Cooja Simulator for IoT Security

The simulator is based on Contiki OS, used for IoT systems.

1. Install the simulator

You need to run a VM, so a virtualization software is needed:

- VirtualBox
- Parallels
- VMWare

1. When you have it available, download the Cooja VM from this link and extract the files https://sourceforge.net/projects/contiki/files/Instant%20Contiki/Instant%20Contiki%203.0/InstantContiki3.0.zip/download
2. Run VirtualBox (in my case) and create a new VM called
   - Name: "Contiki3.0"
   - Type: Linux
   - Version: Ubuntu 32-bit
   Click next.
3. Select memory and cores as you prefer
4. Choose to install "**Instant_Contiki_Ubuntu_12.04_32-bit.vmdk**" and finish the installation
   The password for the `user` is `user`.

1.A Ubuntu VM (Suggested)

From a clean installation of the OS:

```
Installed as https://githuib.com/contiki-ng/contiki-ng/wiki/

$ sudo apt-get udpate
$ sudo apt install build-essential doxygen git curl wireshark python3-
serial srecord rlwrap
# put yes for wireshark popup
$ sudo apt install autoconf automake libxmu-dev

# To install GCC for ARM controller
$ wget -c https://launchpad.net/gcc-arm-embedded/5.0/5-2015-q4-
major/+download/gcc-arm-none-eabi-5_2-2015q4-20151219-linux.tar.bz2
# extract it in home in a folder, it should contain "arm-none-eabi",
```

```
"bin", etc ...
# set in the Path Environment /home/youhome/.bashrc
$ export PATH=$PATH:/home/yourhome/gcc-arm-none-eabi-5_2-2015q4-20151219-
linux/bin
# or in ~./bashrc
$ nano ~/.bashrc
# and paste the same

# now the library
$ sudo apt install gcc-msp430

# java and ant installation
$ sudo apt install default-jdk ant

# for java! Only if you need it, switch between versions
$ sudo apt-get install openjdk-8-jdk
$ sudo update-alternatives --config java # runtime
$ sudo update-alternatives --config javac # compiler

# set the java path
$ export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/bin/java
# and put as before in .bashrc

# CoAP Client
$ sudo apt-get install -y npm && sudo apt-get clean && sudo npm install
coap-cli -g && sudo ln -s /usr/bin/nodejs /usr/bin/node

# MQTT Clients
$ sudo apt-get install -y mosquitto mosquitto-clients

# configuration over
# Now we can finally install contiki ng
$ git clone https://github.com/contiki-ng/contiki-ng.git
$ cd contiki-ng
$ git submodule update --init --recursive
```

2. Try the simulator

Now I'm using the simulator from installation 1.B

```
$ cd contiki-ng/tools/cooja
# or contiki folder in the available VM from their website
```

```
$ ant run
```

- If there is an error, type

```
$ git submodule update --init
```

and run again (from installation 1.B you should not have this error).

- If the error is

```
Buildfile: build.xml does not exist!
Build failed
```

Run

```
~/contiki-ng/tools/cooja$ ./gradlew run
```

Now we can create a new simulation `File -> Create new simulation`.
Select a new `Mote` type, e.g. Sky, and set the name and the file to use (the `hello_world.c` is fine). Compile it and start the simulation selecting the number of nodes and run the simulation.

- If you have error in compilation (`../../Makefile.include:194: *** GCC 4.7 or later is required for MSP430.. Stop.`):

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo install gcc-msp430

or follow https://stackoverflow.com/a/33244652
```

OR (working)

A compiler for MSP430 is needed to use the MSPSim emulator in Cooja. It is possible to install a GCC compiler for MSP430 from the Ubuntu package repository (`sudo apt install gcc-msp430`) but that version is too old to support the full memory of newer versions of MSP430.

The preferred version of MSP430 gcc is 4.7.2. To install it on `/usr/local` if you're using 64-bit Linux, run:

```
$ wget -nv http://simonduq.github.io/resources/mspgcc-4.7.2-
compiled.tar.bz2 && \
```

```
sudo tar xjf mspgcc*.tar.bz2 -C /tmp/ && \
sudo cp -f -r /tmp/msp430/* /usr/local/ && \
sudo rm -rf /tmp/msp430 mspgcc*.tar.bz2
```

3. D’Hondt’s RPL Framework

Credits: https://en.mukiraz.com/2022/06/dhondts-rpl-framework/

Studies on the Simulation of Attacks on RPL Protocol Attacks

In an academic report by D’Hondt et al. (2015) 1, they were able to simulate Flooding Attacks, Version Number Increase Attacks, and Decreased Rank Attacks on the RPL protocol using the Cooja IoT simulator.

This work and the arrangements for the attack simulations are in https://github.com/dhondta/rpl-attacks repository created by D’Hondt et al. (2015).