



# Security in IoT Ecosystem

## Module 5

Smart Bulb Pentest example



# Table of Contents

- 1. Attack Surface Mappin for Smart Bulb**
  1. Radio communications (BLE)
  2. Tasks
- 2. OWASP IoT Top 10**

# First Step

- Attack Surface Mappin
  - Finding as much **information** as possible about the **device**.
  - Focus on the following categories
    1. Embedded device.
    2. Firmware, software, and applications.
    3. **Radio communications.**



# HEKKE AC85-265V

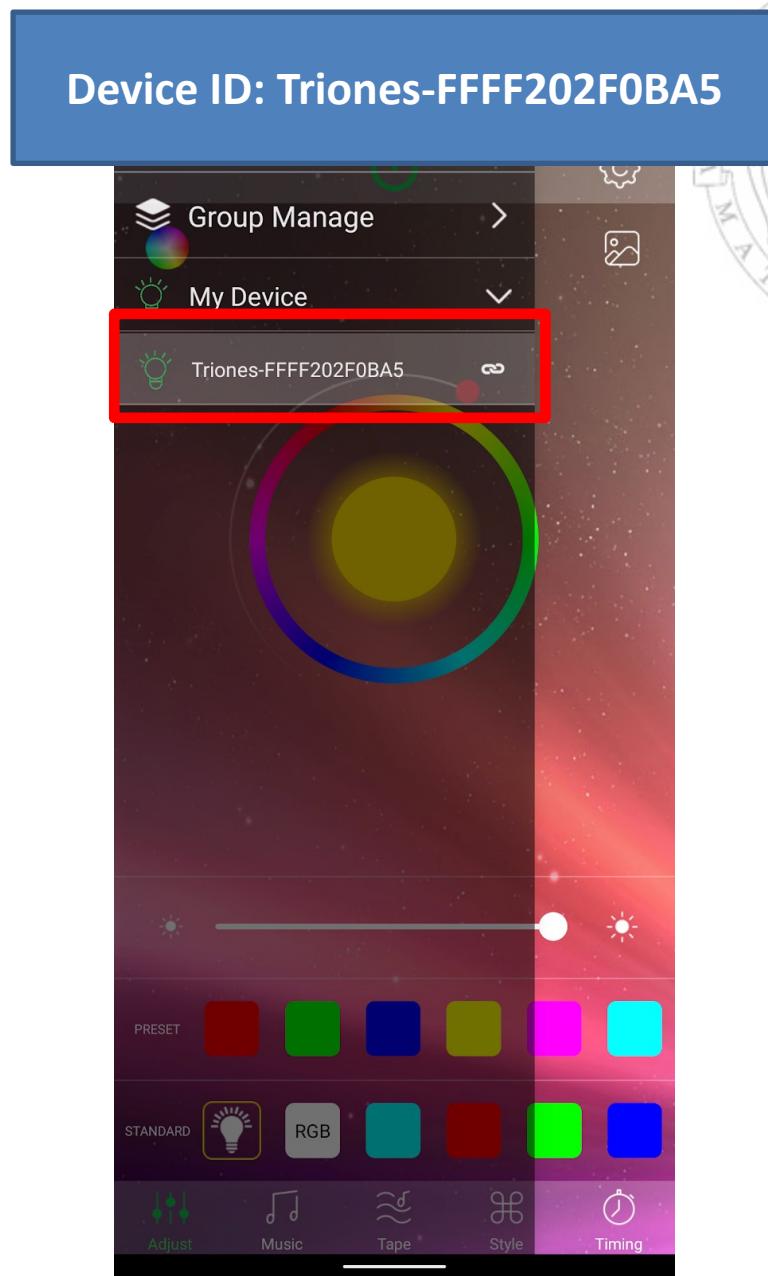
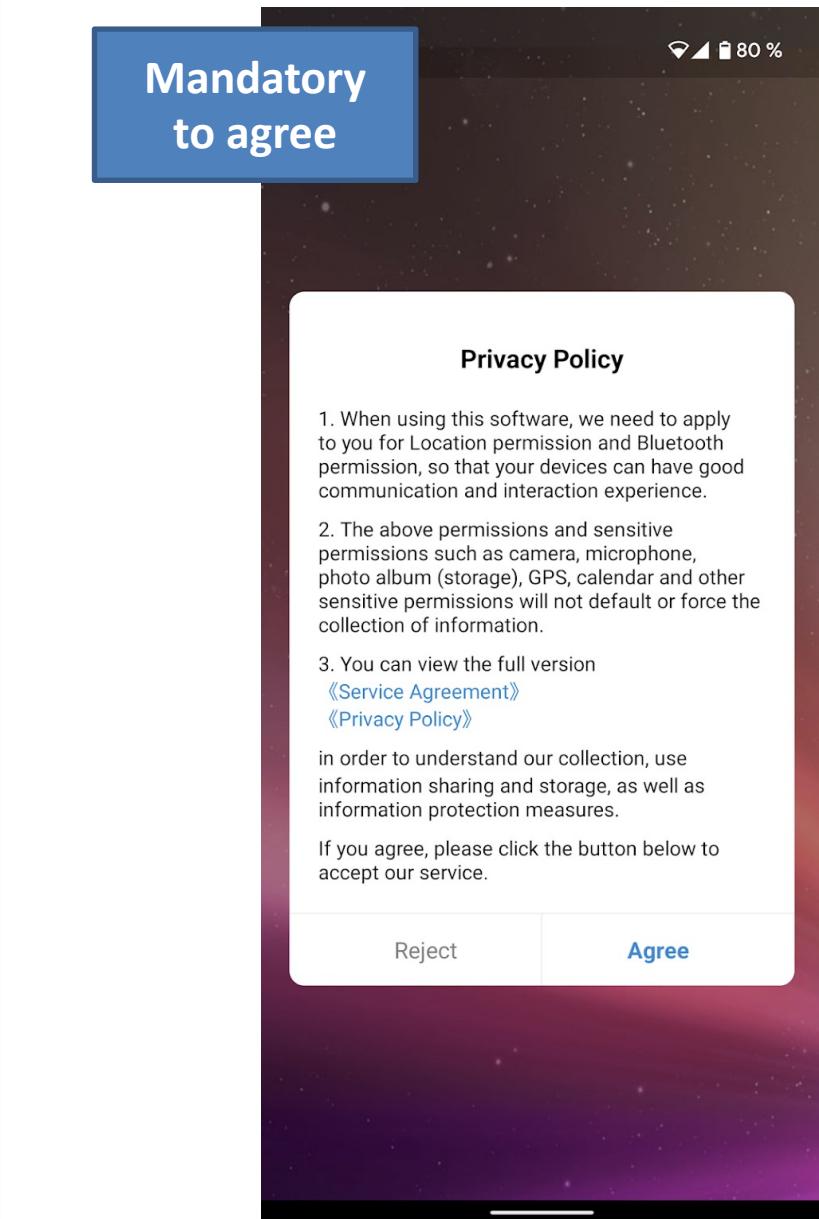
- General specs:
  - Color Changing LEDs.
  - RGB+ Cool White
  - 12 fixed color: Red, Green, Blue, Orange, etc.
  - Dual Memory: help you preset what you set last time.
  - **Bluetooth connectivity**
  - **Android APP:** [HappyLighting](#)

