Documentation projet étude du BLE

Stage Master 1

Gidon Rémi

Avril/Juin 2020

Contents

1 Web interface

The interface is quite simple, it consists of controls to start/stop scans at the top left, a map of located devices, logs from Mirage execution in back-end and last but no least the list of found devices as well as connections. Each row contain informations who can later be used to launch attacks from the CLI using Mirage.

The map scales to the farthest device found, other distances are scaled relatively to it. In the picture below we can see the map is using the blue circle having a radius of 5.62 meters as it's scale. The nearest device, yellow one, only at 79 cm is much smaller and close to our point, which is the centered black spot.

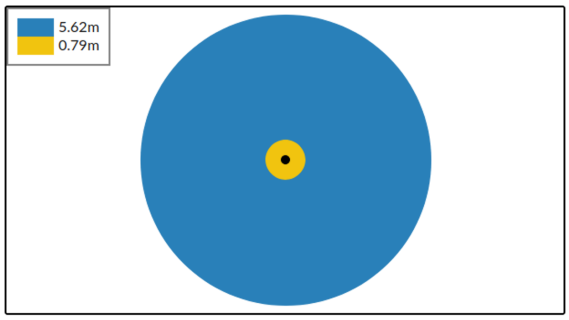

Figure 1: Map of located devices

2 Locate

Module `ble_locate` at `src/modules/ble_locate.py`, locate nearby BLE devices and sniff connections.

2.1 Parameters

Parameter	default value	possible value	description
ENVIRONMENT_FACTOR	2	0 to 4	losses due to environment, 0 is space, 4 is confined
INTERFACE	microbit0	microbitX	micro:bit board to use
TIME	5		Scan duration
DEVICE_CALLBACK	None	`print`	callback to use for periodic updates during scan
CONNECTION_CALLBACK	None	`print`	callback to use for periodic updates during scan
WINDOW	20		size of RSSI samples on which sum is calculated to obtain actual RSSI
SCAN_TYPE	all	devices, connections	scan realised, micro:bit can only sniff wether devices or connections at a time

Use mirage CLI entrypoint to issue commands:

```
libs/mirage/mirage_launcher ble_locate
```

3 Man In The Middle

4 Hijack

As explained in the corresponding documentation[1], this integrated Mirage module named `ble_hijack` allows to:

> *sniff an established connection, synchronize to it and jam the packet emitted by the slave. As a consequence, if the master reachs its timeout value, it disconnects from the slave device and the attacker is able to communicate with the slave device instead of him..*

Also, the module is based on `ble_sniff` to identify target connection and only hijacks the connection, as stated in the documentation:

> *this module needs ble_sniff, and cannot be used alone. Indeed, when the connection is hijacked, the module terminates its execution, allowing to run another module, such as ble_master or ble_discover.*

The module can wait for new connections to be created in order to sniff connection parameters required to follow the connection (sniffing mode new connections[2]) or try to synchronize with an existing connection by recovering connection parameters over time (sniffing mode existing connections[3]).

4.1 Parameters

An interesting parameter is that you can provied a `pcap` file to capture the attack and analyze it later using *Wireshark*. In order to be able to capture the traffic pass `PCAP_FILE=/path/to/file.pcap` where `/path/to/file.pcap` is an absolute path to a non-existent file.
For all possible parameters, refer to the module documentation[4].

Use mirage CLI entrypoint to issue commands:

```
# Hijack then capture connection with a shell as master
libs/mirage/mirage_launcher "ble_hijack|ble_master" HIJACKING_MODE=existingConnections
# equivalent to launching ble_hijack directly
libs/mirage/mirage_launcher "ble_sniff|ble_master" INTERFACE=microbit0 SNIFFING_MODE=exis
```

4.2 Troubleshooting

can take time to catch up connection bc bbc jumps but also conn, possible para not integrated in mirage to monitor multiple data chan

[1] https://homepages.laas.fr/rcayre/mirage-documentation/blemodules.html#id87
[2] https://homepages.laas.fr/rcayre/mirage-documentation/blemodules.html#hijacking-a-new-connection
[3] https://homepages.laas.fr/rcayre/mirage-documentation/blemodules.html#hijacking-an-existing-connection
[4] https://homepages.laas.fr/rcayre/mirage-documentation/blemodules.html#id95

5 Tests

The project provide a test network mocking a slave and BLE master based on Mirage `ble_slave` and `ble_master` modules. Those modules are customised using scenarios `MockMaster` and `MockSlave` found in `poc/src/scenarios`. The scenarios add CLI parameters to modify their behaviors besides those used by their modules.

5.1 MockSlave

Parameter	default value	possible value	description
INTERFACE	hci0	hciX	hci device to use
SCENARIO	MockSlave		scenario to use
NAME	Test Slave		local name emitted in the advertisement
PAIRING	yes		enable pairing
TXPOWER	-55		signal strength measured 1 meter away from device

5.2 MockMaster

Parameter	default value	possible value	description
INTERFACE	hci1	hciX	hci device to use
SCENARIO	MockMaster		scenario to use
NAME	Test Slave		local name used to iddentify target slave
PAIRING	yes		enable pairing
TARGET	scan for slave	BD address	slave BD address, scan devices if empty
REQUESTS	10		number of requests issued to slave device during the connection
INTERVAL	3		time in seconds between each request to slave

Use mirage CLI entrypoint to issue commands:

```
# start slave
libs/mirage/mirage_launcher ble_slave
# start master
libs/mirage/mirage_launcher ble_master
```

Use `--debug` switch to activate debug mode and see exception traces.
Be aware that mirage will stay in foreground while executing thus blocking the CLI until it finishes, use `&` parameter to start a task in background, even if it's recommended to start each mirage task in it's own shell for output readability.