IoT Security – Autumn 2024

Lab 9: Sniffing Bluetooth with the Raspberry Pi

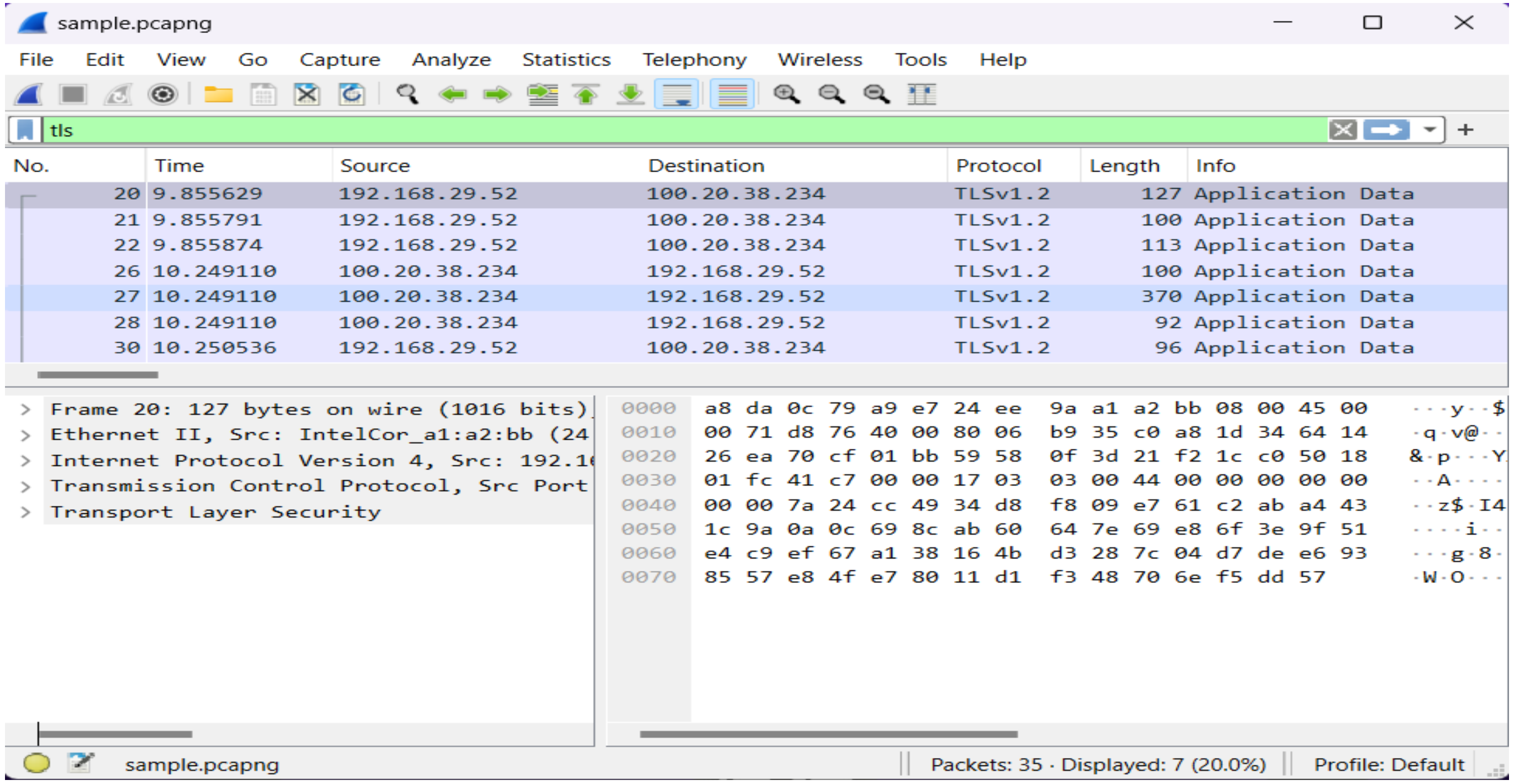
Manh Bui
School of Electrical and Data Engineering
Email: DucManh.Bui@uts.edu.au

Objectives

- Getting familiar with varying levels of security on different common devices that use Bluetooth/BLE (Bluetooth Low Energy)
- Configuring a Raspberry Pi to detect and display information about the Bluetooth devices that are in its range
- The flows of implementation:
  - Part 1: Configuring the Raspberry Pi as a Bluetooth Sniffer
  - Part 2: Using the CLI to Detect and Display Information about Bluetooth Device
  - Part 3: Using the CLI to Pair with a Sniffed Device

What is sniffing?

- **Sniffing** is the process of collecting, monitoring and capturing the packets in **computer networks**.



tls

No.	Time	Source	Destination	Protocol	Length	Info
20	9.855629	192.168.29.52	100.20.38.234	TLSv1.2	127	Application Data
21	9.855791	192.168.29.52	100.20.38.234	TLSv1.2	100	Application Data
22	9.855874	192.168.29.52	100.20.38.234	TLSv1.2	113	Application Data
26	10.249110	100.20.38.234	192.168.29.52	TLSv1.2	100	Application Data
27	10.249110	100.20.38.234	192.168.29.52	TLSv1.2	370	Application Data
28	10.249110	100.20.38.234	192.168.29.52	TLSv1.2	92	Application Data
30	10.250536	192.168.29.52	100.20.38.234	TLSv1.2	96	Application Data

```
> Frame 20: 127 bytes on wire (1016 bits)
> Ethernet II, Src: IntelCor_a1:a2:bb (24
> Internet Protocol Version 4, Src: 192.16
> Transmission Control Protocol, Src Port
> Transport Layer Security
```

```
0000  a8 da 0c 79 a9 e7 24 ee  9a a1 a2 bb 08 00 45 00   ···y··$
0010  00 71 d8 76 40 00 80 06  b9 35 c0 a8 1d 34 64 14   ·q·v@··
0020  26 ea 70 cf 01 bb 59 58  0f 3d 21 f2 1c c0 50 18   &·p···Y
0030  01 fc 41 c7 00 00 17 03  03 00 44 00 00 00 00 00   ··A····
0040  00 00 7a 24 cc 49 34 d8  f8 09 e7 61 c2 ab a4 43   ··z$·I4
0050  1c 9a 0a 0c 69 8c ab 60  64 7e 69 e8 6f 3e 9f 51   ····i··
0060  e4 c9 ef 67 a1 38 16 4b  d3 28 7c 04 d7 de e6 93   ···g·8·
0070  85 57 e8 4f e7 80 11 d1  f3 48 70 6e f5 dd 57      ·W·O···
```