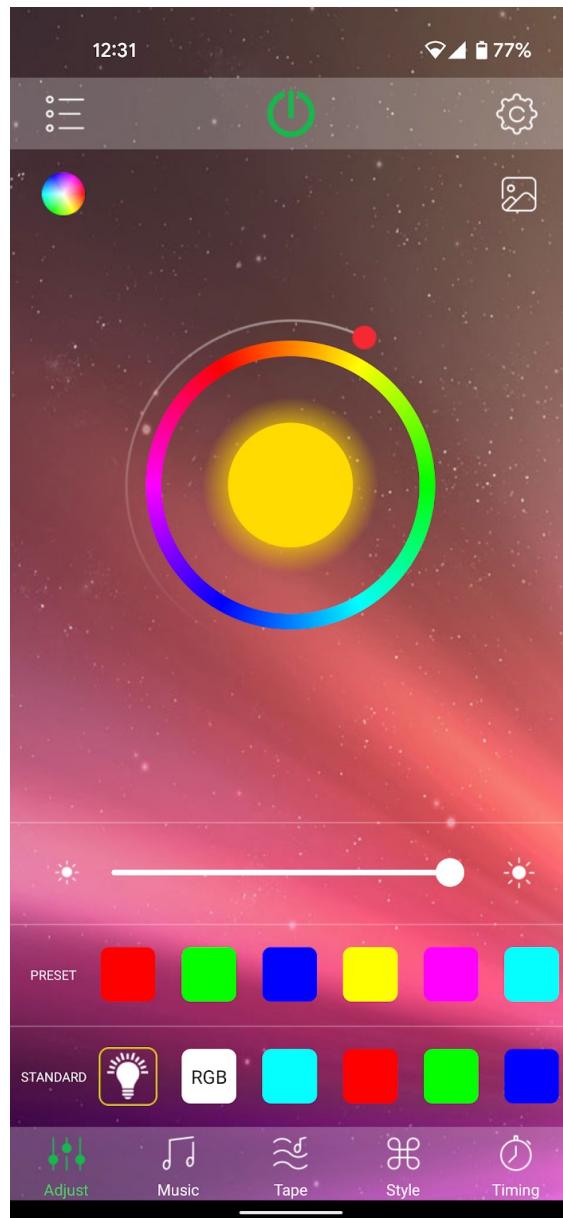


# HappyLighting

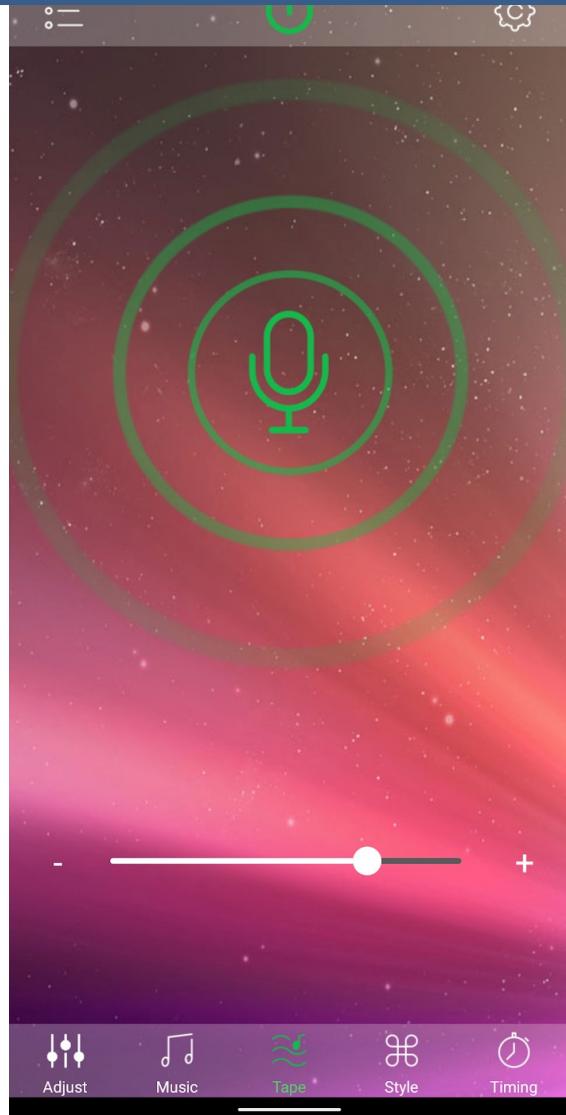


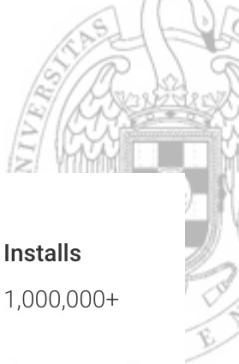
- HappyLighting is a Bluetooth lamp control software:
  1. You can control the HappyLighting Bluetooth lights for color matching.
  2. You can control the timing of HappyLighting Bluetooth lights.
  3. Can control the HappyLighting Bluetooth lamp to set the lighting mode.
  4. You can change the color of the light according to the music.
- If your Bluetooth lamp doesn't start with "**Triones, BRGlight, Dream, Light**" in the Bluetooth list, don't download this app, because this app only controls these Bluetooth lamp devices.





Record audio and access to files





# HappyLighting

qh-tek

Showing permissions for all versions of this app

This app has access to:

Location

- approximate location (network-based)
- precise location (GPS and network-based)

Photos/Media/Files

- read the contents of your USB storage
- modify or delete the contents of your USB storage

Updates to HappyLighting may automatically add additional capabilities within each group. [Learn more](#)

X

## ADDITIONAL INFORMATION

Updated	Size	Installs
December 16, 2021	35M	1,000,000+
Current Version	Requires Android	Content Rating
1.6.1.7	4.3 and up	Everyone
		<a href="#">Learn more</a>
Permissions	Report	Offered By
<a href="#">View details</a>	<a href="#">Flag as inappropriate</a>	qh-tek
Developer		
service@qh-tek.com		
<a href="#">Privacy Policy</a>		

Cancel

## REVIEWS

[Review policy and info](#)

