyberObjects

CyberObjects are persistent objects on a network or other cyber infrastructure.

Inherits *_CyberDEMBase*. Optionally sets the name and/or description parameters for any CyberObject subclass.

Parameters

- **name** (*string, optional*) – The name of the object
- **description** (*string, optional*) – A description of the object
- **kwargs** (*dictionary, optional*) – Arguments to pass to the *_CyberDEMBase* class

CHAPTER FOUR

STRUCTURES

CyberDEM Structures module provides classes for CyberDEM data types.

class cyberdem.structures.**Relationship**(*related_object_1*, *related_object_2*, *relationship_type=None*, *id=None*, *privileges=None*)

Represents a relationship between two CyberObjects.

Given two CyberObjects A and B, where A administers B, the `related_object_1` would be the id of A and `related_object_2` would be the ID of B, preserving the ordering of "A administers B".

Parameters

- **related_object_1** (*UUIDv4 string, required*) – ID of a CyberObject
- **related_object_2** (*UUIDv4 string, required*) – ID of a CyberObject
- **relationship_type** (*string, optional*) – value from *RelationshipType*
- **id** (*string, optional*) – unique ID (UUIDv4)
- **privileges** (*list of strings, optional*) – [desc]

Example

```
>>> # Where my_application is installed on my_device
>>> from cyberdem.base import Application, Device
>>> from cyberdem.structures import Relationship
>>> my_application = Application()
>>> my_device = Device()
>>> my_rel = Relationship(
...     my_device.id, my_application.id,
...     relationship_type='ResidesOn', privileges=['priv1', 'priv2'])
```

CHAPTER FIVE

FILESYSTEM

Methods for saving, searching, and retrieving CyberDEM objects and events.

class cyberdem.filesystem.**FileSystem**(*path*)

Create a directory structure and file storage and retrieval methods.

Creates file storage, retrieval, and query methods for storing and retrieving CyberObjects, CyberEvents, and Relationships.

Parameters **path** (*string, required*) – directory path to store CyberDEM json files; can be existing directory or non-existing

Example

```
>>> from cyberdem import filesystem
>>> fs = filesystem.FileSystem("./test-fs")
Using existing FileSystem at ./test-fs
>>> fs.path
'./test-fs'
```

get(*ids*, *obj_type=None*)

Get an object or list of objects by ID

Parameters

- **ids** (*string or list of strings, required*) – UUID(s) of object(s) to retrieve
- **obj_type** (*string, optional*) – CyberDEM type of the id(s). Ex. “Application”

Returns instance(s) of the requested object(s)

Return type object or list of objects, or None if no matching IDs are found

Example

```
>>> my_object = fs.get("82ca4ed1-a053-4fc1-b1cc-f4b58b4dbf8c",                ␣
↪          "Application")
>>> str(my_object)
Application(
    id: 82ca4ed1-a053-4fc1-b1cc-f4b58b4dbf8c
)
>>> my_objects = fs.get(["46545b7a-1840-4e34-a26f-aef5eb954b25",              ␣
↪              "82ca4ed1-a053-4fc1-b1cc-f4b58b4dbf8c"])
>>> for obj in my_obects:
...     print(obj)
Service(
    name='httpd',
```

(continues on next page)

(continued from previous page)

```
    description='Apache web server',
    version='2.4.20',
    service_type='WebService',
    address='192.168.100.40'
)
Application(
    id: 82ca4ed1-a053-4fc1-b1cc-f4b58b4dbf8c
)
```

query (*query_string*)

Search the FileSystem for a specific object or objects

Parameters **query_string** (*string, required*) – SQL formatted query string

Returns attribute names (headers), values of matching objects

Return type 2-tuple of lists

Example query strings

- `SELECT * FROM *` (you probably shouldn't do this one…)
- `SELECT attr1,attr2 FROM * WHERE attr3=value`
- `SELECT id,name,description FROM Device,System WHERE name='my device'`
- `SELECT id FROM * WHERE (name='foo' AND description='bar') OR version<>'foobar'`

Example

```
>>> query = "SELECT id FROM * WHERE name='Rapid SCADA'"
>>> fs.query(query)
(['id'], [('9293510b-534b-4dd0-b7c5-78d92e279400',)])
>>> query = "SELECT id,name FROM Application"
>>> headers,results = fs.query(query)
>>> headers
['id','name']
>>> results
[('9293510b-534b-4dd0-b7c5-78d92e279400',),          ('46545b7a-
↪1840-4e34-a26f-aef5eb954b25','My application')]
```

save (*objects, overwrite=False*)