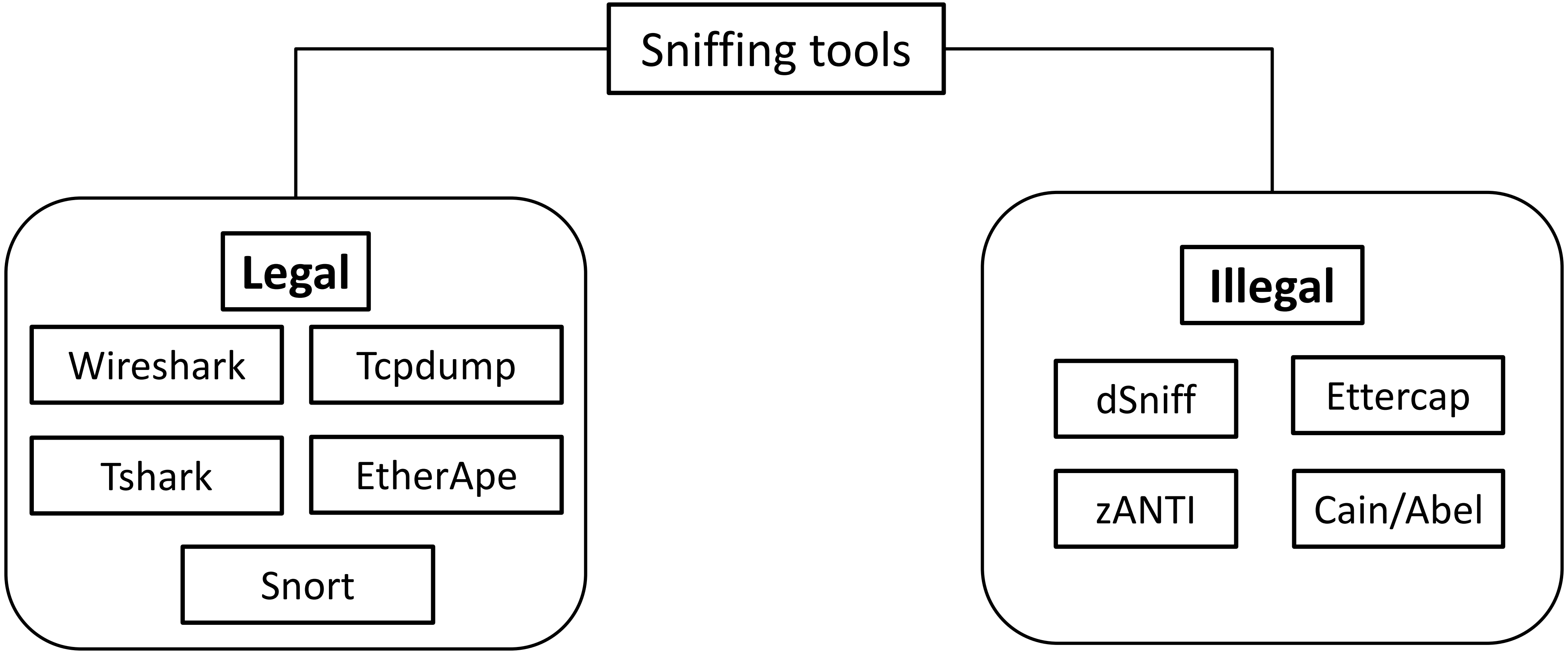
sample.pcapng | Packets: 35 · Displayed: 7 (20.0%) | Profile: Default

Figure 1: An example of traffic file captured by Wireshark

Sniffing Tools

Sniffing can be used for **legal** or **illegal** purposes:

- Legal: Monitoring network traffic for industrial purposes
- Illegal: Performing cyberattacks to **steal user's information**



Sniffing tools

Legal

Wireshark

Tcpdump

Tshark

EtherApe

Snort

Illegal

dSniff

Ettercap

zANTI

Cain/Abel

Sniffing Attack

- Attackers use sniffing tools to monitor the traffic between client and server (router, switch,…)
- Attackers can decode the packet to steal user’s information or reply to the server with modified messages
- In terms of cyberattack, Sniffing is a branch of Man-in-the-middle (MitM) attack
- Types of sniffing attacks: ARP Spoofing, DHCP Spoofing, Active/Passive Sniffing, SSL Sniffing,…

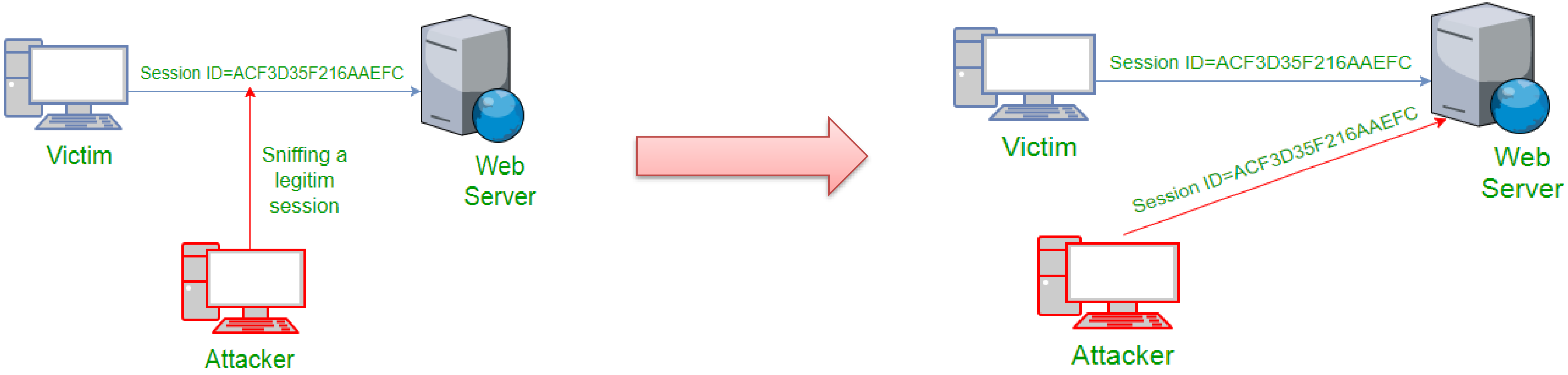
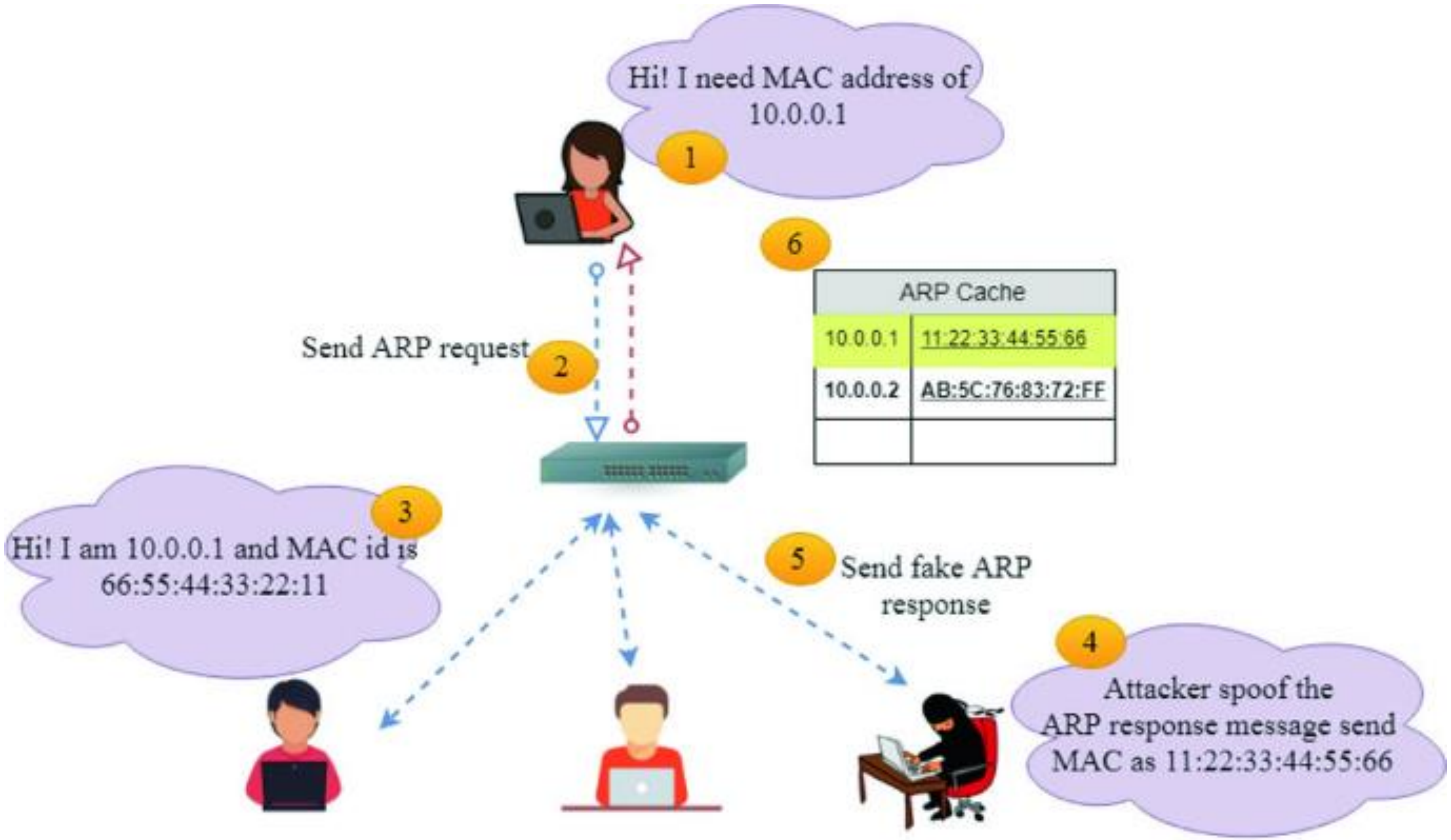

