**DHALSIM source code**

**Installation note:**

Only installs properly on Ubuntu 20.04.

Tried to install on Fedora, didnt work.

**Config files and how to Run**

> dhalsim <main\_config.yml>

There needs to be following config files:

1. the main\_config.yml (Main configuration file, specifies how many iteration the experiment will run and location of other config files)

2. map.inp (WNTR input file that describes the water network - this is the water plant or physical part of the cyber physical system)

3. demand.csv (water demand pattern for the water WNTR network)

4. plc.yml (details of each plc - cyber part)

5. event.yml (specifies what kind of event - like packet loss, or packet delay. or some kind of attack. Specifies target plc)

6. delay\_loss.yml (specifies details of above mentioned event)

7. dos.yml, mitm.yml etc (specifies details of the attack)

**Output**

1. a folder containing the configuration used

2. scada.csv (tank and pump readings for each iteration with timestamp. These are values for cyber part)

3. ground\_truth.csv (tank and pump readings for each iteration with timestamp. These are values for physical part)

4. .pcap files for each network interface.

**Execution flow and explanation of source files**

1. Location of the executable script is /usr/local/bin/dhalsim . When it runs it basically loads the directory of the DHALSIM python files. In effect dhalsim/command\_line.py starts executing.

2. dhalsim/command\_line.py

- it runs mininet

- it runs /dhalsim/python2/automatic\_run.py as a script (see below for its function)

3. dhalsim/python2/automatic\_run.py

This file launches cyber and physical simulation by calling several python files in /dhalsim//python2/ directory. These files are:

a. automatic\_plc.py

b. automatic\_scada.py

c. automatic\_attacker.py

d. automatic\_event.py

e. automatic\_plant.py

The above files/scripts are run as subprocess, so each run in separate threads.

It also kills all the subprocesses after everything is done.

It also writes output file output/configuration/mininet\_links.md before closing everything.

Below are the details of the scripts that are launched as subprocess

4. dhalsim/python2/automatic\_plant.py

It is responsible for launching the physical simulation and it does that by running dhalsim/physical\_process.py

a) dhalsim/physical\_process.py

- it basically launches the physical simulation using wntr and epynet.

- uses map.inp config file for wntr.

- writes the output in output/ground\_truth.csv file. In the process updates relevant DB table for each iteration.

5. dhalsim/python2/automatic\_plc.py

- It is responsible to simulate PLCs and attack on them,

- It writes the output in .pcap files for each network interface

- all the details of PLC simulation is actually in generic\_plc.py, so it calls that script

a. dhalsim/python2/generic\_plc.py

- the main class is a subclass of BasePlc

- This actually simulates attacks and events

- The detail of event/attack is in entities dhalsim/python2/submodule. This submodule defines an attack that is executed on a certain trigger. A trigger is a tag going above a certain value. For example, we can define an attack that starts when a tank level goes above a given value.

6. halsim/python2/automatic\_scada.py

This controller for SCADA server.

tcpdump output is written in output/scada-eth0.pcap

SCADA server itself is implemented in /halsim/python2/generic\_scada.py file, which is launched from here

a. halsim/python2/generic\_scada.py

- Here SCADA server is defined.

- all the details/description of the SCADA server like name, scada protocol, name/tag for each sensor are difined.

- SCADA server will be running in loop, here functionalities for a single loop is defined in main\_loop() function.

- database, cache will be updated for each loop

7. c. halsim/python2/automatic\_attacker.py

This controls the attacks.

reads attack details from dos.yml, mitm.yml files.

writes tcpdump output in .pcap file

actual attack details are in /dhalsim/network\_attacks module.

some attack files are:

Simple\_dos\_attack.py

1. attack launches an ARP spoofing attack and then stops forwarding CIP packets to the target PLC
2. uses scapy package to generate ARP
3. uses iptables to direct packets

8. halsim/python2/automatic\_event.py

similarly this controls events (packet loss, delay).

actual event details are in /dhalsim/network\_events module.

files in /dhalsim/network\_events

delay\_and\_loss.py

network\_delay.py

packet\_loss.py

All use tc qdisc add/del dev to implement these events.

**Database**

There are different components in this simulation program and they work in unison. That means output from one component is often used as input for other component. Since different components run as subprocesses in different threads, it is difficult to communicate between the components. Database has been used as the interface to communicate among the components. There are 5 tables in the database:

plant

master\_time

sync

attack

event

These tables are created temporarily. After simulation they get deleted.