3.5



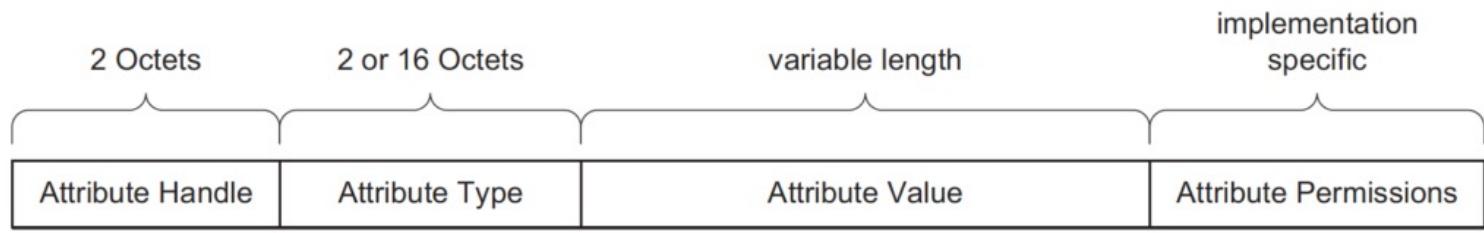
8,672 total





# Basics of Bluetooth Low Energy (BLE)

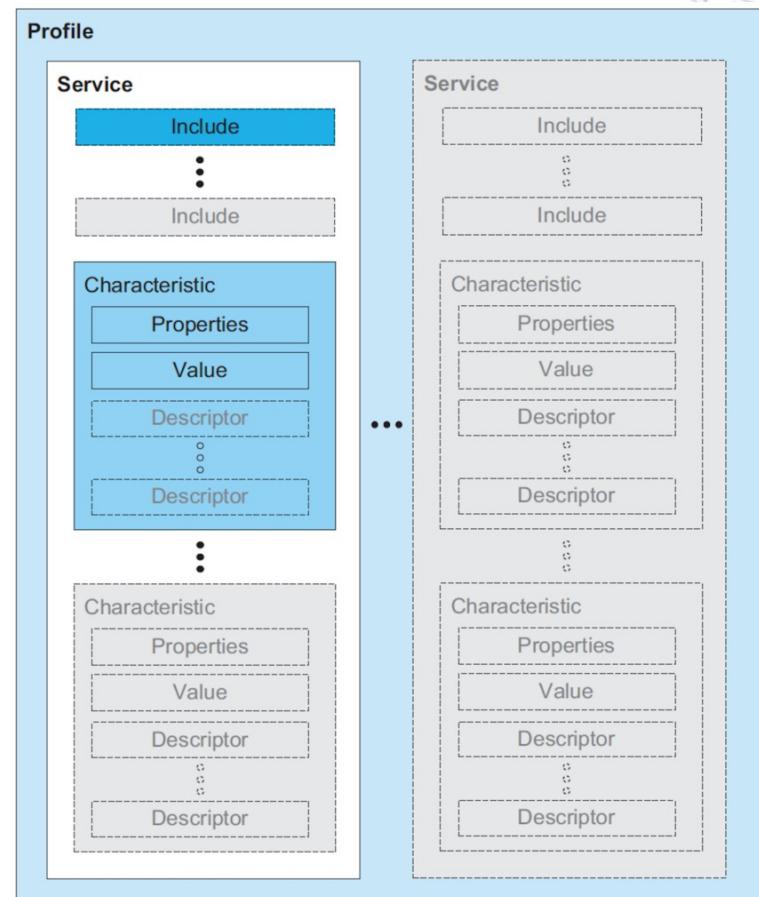
- Attribute Protocol (**ATT**): defines how a server exposes its data to a client and how this data is structured:
  - **Server**: This is the device that exposes the data
  - **Client**: This is the device that interfaces with the server with the purpose of reading the server's exposed data and/or controlling the server's behaviour
- The data that the server exposes is structured as **attributes**:
  - **Attribute type (Universally Unique Identifier or UUID)**: a 16/128b
  - **Attribute Handle**: a 16-bit value that the server assigns to each of its attributes
  - **Attribute Permissions**: Permissions determine whether an attribute can be **read** or **written** to, whether it can be **notified** or **indicated**, and what **security levels** are required for each of these operations.





# Basics of Bluetooth Low Energy (BLE)

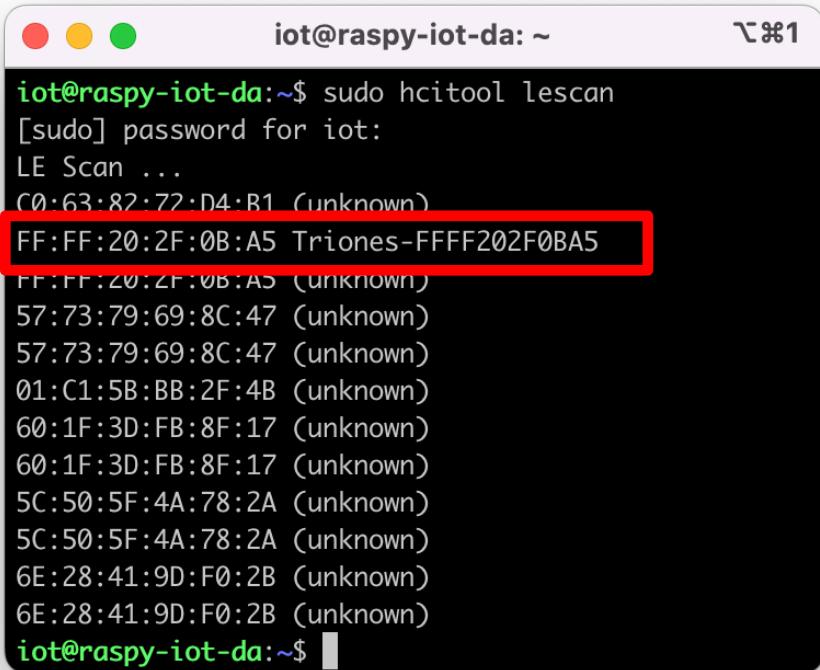
- The Generic Attribute Profile (GATT) is used after a connection has been established between the two devices:
  - A **service** is a grouping of one or more attributes, some of which are characteristics
  - A **characteristic** is always part of a service and it represents a piece of information/data that a server wants to expose to a client.
  - **Profiles** are much broader in definition than services. They are concerned with defining the behavior of both the client and server when it comes to services, characteristics, and even connections and security requirements.