Victim
Session ID=ACF3D35F216AAEFC
Web Server
Sniffing a legitim session
Attacker
Victim
Session ID=ACF3D35F216AAEFC
Web Server
Session ID=ACF3D35F216AAEFC
Attacker

Figure 2: Diagram of sniffing attack (Ref)

Example

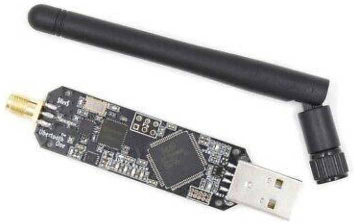

1
Hi! I need MAC address of 10.0.0.1
2
Send ARP request
3
Hi! I am 10.0.0.1 and MAC id is 66:55:44:33:22:11
4
Attacker spoof the ARP response message send MAC as 11:22:33:44:55:66
5
Send fake ARP response
6

ARP Cache	
10.0.0.1	11:22:33:44:55:66
10.0.0.2	AB:5C:76:83:72:FF

Figure 3: Example of ARP Spoofing attack (Ref)

What is Bluetooth Sniffing?

- Bluetooth sniffing is the process of capturing and monitoring the communication **between Bluetooth devices**
- It is used to analyse the connection between devices, thereby enhancing the security and privacy of communication
- Require specific hardware **which can listen to Bluetooth signal** (Ubertooth, dongle Bluetooth) and software **which can collect and analyse Bluetooth communication** (hcitool,…)



Ubertooth

Bluetooth tracker

BLE USB Sniffer

Bluetooth sniffing attacks

- There are many types of Bluetooth sniffing attacks: MitM, Bluetooth Snarfing, BlueBugging, …
- The motivations of attackers include:
  - **Stealing information** from user’s devices (history calls, phone books, messages,…)
  - **Taking full control** of the user’s devices

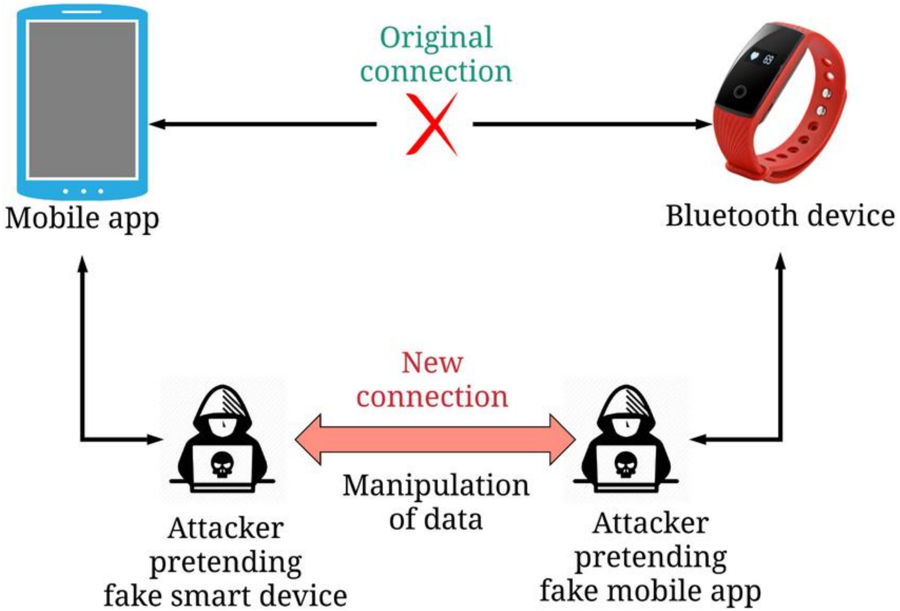
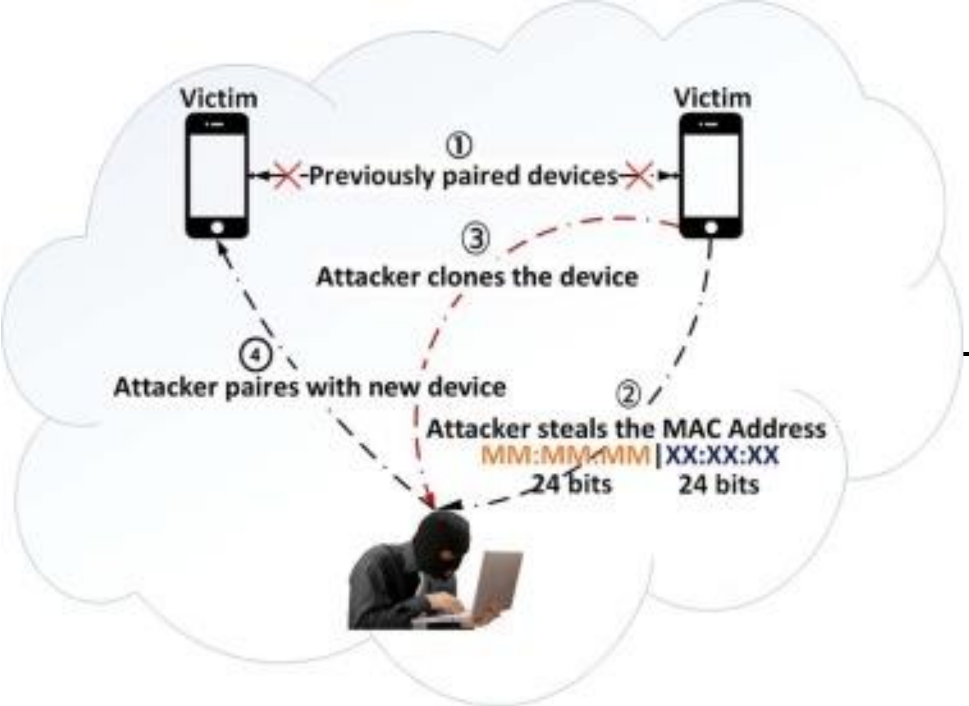
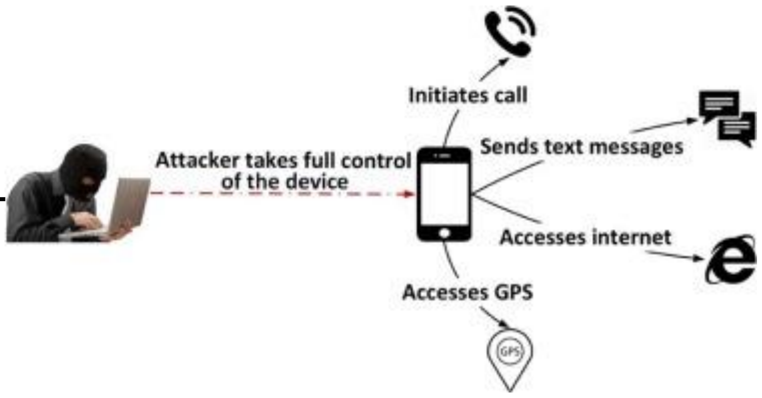


Figure 5: Example of MitM attack on Bluetooth devices

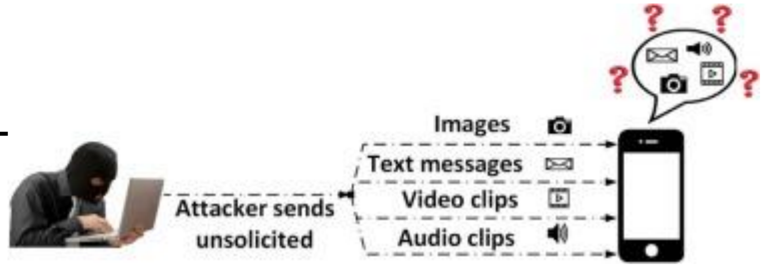
Example


Victim
Victim
①
Previously paired devices
③
Attacker clones the device
④
Attacker paires with new device
②
Attacker steals the MAC Address
MM:MM:MM | XX:XX:XX
24 bits
24 bits

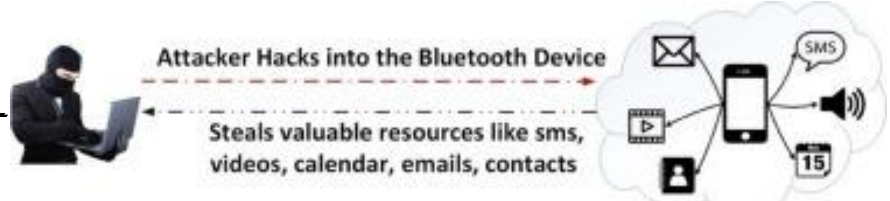
MitM attack to pair with Victim’s device (Reference)



BlueBugging attack which takes full control of device



BlueJacking attack which sends spammed messages to victim’s device


Attacker Hacks into the Bluetooth Device
Steals valuable resources like sms, videos, calendar, emails, contacts
SMS
15

BlueSnarfing attack which steals information from a device

Bluetooth Sniffing in IoT scenario

- Many IoT sensors utilise Bluetooth for their connectivity
- The attacker can exploit the vulnerability of Bluetooth technology by the **sniffer** to **track** and **decode** broadcasting information
- Attackers can steal users' information by hosting cyberattacks on IoT systems: smart homes, smart farms,...