Save CyberDEM objects and events to the FileSystem as json files

Parameters

- **objects** (CyberDEM class instance from `base`, or a list of objects) – CyberDEM object or event instance (or list of instances)
- **overwrite** (*bool, optional*) – allow object with the same ID as one already in the FileSystem to overwrite the existing file, defaults to False

Raises **Exception** – if object is already in FileSystem and overwrite is set to False

Example

```
>>> from cyberdem.base import Service
>>> my_service = Service(
        name='httpd', description='Apache web server',                 ␣
↪       version='2.4.20', service_type='WebService',                    ␣
↪     address='192.168.100.40')
```

(continues on next page)

(continued from previous page)

```
>>> fs.save(my_service)
>>>
$ ls ./test-fs/Service
82ca4ed1-a053-4fc1-b1cc-f4b58b4dbf8c.json
```

CHAPTER

SIX

ENUMERATIONS

class cyberdem.enumerations.**CyberEventPhaseType**

CyberDEM CyberEventPhaseType enumeration

Options 'Continue', 'ContinueWithChanges', 'End', 'Start', 'Suspend'

class cyberdem.enumerations.**DataLinkProtocolType**

CyberDEM DataLinkProtocolType enumeration

Options 'ATM', 'Bluetooth', 'Ethernet', 'LocalTalk', 'PPP', 'TokenRing', 'VLAN', 'WiFi', '1553Bus'

class cyberdem.enumerations.**DataStatus**

CyberDEM DataStatus enumeration

Options 'Corrupted', 'Erased', 'Manipulated', 'NonDecryptable', 'Uncompromised'

class cyberdem.enumerations.**DataType**

CyberDEM DataType enumeration

Options 'Code', 'Credentials', 'File'

class cyberdem.enumerations.**DeviceType**

CyberDEM DeviceType enumeration

Options 'Communications', 'ComputerNode', 'Controller', 'Generic', 'HMI', 'IoT', 'Monitoring', 'Networking', 'PortableComputer', 'Printer', 'Scanner', 'Sensor', 'StorageDevice'

_check_prop (*value*)

Checks to see if `value` is an allowed enumeration value.

Overrides the *_check_prop()* function from the super *_CyberDEMEnumeration*

Campares the given value to the allowed options for the current enumeration class (sub-class to *_CyberDEMEnumeration*).

Parameters **value** (*list, required*) – user-provided value for the enumeration type

Raises **ValueError** – if the values in `value` are not in the allowed options.

class cyberdem.enumerations.**EncryptionType**

CyberDEM EncryptionType enumeration

Options 'AES', 'DES', 'RSA', 'SHA', 'TripleDES', 'TwoFish'

class cyberdem.enumerations.**HardwareDamageType**

CyberDEM HardwareDamageType enumeration

Options 'BootLoop', 'HardDriveErased', 'PhysicalDestruction',

class cyberdem.enumerations.**HardwareDegradeType**
CyberDEM HardwareDegradeType enumeration

Options 'BlueScreen', 'Display', 'Keyboard', 'Mouse', 'RandomText', 'Reboot', 'Sound'

class cyberdem.enumerations.**LoadRateType**
CyberDEM LoadRateType enumeration

Options 'Download', 'Upload'

class cyberdem.enumerations.**MessageType**
CyberDEM MessageType enumeration

Options 'Chat', 'Email', 'SocialMedia', 'Text'

class cyberdem.enumerations.**NetworkProtocolType**
CyberDEM NetworkProtocolType enumeration

Options 'ARP', 'ICMP', 'InternetProtocol', 'IPsec', 'NAT', 'OSPF', 'RIP'

class cyberdem.enumerations.**OperatingSystemType**
CyberDEM OperatingSystemType enumeration

Options 'Android', 'AppleiOS', 'AppleMacOS', 'BellLabsUnix', 'BSDUnix', 'CiscoIOS', 'DECHP_UX', 'DECVMS', 'Firmware', 'GNUUnix', 'IBMOS_2', 'LinuxRedHat', 'MicrosoftDOS', 'MicrosoftWindows', 'OpenSolaris', 'Ubuntu'

class cyberdem.enumerations.**PacketManipulationType**
CyberDEM PacketManipulationType enumeration