# Bluetooth analysis: hcitool

- [hcitool](#) is used to configure Bluetooth connections and send some special command to Bluetooth devices:  
scan: Inquire remote devices. For each discovered device, device name are printed.



A terminal window titled "iot@raspy-iot-da: ~" showing the output of the "hcitool lescan" command. The command is run with sudo privileges. The output lists various Bluetooth devices discovered during an LE scan. One entry, "FF:FF:20:2F:0B:A5 Triones-FFFF202F0BA5", is highlighted with a red rectangle.

```
iot@raspy-iot-da:~$ sudo hcitool lescan
[sudo] password for iot:
LE Scan ...
C0:63:82:72:D4:B1 (unknown)
FF:FF:20:2F:0B:A5 Triones-FFFF202F0BA5
FF:FF:20:2F:0B:A5 (unknown)
57:73:79:69:8C:47 (unknown)
57:73:79:69:8C:47 (unknown)
01:C1:5B:BB:2F:4B (unknown)
60:1F:3D:FB:8F:17 (unknown)
60:1F:3D:FB:8F:17 (unknown)
5C:50:5F:4A:78:2A (unknown)
5C:50:5F:4A:78:2A (unknown)
6E:28:41:9D:F0:2B (unknown)
6E:28:41:9D:F0:2B (unknown)
iot@raspy-iot-da:~$
```



# Bluetooth analysis: gatttool

- gatttool is tool that can be used to manipulate these attributes with a Bluetooth Low Energy device:
  - b, --device=MAC Specify remote Bluetooth address
  - I, ---interactive Use interactive mode

```
iot@raspy-iot-da:~$ sudo gatttool -I -b FF:FF:20:2F:0B:A5
[FF:FF:20:2F:0B:A5][LE]> help
help                                     Show this help
exit                                      Exit interactive mode
quit                                     Exit interactive mode
connect [address [address type]]        Connect to a remote device
disconnect                                Disconnect from a remote device
primary [UUID]                            Primary Service Discovery
included [start hnd |end hnd]           Find Included Services
characteristics [start hnd [end hnd [UUID]]] Characteristics Discovery
cchar-desc [start hnd] [end hnd]          Characteristics Descriptor Discovery
char-read-hnd <handle>                  Characteristics Value/Descriptor Read by handle
char-read-uuid <UUID> [start hnd] [end hnd] Characteristics Value/Descriptor Read by UUID
char-write-req <handle> <new value>    Characteristic Value Write (Write Request)
cchar-write-cmd <handle> <new value>   Characteristic value write (No response)
sec-level [low | medium | high]         Set security level. Default: low
mtu <value>                             Exchange MTU for GATT/ATT
[FF:FF:20:2F:0B:A5][LE]> []
```



# Bluetooth analysis: gatttool

```
iot@raspy-iot-da:~$ sudo gatttool -I -b FF:FF:20:2F:0B:A5
[FF:FF:20:2F:0B:A5][LE]> connect
Attempting to connect to FF:FF:20:2F:0B:A5
Connection successful
[FF:FF:20:2F:0B:A5][LE]> primary
attr handle: 0x0001, end grp handle: 0x0004 uuid: 0000ffd0-0000-1000-8000-00805f9b34fb
attr handle: 0x0005, end grp handle: 0x0007 uuid: 0000ffd5-0000-1000-8000-00805f9b34fb
[FF:FF:20:2F:0B:A5][LE]> characteristics 0x0001 0x0004
handle: 0x0002, char properties: 0x12, char value handle: 0x0003, uuid: 0000ffd4-0000-1000-8000-00805f9b34fb
[FF:FF:20:2F:0B:A5][LE]> characteristics 0x0005 0x0007
handle: 0x0006, char properties: 0x04, char value handle: 0x0007, uuid: 0000ffd9-0000-1000-8000-00805f9b34fb
[FF:FF:20:2F:0B:A5][LE]> █
```

- Lets try a *Reply Attack*:
  - Reply attack: a form of network attack in which valid data transmission is maliciously or fraudulently repeated.
  - This is carried out either by an adversary who intercepts the data and re-transmits it.