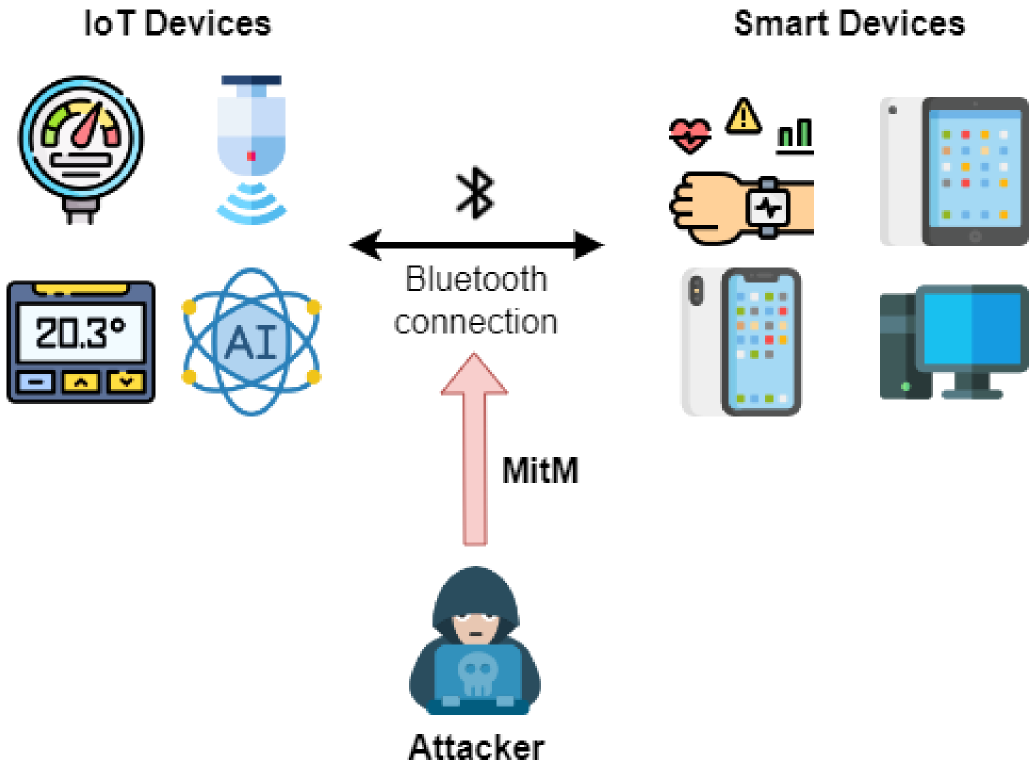
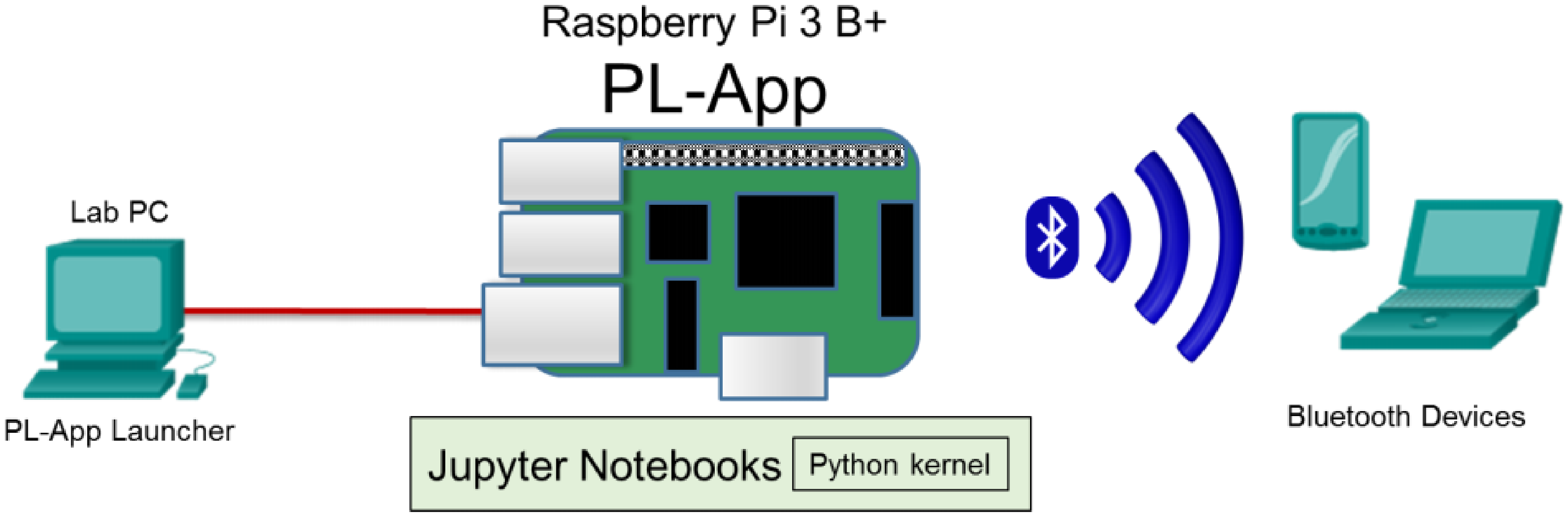