Options 'Corruption', 'Dropped', 'Duplication', 'Redordering'

class cyberdem.enumerations.**PhysicalLayerType**
CyberDEM PhysicalLayerType enumeration

Options 'Wired', 'Wireless'

class cyberdem.enumerations.**ReconType**
CyberDEM ReconType enumeration

Options 'AccountEnumeration', 'ARPScan', 'DeviceEnumeration', 'DNSEnumeration', 'DomainEnumeration', 'LDAPScan', 'NetBiosScan', 'NetworkMap', 'NTPEnumeration', 'OSScan', 'Ping', 'PingScan', 'PortScan', 'PortSweep', 'ServiceEnumeration', 'SMTPEnumeration', 'SNMPSweep', 'TraceRoute', 'UNIX-LinuxEnumeration', 'VulnerabilityEnumeration', 'WindowsEnumeration'

class cyberdem.enumerations.**RelationshipType**
CyberDEM RelationshipType enumeration

Options 'Administers', 'ComponentOf', 'ContainedIn', 'ProvidedBy', 'ResidesOn'

class cyberdem.enumerations.**SensitivityType**
CyberDEM SensitivityType enumeration

Options 'Confidential', 'CosmicTopSecret', 'FOUO', 'FVEY', 'GDPR', 'HIPPA', 'NATOConfidential', 'NATORestricted', 'NATOSecret', 'PII', 'Proprietary', 'Public', 'Secret', 'SecretNoForn', 'TS', 'TS_SCI', 'Unclassified'

class cyberdem.enumerations.**ServiceType**
CyberDEM ServiceType enumeration

Options 'ChatServer', 'DatabaseServer', 'DomainNameServer', 'EmailServer', 'FileShare', 'Forum', 'SocialMediaServer', 'WebService'

class cyberdem.enumerations.**SystemType**

CyberDEM SystemType enumeration

Options 'C2', 'Generic', 'ICS', 'SCADA'

class cyberdem.enumerations.**_CyberDEMEnumeration**

Super class for all CyberDEM enumerations

_check_prop(*value*)

Checks to see if `value` is an allowed enumeration value.

Campares the given value to the allowed options for the current enumeration subclass.

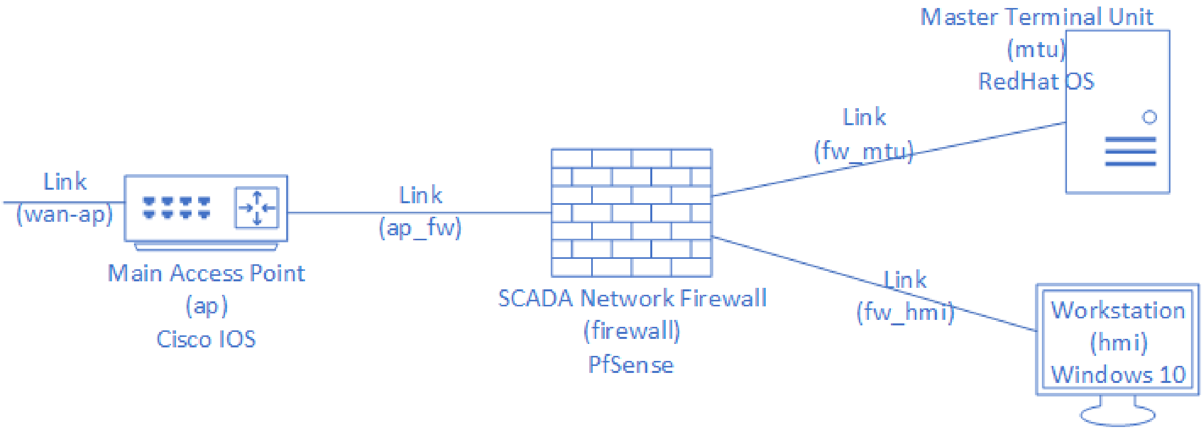
Parameters **value** (*required*) – user-provided value for the enumeration type

Raises **ValueError** – if the `value` is not in the allowed options of the enumeration class.