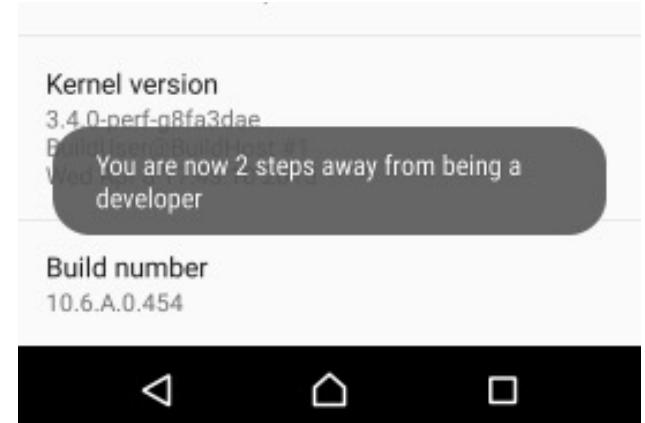
# Reply attack

- We need valid packets to send to the device
- We can obtain these packets by:
  - Sniffing the air transmission
    - Ubertooth One: open source 2.4 GHz wireless development platform suitable for Bluetooth experimentation.
  - **Record a valid transmission between a controlled android device and the Light Bulb**



# Bluetooth analysis

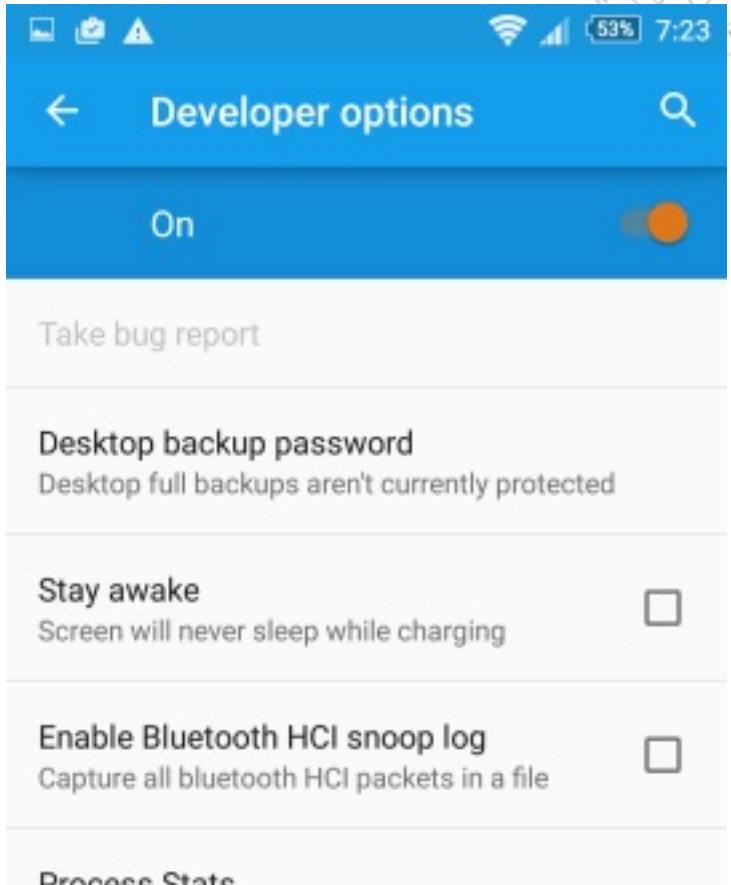
- Many Android phones are capable to log the full Bluetooth communication.
- Starting with Android 4.4 the log option is available on all phones in the Android developer settings.
- Enable Developer Settings
  - Usually the developer settings are invisible.
  - You can simply enable them by tapping 7 times on the build number in Android settings.





# Bluetooth analysis

- Create Log:
  - Open the developer menu in Android settings.
  - You see a checkbox labelled "*Enable Bluetooth HCI Snoop Log*"
  - Start the log before you power on the car and stop the log before you send the file.
  - The log file is called *btsnoop\_hci.log* and is usually stored in the root of the USB/SD storage.
- If you want to peek into the file you can use **WireShark**.



The logo for Wireshark, featuring the word "WIRESHARK" in a large, bold, sans-serif font. A stylized shark fin graphic is positioned above the letter "I".

# WIRESHARK



- **Wireshark** is the world's foremost and widely-used network protocol analyzer: It lets you see what's happening on your network.
- **Wireshark** has a rich feature set which includes the following:
  - Deep inspection of hundreds of protocols
  - Live capture and offline analysis
  - Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
  - Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
  - The most powerful display filters in the industry
  - Rich VoIP analysis
  - Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, **Bluetooth**, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
  - **Decryption support** for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
  - Etc.