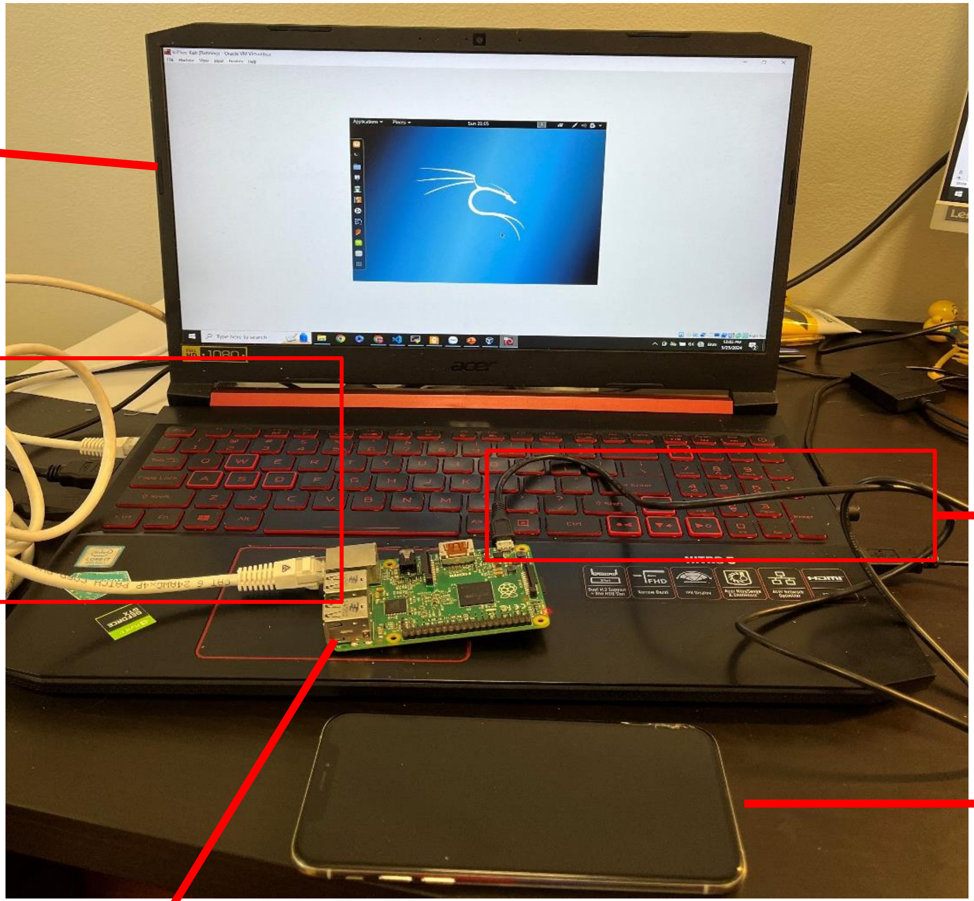
Figure 6: MitM attack on smart home

Required Resources

- Raspberry Pi 3 Model B (with Micro SD card attached)
- PC/Laptop with IoTSec Kali VM
- Network connectivity between PC and Raspberry Pi
- Bluetooth devices (PC, Smartphone,…)



Environment Set-up



PC with Kali VM

Network connectivity via Ethernet cable

Charging via microUSB cable

Bluetooth Device

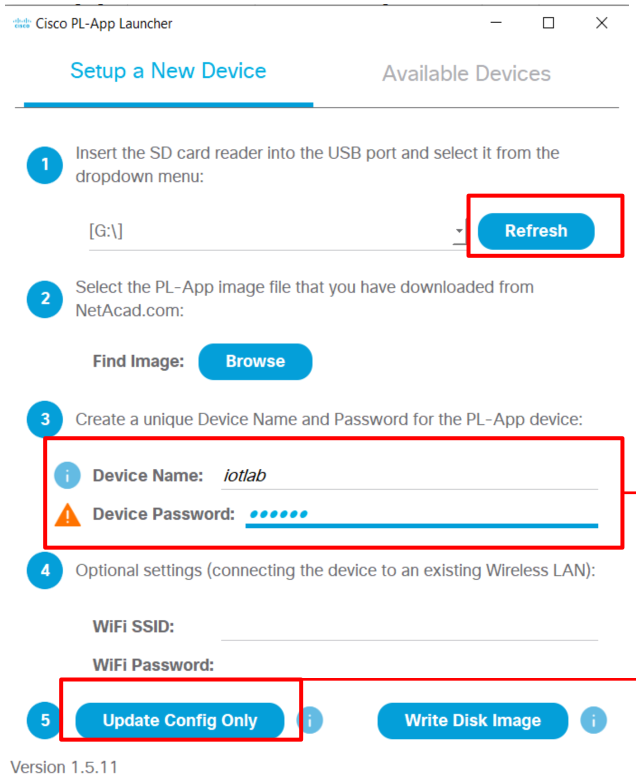
Raspberry Pi 3 Model B

Environment Set-up

1. Update config to PL-App Launcher:
   - Insert your SD card to your PC
   - Open PL-App Launcher
   - Create device name and password
   - Click **Update Config Only**

2. Log in to Kali-linux
   - Username: root
   - Password: toor

3. Open terminal and start DHCP server on Kali VM by running command:
lab_support_files/scripts/start_dhcp.sh



1 Insert the SD card reader into the USB port and select it from the dropdown menu:

[G:\] Refresh

2 Select the PL-App image file that you have downloaded from NetAcad.com:

Find Image: Browse

3 Create a unique Device Name and Password for the PL-App device:

Device Name: *iotlab*

Device Password: ••••••

4 Optional settings (connecting the device to an existing Wireless LAN):

WiFi SSID:

WiFi Password:

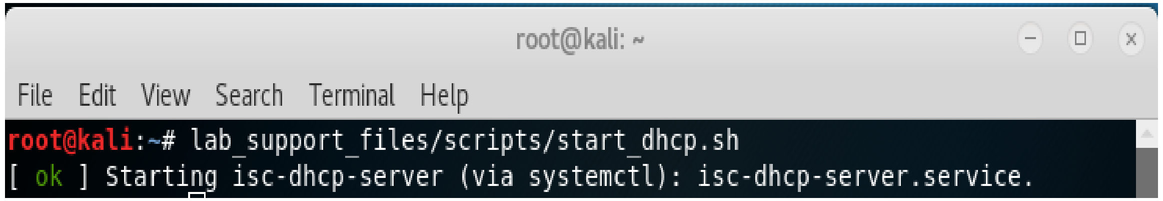
5 Update Config Only Write Disk Image

Version 1.5.11

Click here to update the SD card

Update device name and password

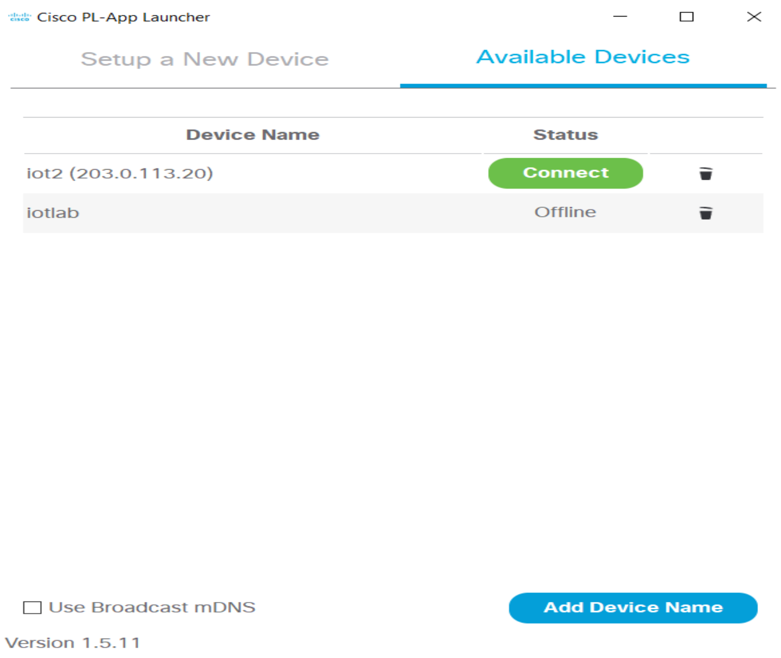
Click here to update all the config



```
root@kali:~# lab_support_files/scripts/start_dhcp.sh
[ ok ] Starting isc-dhcp-server (via systemctl): isc-dhcp-server.service.
```

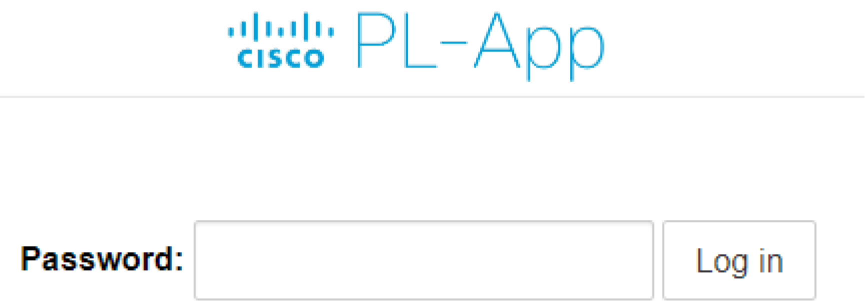
Environment Set-up

4. Set up network connection
   - Eject the SDcard and put it into the Raspberry Pi
   - Charge the Rasp via microUSB cable and connect Ethernet cable from PC to Rasp
   - Comeback to PL-App Launcher and you will see the Rasp is connected