CHAPTER SEVEN

EXAMPLE

The following is an example script covering the use of the modules and methods in CyberDEM Python. It models the toy SCADA network in the figure below and instantiates a series of events on a particular attack chain.



License for example.py

```
'''
CyberDEM Example Script

CyberDEM Python

Copyright 2020 Carnegie Mellon University.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE
MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO
WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER
INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR
MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL.
CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT
TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Released under a MIT (SEI)-style license, please see license.txt or contact
permission@sei.cmu.edu for full terms.

[DISTRIBUTION STATEMENT A] This material has been approved for public release
and unlimited distribution.  Please see Copyright notice for non-US Government
use and distribution.

DM20-0711
'''
```

Import the necessary modules.

```
from cyberdem.filesystem import FileSystem
from cyberdem.base import *
from cyberdem.structures import Relationship
from datetime import datetime, timedelta
```

Set up the file system to store CyberDEM objects and events.

```
    # Set up the FileSystem for storage and retrieval of CyberDEM objects
    fs = FileSystem('./test-fs')
```

Instantiate a set of known cyber objects representing the toy network. Save each of the objects to the file system.

```
    # Instantiate a known set of CyberObjects
    ap = Device(
        name="Access Point", description="Main access point", is_virtual=False,
        network_interfaces=[("eth0", "10.10.30.40"), ("eth1", "192.168.10.2")])
    fs.save(ap)

    firewall = Device(
        name="Firewall", description="Firewall", is_virtual=False,
        network_interfaces=[
                ("eth0", "192.168.10.3"), ("eth1", "192.168.10.4"),
                ("eth2", "192.168.10.5")])
    fs.save(firewall)

    mtu = Device(
        name="MTU", description="Master Terminal Unit", is_virtual=False,
        network_interfaces=[("eth0", "192.168.10.6")])
    fs.save(mtu)

    hmi = Device(
        name="HMI", description="HMI Workstation", device_types=['HMI'],
        is_virtual=False, network_interfaces=[("eth0", "192.168.10.7")])
    fs.save(hmi)

    wan_ap = NetworkLink(
        name='WAN-AP', description='WAN to AP link', physical_layer='Wired',
        is_logical=False, data_link_protocol='Ethernet')
    fs.save(wan_ap)

    ap_fw = NetworkLink(
        name='AP-FW', description='AP to FW link', physical_layer='Wired',
        is_logical=False, data_link_protocol='Ethernet')
    fs.save(ap_fw)

    fw_mtu = NetworkLink(
        name='FW-MTU', description='FW to MTU link', physical_layer='Wired',
        is_logical=False, data_link_protocol='Ethernet')
    fs.save(fw_mtu)

    fw_hmi = NetworkLink(
        name='FW-HMI', description='FW to HMI link', physical_layer='Wired',
        is_logical=False, data_link_protocol='Ethernet')
    fs.save(fw_hmi)

    cisco_ios = OperatingSystem(
```

(continues on next page)

(continued from previous page)

```
        name='Cisco IOS', description='AP OS is Cisco IOS', version='15.4(3)M',
        os_type='CiscoIOS')
    fs.save(cisco_ios)

    redhat = OperatingSystem(
        nam='RedHat', description='RedHat OS', version='8',
        os_type='LinuxRedHat')
    fs.save(redhat)

    win_10 = OperatingSystem(
        name='Windows 10', description='HMI OS is Win10',
        version='Win 10, 2004', os_type='MicrosoftWindows')
    fs.save(win_10)

    pfsense = Application(
        name='PfSense', description='PfSense Firewall', version='2.4.2')
    fs.save(pfsense)

    firefox = Application(
        name='Firefox', description='Firefox browser', version='60')
    fs.save(firefox)

    rapid_scada = Application(
        name='Rapid SCADA', description='Rapid SCADA software', version='5')
    fs.save(rapid_scada)

    httpd_service = Service(
        name='httpd', description='Apache web server', version='2.4.20',
        service_type='WebService', address='192.168.100.40')
    fs.save(httpd_service)

    generic_admin = Persona(
        name="Network admin", description="Runs the systems")
    fs.save(generic_admin)
```