# WireShark: *btsnoop\_hci.log*

Too much information!!!

What do we want to look for?

- Triones device

Protocol	Length	Info
HCI_CMD	4	Sent Reset
HCI_EVT	7	Rcvd Command Complete (Reset)
HCI_CMD	4	Sent Read Buffer Size
HCI_EVT	14	Rcvd Command Complete (Read Buffer Size)
HCI_CMD	11	Sent Host Buffer Size
HCI_EVT	7	Rcvd Command Complete (Host Buffer Size)
HCI_CMD	4	Sent Read Local Version Information
HCI_EVT	15	Rcvd Command Complete (Read Local Version Information)
HCI_CMD	4	Sent Read BD ADDR
HCI_EVT	13	Rcvd Command Complete (Read BD ADDR)
	4	Sent Read Local Supported Commands
	71	Rcvd Command Complete (Read Local Supported Commands)
	5	Sent Read Local Extended Features
	17	Rcvd Command Complete (Read Local Extended Features)
	5	Sent Write Simple Pairing Mode
	7	Rcvd Command Complete (Write Simple Pairing Mode)
	6	Sent Write LE Host Supported
	7	Rcvd Command Complete (Write LE Host Supported)
	5	Sent Read Local Extended Features
	17	Rcvd Command Complete (Read Local Extended Features)
	5	Sent Read Local Extended Features
	17	Rcvd Command Complete (Read Local Extended Features)
	4	Sent Vendor Command 0x0153 (opcode 0xFD53)
	21	Rcvd Command Complete (Vendor Command 0x0153 [opcode 0xFD53])
	5	Sent Write Secure Connections Host Support

```
> Frame 1: 4 bytes on wire (32 bits), 4 bytes captured (32 bits)
> Bluetooth
> Bluetooth HCI H4
  < Bluetooth HCI_Command_Packet
    0000  01 03 0c 00  ...

```

Packets: 350 · Displayed: 350 (100.0%) · Profile: Default



# WireShark: *BLE devices*

- Wireless -> Bluetooth Devices

The screenshot shows a window titled "Bluetooth Devices" with a table of discovered devices. The columns are: BD\_ADDR, OUI, Name, LMP Version, LMP Subversion, Manufacturer, HCI Version, HCI Revision, and Is Local Adapter. A red box highlights the row for a Raspberry Pi 4. The table data is as follows:

BD_ADDR	OUI	Name	LMP Version	LMP Subversion	Manufacturer	HCI Version	HCI Revision	Is Local Adapter
00:00:00:00:00:00	00:00:00							
1a:6c:16:c6:ad:ec								
59:27:1f:67:17:80								
61:cd:83:0f:47:51								
66:9b:8c:f6:d7:4d								
b8:27:eb:8b:14:34	Raspberry	Raspberry Pi 4	5.0	24857	Cypress Semiconductor	5.0	339	true
ff:ff:20:2f:0b:a5		Triones-FFFF202F0BA5	4.0	16643	Telink Semiconductor Co. Ltd			

At the bottom left, there are buttons for "All Interfaces" and "Show information steps". At the bottom right, there is a "Close" button.



# WireShark filters

- Useful filters ([manual](#)):

Display Filter Reference: Bluetooth

**Protocol field name:** bluetooth

**Versions:** 2.0.0 to 3.6.3

[Back to Display Filter Reference](#)

FIELD NAME	DESCRIPTION	TYPE	VERSIONS
bluetooth.addr	Source or Destination	Ethernet or other MAC address	2.0.0 to 3.6.3
bluetooth.addr_str	Source or Destination	Character string	2.2.0 to 3.6.3
bluetooth.dst	Destination	Ethernet or other MAC address	2.0.0 to 3.6.3
bluetooth.dst_str	Destination	Character string	2.2.0 to 3.6.3
bluetooth.src	Source	Ethernet or other MAC address	2.0.0 to 3.6.3
bluetooth.src_str	Source	Character string	2.2.0 to 3.6.3

- More filters filters ([Bluetooth Attribute Protocol](#)):

- `btatt.handle == <num>` # Interactions with handle
- `btatt.opcode == 0x12` # Write request



# WireShark: *btsnoop\_hci.log*

WireShark screenshot showing a capture of *btsnoop\_hci\_onoff.log*. The packet list shows several HCI\_EVT frames from a controller to a host, with the last one being an HCI\_EVT (LE Connection Complete). The details pane shows a selected LE Advertising Report event. The selected packet's bytes are shown in the bottom hex, ASCII, and EUI-64 panes.