Device Name	Status
iot2 (203.0.113.20)	Connect
iotlab	Offline

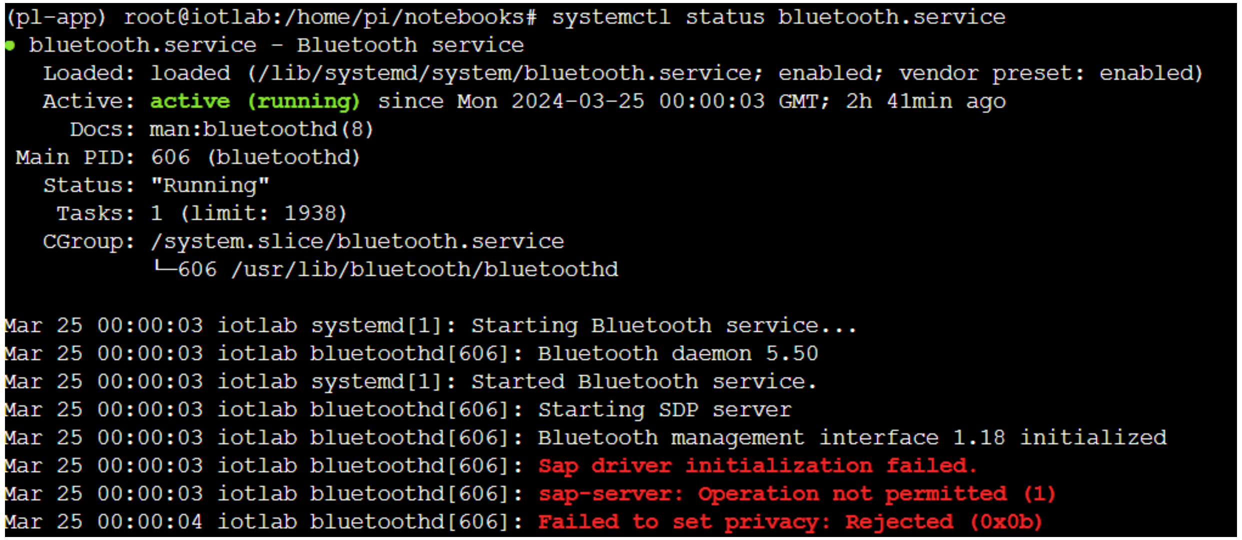
5. Access Raspberry Pi PL-App
   - Click on connect and insert your password
   - You will see all material of the PL-App and the set-up is finished



Part 1: Configuring Raspberry Pi as a Bluetooth Sniffer

1. Open the terminal in PL-App and use command: **rfkill list** (if Bluetooth is blocked, use **rfkill unblock bluetooth)**

2. Use command: **systemctl status bluetooth.service**

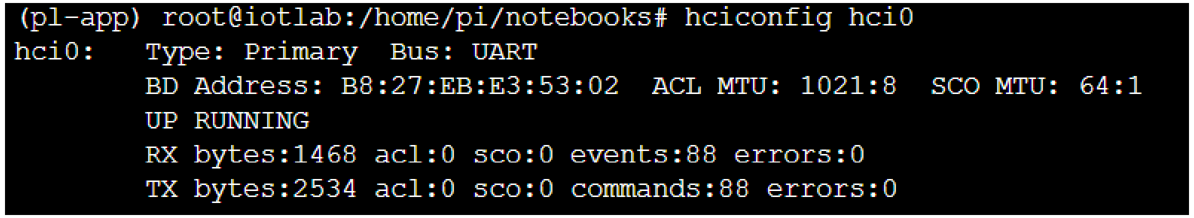


```
(pl-app) root@iotlab:/home/pi/notebooks# systemctl status bluetooth.service
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-03-25 00:00:03 GMT; 2h 41min ago
     Docs: man:bluetoothd(8)
 Main PID: 606 (bluetoothd)
   Status: "Running"
    Tasks: 1 (limit: 1938)
   CGroup: /system.slice/bluetooth.service
           └─606 /usr/lib/bluetooth/bluetoothd

Mar 25 00:00:03 iotlab systemd[1]: Starting Bluetooth service...
Mar 25 00:00:03 iotlab bluetoothd[606]: Bluetooth daemon 5.50
Mar 25 00:00:03 iotlab systemd[1]: Started Bluetooth service.
Mar 25 00:00:03 iotlab bluetoothd[606]: Starting SDP server
Mar 25 00:00:03 iotlab bluetoothd[606]: Bluetooth management interface 1.18 initialized
Mar 25 00:00:03 iotlab bluetoothd[606]: Sap driver initialization failed.
Mar 25 00:00:03 iotlab bluetoothd[606]: sap-server: Operation not permitted (1)
Mar 25 00:00:04 iotlab bluetoothd[606]: Failed to set privacy: Rejected (0x0b)
```

If it shows inactive (dead), use command: **systemctl start bluetooth.service**

3. Use command: **hciconfig hci0** (to display the status of Bluetooth interface)

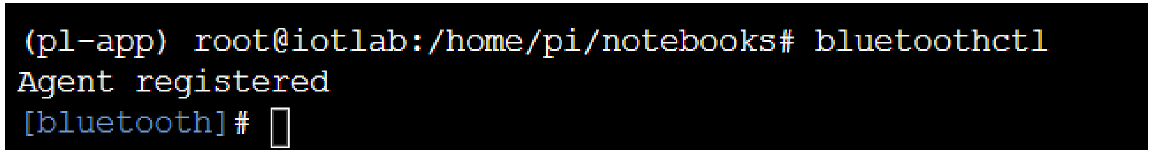


```
(pl-app) root@iotlab:/home/pi/notebooks# hciconfig hci0
hci0:   Type: Primary  Bus: UART
        BD Address: B8:27:EB:E3:53:02  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING
        RX bytes:1468 acl:0 sco:0 events:88 errors:0
        TX bytes:2534 acl:0 sco:0 commands:88 errors:0
```

If the hci0 status is down, use command: **hciconfig hci0 up**

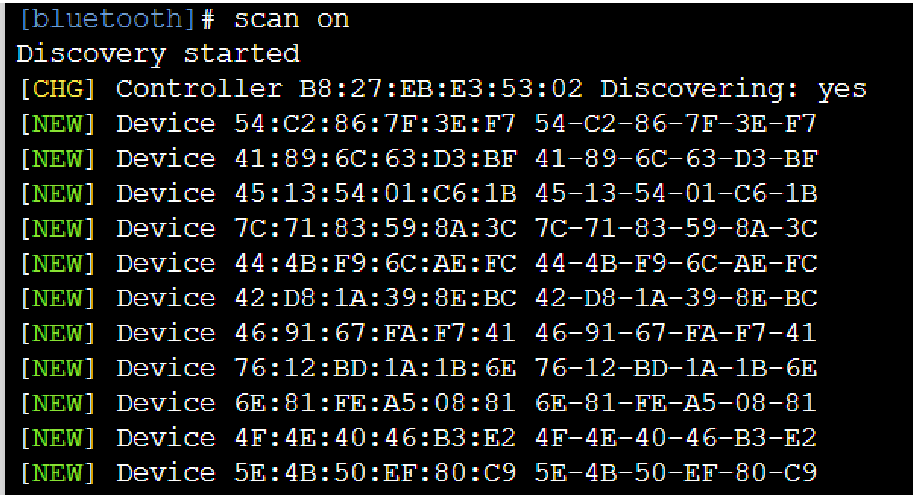
Part 2: Detect and Display Information about Bluetooth Devices

4. Use command: **bluetoothctl** to launch the Bluetooth tool



```
(pl-app) root@iotlab:/home/pi/notebooks# bluetoothctl
Agent registered
[bluetooth]# 
```

