MTDSimTime Document

Wenxiao Zhang

24 May 2023

Contents

1 Installation

1. Installing conda

2. Creating conda environment

```
conda env create -f environment.yml
conda config --add channels conda-forge
```

3. Activating the environment

```
conda activate mtdsimtime
```

4. Updating the environment

```
conda env update --name mtdsimtime --file environment.yml --prune
```

5. Running examples in simulation.ipynb

2 Architecture

Check thesis section 4.1 .

3 Directory Structure and Overview

3.1 component

This directory contained the code that defined essential components of the simulated system.

3.1.1 Adversary

adversary.py defines an object called Adversary that represents an adversary in a network. The Adversary class has an initializer (__init__) which takes two parameters: network (the simulation network) and attack_threshold (set in constant value). It initializes various instance variables that are used to track the state of the adversary in the network.

3.1.2 Host

host.py defines a Host object in a simulation of a network. The Host class represents a network host with an operating system, a unique ID, an IP address, and a set of internal services that it runs.

3.1.3 MTDScheme

mtd_scheme.py defined the MTDScheme class, which was used for implementing and selecting the execution schemes (thesis section 4.3.2) for executing MTD.

3.1.4 Network

network.py defined the simulated network object (thesis section 4.2).

3.1.5 Services

services.py defined classes and functions related to generating and managing services and vulnerabilities in a simulated network system. It included Vulnerability, Service, and ServicesGenerator.

The Vulnerability class represented a vulnerability that can be assigned to a set of versions for a service.

The Service class represented a service that can have multiple vulnerabilities.

The ServicesGenerator class was responsible for generating and managing services for the simulation.

3.1.6 TargetNetwork

target_network.py defined a class TargetNetwork that inherits from a class Network. The TargetNetwork class adds some additional functionality to the base Network class. This class aims to separate the targetted attack scenario from the Network class while inheriting its functionalities, but the graph generation function used for the targetted attack scenario was commented out in the Network class right after graph_gen.

3.1.7 TimeGenerator

time_generator.py the original purpose for creating this is to encapsulate different distribution methods in the scipy, enabling efficient generation of time values based on different distribution approaches. However, for the scope of this work, only the "exponential_variates" method is utilized and no further changes are introduced.

3.1.8 TimeNetwork

time_network.py defines TimeNetwork class, which is derived from the Network class. It extends the functionality of the base Network class by adding additional attributes and methods specific to time-related network simulations.

3.2 data

This directory contained sample data and constants utilized in the simulator.

3.2.1 Constants

constants.py used for setting up various parameter values and constants.

3.2.2 User

first-names.txt used for setting up user names on each host for UserShuffle-related MTD functionality.

3.2.3 Services

words.txt used for generating services with different names.

3.3 mtd

This directory contained the implementation of various MTD techniques. Each implementation is derived from MTD class in the __init.py__.

3.4 operation

This directory contained the implementation of MTD operation (thesis section 4.3.3) and Attack operation (thesis section 4.4.2).

3.5 snapshot

This directory contained the implementation of snapshot functionality, enabling the saving and loading of network or adversary object instances based on simulation time or network size. The code was a bit messy and can be optimized and expanded to better accommodate custom use cases.

3.6 statistic

This directory contained data recording and evaluation methods implemented in this work and previous works. Although it can provide insights into the simulation, it is recommended to customize your own methods for specific use cases.