**Selected Packet Details:**

- No. 127 Time 6.810841 Source controller Destination host Protocol HCI\_EVT Length 40 Info Rcvd LE Meta (LE Advertising Report)
- No. 128 Time 6.811409 Source controller Destination host Protocol HCI\_EVT Length 15 Info Rcvd LE Meta (LE Advertising Report)
- No. 134 Time 6.962229 Source controller Destination host Protocol HCI\_EVT Length 40 Info Rcvd LE Meta (LE Advertising Report)
- No. 135 Time 6.962937 Source controller Destination host Protocol HCI\_EVT Length 15 Info Rcvd LE Meta (LE Advertising Report)
- No. 144 Time 8.322518 Source controller Destination host Protocol HCI\_EVT Length 22 Info Rcvd LE Meta (LE Connection Complete)

**Selected Sub-Event: LE Advertising Report (0x02) Details:**

- Event Type: Connectable Undirected Advertising (0x00)
- Peer Address Type: Public Device Address (0x00)
- BD\_ADDR: ff:ff:20:2f:0b:a5 (ff:ff:20:2f:0b:a5)

**Selected Advertising Data Details:**

- Flags
  - Length: 2
  - Type: Flags (0x01)
  - 000. .... = Reserved: 0x0
  - .... 0.... = Simultaneous LE and BR/EDR to Same Device Capable (Host): false (0x0)
  - .... 0... = Simultaneous LE and BR/EDR to Same Device Capable (Controller): false (0x0)
  - .... .1.. = BR/EDR Not Supported: true (0x1)
  - .... ..1. = LE General Discoverable Mode: true (0x1)
  - .... ..0 = LE Limited Discoverable Mode: false (0x0)
- Device Name: Triones-FFFF202F0BA5
  - Length: 21
  - Type: Device Name (0x09)
  - Device Name: Triones-FFFF202F0BA5

**Selected Bytes:**

0000	04	3e	25	02	01	00	00	a5	0b	2f	20	ff	ff	19	02	01	>%..... / .....
0010	06	15	09	54	72	69	6f	6e	65	73	2d	46	46	46	46	32	...Triones-FFFF2
0020	30	32	46	30	42	41	35	d1								02F0BA5	

Device Name (btcommon.eir\_ad.entry.device\_name), 20 bytes

Packets: 350 · Displayed: 5 (1.4%) · Profile: Default

`bthci_evt.bd_addr == ff:ff:20:2f:0b:a5`



# Filter by address

btsnoop\_hci\_onoff.log

No.	Time	Source	Destination	Protocol	Length	Info
158	8.645507	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Group Type Request
160	8.789569	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	23	Rcvd Read By Group Type Response
161	8.789872	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Group Type Request
163	8.887003	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	14	Rcvd Error Response - Attribute
164	8.887402	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Type Request, GATT
165	8.984519	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	14	Rcvd Error Response - Attribute
166	8.984933	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Type Request, GATT
169	9.047076	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	18	Rcvd Read By Type Response, Att
170	9.047377	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Type Request, GATT
171	9.060259	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	14	Rcvd Error Response - Attribute
172	9.062484	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	14	Sent Find Information Request, I
174	9.077004	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	15	Rcvd Find Information Response,
175	9.077464	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Type Request, GATT
176	9.092032	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	Raspber_8b:14:34 (Raspberry Pi 4)	ATT	14	Rcvd Error Response - Attribute
177	9.092423	Raspber_8b:14:34 (Raspberry Pi 4)	ff:ff:20:2f:0b:a5 (Triones-FFFF202F0BA5)	ATT	16	Sent Read By Type Request, GATT Charac

```
> Frame 158: 16 bytes on wire (128 bits), 16 bytes captured (128 bits)
> Bluetooth
> Bluetooth HCI H4
> Bluetooth HCI ACL Packet
> Bluetooth L2CAP Protocol
> Bluetooth Attribute Protocol
```

0000 02 40 00 0b 00 07 00 04 00 10 01 00 ff ff 00 28 @..... .(

btsnoop\_hci\_onoff.log

Packets: 350 · Displayed: 45 (12.9%)

Profile: Default

`bluetooth.addr == ff:ff:20:2f:0b:a5`

# Tasks



1. Download btsnoop\_hci.log file for your group available in the SEC web page.
  - Multiple on/off sequences
  - Multiple color sets
2. Identify the MAC address on the Light Bulb
  - Wireless -> Bluetooth Devices
  - Identify *Triones* device
3. Filter the packets to see only “Write Request”
  - btatt.opcode == 0x12
  - In which handle is the Android (Raspi4) writing?
4. Identify the payload for reply attack



# Table of Contents

- 1. Attack Surface Mappin for Smart Bulb**
  1. Radio communications (BLE)
  2. Tasks
- 2. OWASP IoT Top 10**



# OWASP Recommendations

- If we had followed the OWASP recommendations, would we have avoided these security holes?
- Task: identify in the OWASP Top 10 recommendations not followed and detected in this Pentest.



# OWASP IoT Top 10