5. Use command: **scan on** to start scanning on nearby devices



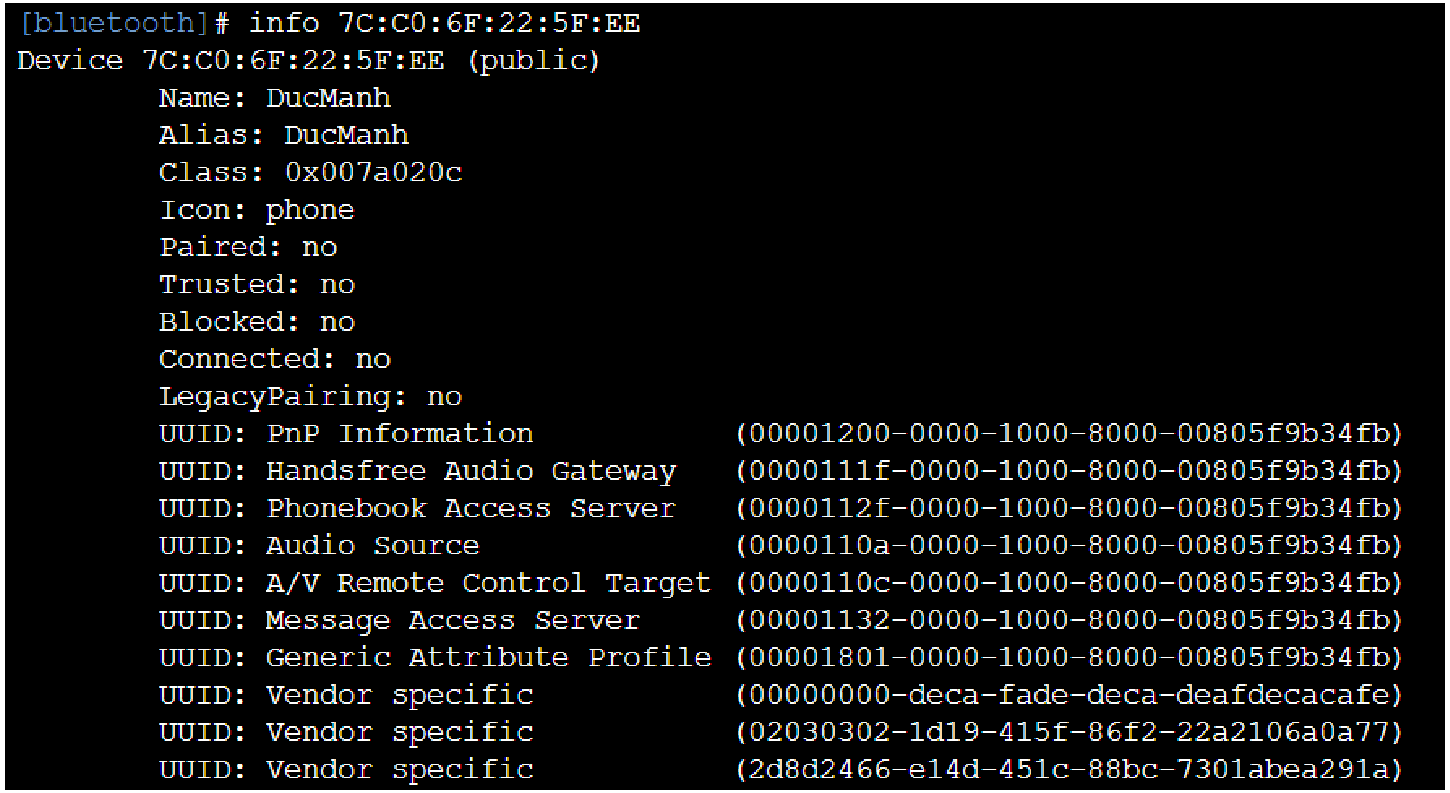
```
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:E3:53:02 Discovering: yes
[NEW] Device 54:C2:86:7F:3E:F7 54-C2-86-7F-3E-F7
[NEW] Device 41:89:6C:63:D3:BF 41-89-6C-63-D3-BF
[NEW] Device 45:13:54:01:C6:1B 45-13-54-01-C6-1B
[NEW] Device 7C:71:83:59:8A:3C 7C-71-83-59-8A-3C
[NEW] Device 44:4B:F9:6C:AE:FC 44-4B-F9-6C-AE-FC
[NEW] Device 42:D8:1A:39:8E:BC 42-D8-1A-39-8E-BC
[NEW] Device 46:91:67:FA:F7:41 46-91-67-FA-F7-41
[NEW] Device 76:12:BD:1A:1B:6E 76-12-BD-1A-1B-6E
[NEW] Device 6E:81:FE:A5:08:81 6E-81-FE-A5-08-81
[NEW] Device 4F:4E:40:46:B3:E2 4F-4E-40-46-B3-E2
[NEW] Device 5E:4B:50:EF:80:C9 5E-4B-50-EF-80-C9
```

Question: Take notes of the MAC address of some discovered Bluetooth devices.

6. After a few second of scanning, turn off by command: **scan off**

Part 2: Detect and Display Information about Bluetooth Devices

7. Use command **info [MAC address]** to show detailed information of the devices



```
[bluetooth]# info 7C:C0:6F:22:5F:EE
Device 7C:C0:6F:22:5F:EE (public)
        Name: DucManh
        Alias: DucManh
        Class: 0x007a020c
        Icon: phone
        Paired: no
        Trusted: no
        Blocked: no
        Connected: no
        LegacyPairing: no
        UUID: PnP Information           (00001200-0000-1000-8000-00805f9b34fb)
        UUID: Handsfree Audio Gateway   (0000111f-0000-1000-8000-00805f9b34fb)
        UUID: Phonebook Access Server   (0000112f-0000-1000-8000-00805f9b34fb)
        UUID: Audio Source              (0000110a-0000-1000-8000-00805f9b34fb)
        UUID: A/V Remote Control Target (0000110c-0000-1000-8000-00805f9b34fb)
        UUID: Message Access Server     (00001132-0000-1000-8000-00805f9b34fb)
        UUID: Generic Attribute Profile (00001801-0000-1000-8000-00805f9b34fb)
        UUID: Vendor specific           (00000000-deca-fade-deca-deafdecacafe)
        UUID: Vendor specific           (02030302-1d19-415f-86f2-22a2106a0a77)
        UUID: Vendor specific           (2d8d2466-e14d-451c-88bc-7301abea291a)
```

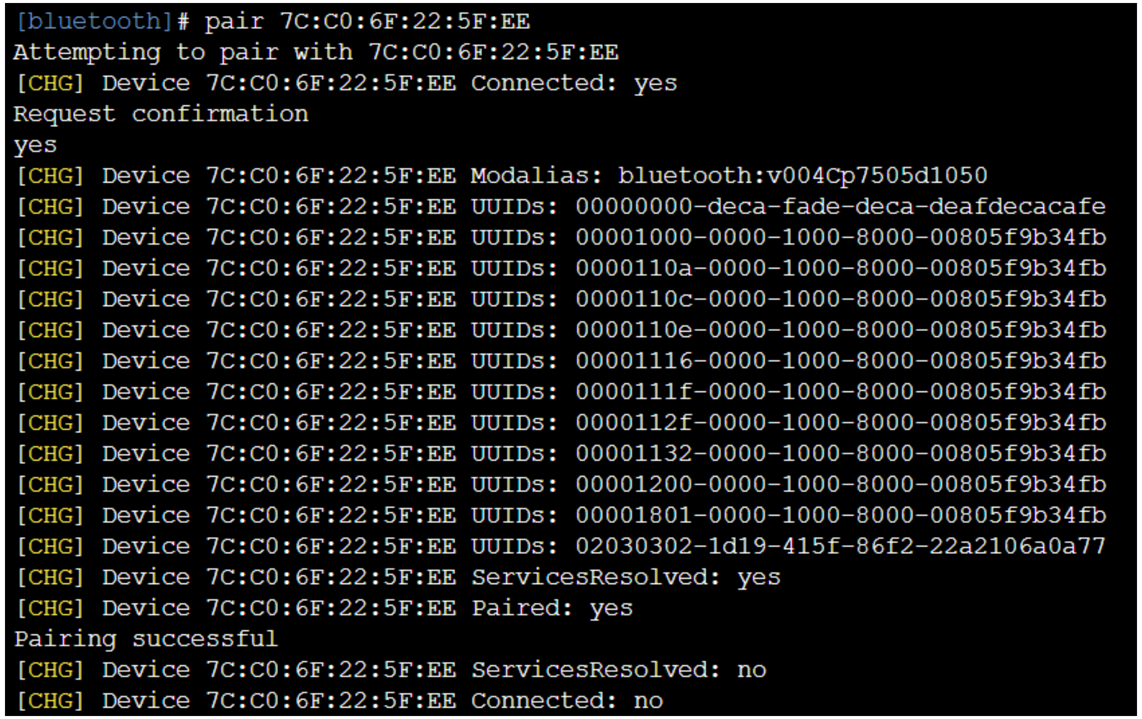
For example, this is the obtained information of a smart phone

Question: Take note of the information displayed and perform a web search for the meaning of some displayed features.