Create and save extra CyberObjects (4 of each type) for background data.

```
    # Save a bunch of random objects
    obj_types = [
        Application, Data, Device, Network, NetworkLink, Persona,
        System, OperatingSystem, Service, Deny, Detect, Manipulate,
        DataExfiltration, Destroy, Degrade, Disrupt, PacketManipulationEffect,
        ManipulationAttack, PhishingAttack, BlockTrafficEffect,
        HardwareDamageEffect, LoadRateEffect, DelayEffect, JitterEffect,
        CPULoadEffect, MemoryUseEffect, DropEffect, HardwareDegradeEffect,
        OtherDegradeEffect]
    for ot in obj_types:
        for _ in range(0, 4):
            fs.save(ot())
```

Create and save relationships between the known CyberObjects.

```
    # Build relationships between the known objects
    fs.save(Relationship(ap.id, wan_ap.id))
    fs.save(Relationship(ap.id, ap_fw.id))
    fs.save(Relationship(ap.id, cisco_ios.id))
    fs.save(Relationship(firewall.id, ap_fw.id))
```

(continues on next page)

(continued from previous page)

```
    fs.save(Relationship(firewall.id, fw_mtu.id))
    fs.save(Relationship(firewall.id, fw_hmi.id))
    fs.save(Relationship(
        redhat.id, firewall.id, relationship_type='ResidesOn'))
    fs.save(Relationship(mtu.id, fw_mtu.id))
    fs.save(Relationship(redhat.id, mtu.id, relationship_type='ResidesOn'))
    fs.save(Relationship(
        httpd_service.id, mtu.id, relationship_type='ResidesOn'))
    fs.save(Relationship(
        rapid_scada.id, mtu.id, relationship_type='ResidesOn'))
    fs.save(Relationship(hmi.id, fw_hmi.id))
    fs.save(Relationship(win_10.id, hmi.id, relationship_type='ResidesOn'))
    fs.save(Relationship(firefox.id, hmi.id, relationship_type='ResidesOn'))
```

Create the CyberEvents of an attack on the toy network.

```
    # Save the  events of a specific attack chain against the toy network
    # phishing attack via email targeting the SCADA administrator
    fs.save(PhishingAttack(
        message_type='Email', targets=[generic_admin.id],
        event_time=datetime(2020, 9, 18)))
    # actor on the HMI ping scans the network block
    scada_netblock = NetworkLink(mask="192.168.10.0/24")
    fs.save(scada_netblock)
    fs.save(CyberRecon(
        recon_type="PingScan", event_time=datetime(2020, 9, 19),
        targets=[scada_netblock.id], duration=timedelta(seconds=300),
        source_ids=[hmi.id]))
    # actor using the HMI installs a malicious file on the MTU
    fs.save(Manipulate(
        description="installation of malicious file",
        event_time=datetime(2020, 9, 19), targets=[mtu.id],
        source_ids=[hmi.id]))
    fs.save(Manipulate(
        description="malicious code changes readings on MTU",
        event_time=datetime(2020, 9, 19), targets=[mtu.id], phase='Continue'))
```

Query the file system for objects/events with various characteristics.

```
    # Query the file system
    headers, resp = fs.query("SELECT * FROM Application")
    print(f'\nQUERY 1: SELECT * FROM Application\n--------\n{headers}')
    for line in resp:
        print(line)

    query2 = (
        "SELECT description,name FROM * "
        "WHERE (is_logical=False AND physical_layer<>'Wired') OR "
        "event_time<>'2020-09-18'")
    # @TODO query2 is not currently returning what is expected
    headers, resp = fs.query(query2)
    print(f'\nQUERY 2\n--------\n{headers}')
    for line in resp:
        print(line)

    query3 = (
        "SELECT id FROM Application,OperatingSystem "
```

(continues on next page)

(continued from previous page)

```
        "WHERE name='PfSense' OR os_type='LinuxRedHat'")
    headers, resp = fs.query(query3)
    print(f'\nQUERY 3: {query3}\n--------\n{headers}')
    for line in resp:
        print(line)
```

Reinstantiate objects found in the last query, update their version numbers, and re-save to the filesystem.

```
    # Load instances of objects and events from the file system
    # change some property and re-save the instance
    print("\nUpdating app versions...")
    for line in resp:
        app = fs.get(line[0])
        del app.description
        if isinstance(app, Application):
            app.version = '2.5.0'
        else:
            app.version = '8.0'
        print(app)
        fs.save(app, overwrite=True)
```

PYTHON MODULE INDEX

INDEX