Part 3: Pairing Sniffed Device with Raspberry Pi

8. Use command: **paired-devices** to view device paired with the Pi

9. Use command: **pair [MAC address]** to pair the device
(it is noted that only the **insecure** Bluetooth device or high-secured device with **authorization** can be paired)



```
[bluetooth]# pair 7C:C0:6F:22:5F:EE
Attempting to pair with 7C:C0:6F:22:5F:EE
[CHG] Device 7C:C0:6F:22:5F:EE Connected: yes
Request confirmation
yes
[CHG] Device 7C:C0:6F:22:5F:EE Modalias: bluetooth:v004Cp7505d1050
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 00000000-deca-fade-deca-deafdecacafe
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 00001000-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 00001116-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 0000112f-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 00001132-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device 7C:C0:6F:22:5F:EE UUIDs: 02030302-1d19-415f-86f2-22a2106a0a77
[CHG] Device 7C:C0:6F:22:5F:EE ServicesResolved: yes
[CHG] Device 7C:C0:6F:22:5F:EE Paired: yes
Pairing successful
[CHG] Device 7C:C0:6F:22:5F:EE ServicesResolved: no
[CHG] Device 7C:C0:6F:22:5F:EE Connected: no
```

The output of successful pairing with authorization from smartphone

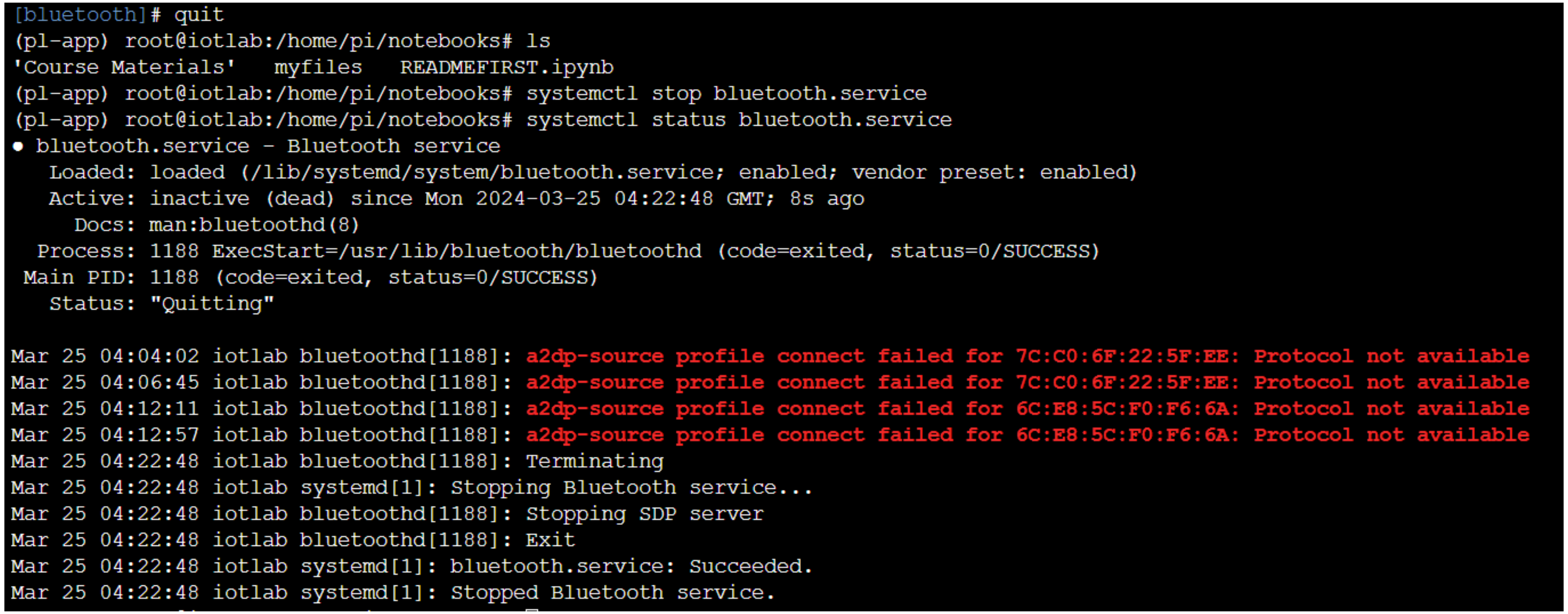
Part 3: Pairing Sniffed Device with Raspberry Pi

10. Try to connect to device via command: **connect [MAC address]**

Question: Are you able to connect (without permission of the owner) to Bluetooth devices?

Final step is to clean everything after the implementation:

- Use command: **quit** to exit from Bluetooth CLI
- Use command: systemctl stop bluetooth.service



```
[bluetooth]# quit
(pl-app) root@iotlab:/home/pi/notebooks# ls
'Course Materials'   myfiles   READMEFIRST.ipynb
(pl-app) root@iotlab:/home/pi/notebooks# systemctl stop bluetooth.service
(pl-app) root@iotlab:/home/pi/notebooks# systemctl status bluetooth.service
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Mon 2024-03-25 04:22:48 GMT; 8s ago
     Docs: man:bluetoothd(8)
  Process: 1188 ExecStart=/usr/lib/bluetooth/bluetoothd (code=exited, status=0/SUCCESS)
 Main PID: 1188 (code=exited, status=0/SUCCESS)
   Status: "Quitting"

Mar 25 04:04:02 iotlab bluetoothd[1188]: a2dp-source profile connect failed for 7C:C0:6F:22:5F:EE: Protocol not available
Mar 25 04:06:45 iotlab bluetoothd[1188]: a2dp-source profile connect failed for 7C:C0:6F:22:5F:EE: Protocol not available
Mar 25 04:12:11 iotlab bluetoothd[1188]: a2dp-source profile connect failed for 6C:E8:5C:F0:F6:6A: Protocol not available
Mar 25 04:12:57 iotlab bluetoothd[1188]: a2dp-source profile connect failed for 6C:E8:5C:F0:F6:6A: Protocol not available
Mar 25 04:22:48 iotlab bluetoothd[1188]: Terminating
Mar 25 04:22:48 iotlab systemd[1]: Stopping Bluetooth service...
Mar 25 04:22:48 iotlab bluetoothd[1188]: Stopping SDP server
Mar 25 04:22:48 iotlab bluetoothd[1188]: Exit
Mar 25 04:22:48 iotlab systemd[1]: bluetooth.service: Succeeded.
Mar 25 04:22:48 iotlab systemd[1]: Stopped Bluetooth service.
```

Reference

1. Definition of a sniffing attack

https://www.geeksforgeeks.org/what-is-sniffing-attack-in-system-hacking/

2. Pairing Raspberry Pi documentation

https://bluedot.readthedocs.io/en/latest/pairpiandroid.html

3. Tools used to config Raspberry Pi to a Sniffer

https://github.com/IanMercer/pi-sniffer

Thank
You
Vinaka
köszönöm
Kiitos
Maake
Asante
Shukria
Dhanyavadagalu
감사합니다
Dankscheen
спасибо
Kam Sah Hammida
اركش
Manana
Dankon
Dank Je
Mauruuru
Biyan
Diolch i Chi
Terima Kasih
Taiku
Matondo
Blagodaram
Ngiyabonga
Dziekuje
Chokrane
Tack
Grazie
Juspaxar
Arigato
Mochchakkeram
Gracias
நன்றி
Bedankt
Dakujem
धन्यवाद
Grazas
cảm ơn bạn
Khap
Paldies
Tingki
Gratias Tibi
Ua Tsaug Rau Koj
Di Ou Mèsi
Kia Ora
Kop Khun
Obrigado
Děkuji
Nuwun
Nirringrazzjak
ありがとう
Suksama
Rahmat
Matur
谢谢
хвала
Hvala
Welalin
Djiere Dieuf
Misaotra
Danke
Merci
Go Raibh Maith Agat
Tuke
Eskerrik Asko
Salamat
ขอขอบคุณคุณ
Najis