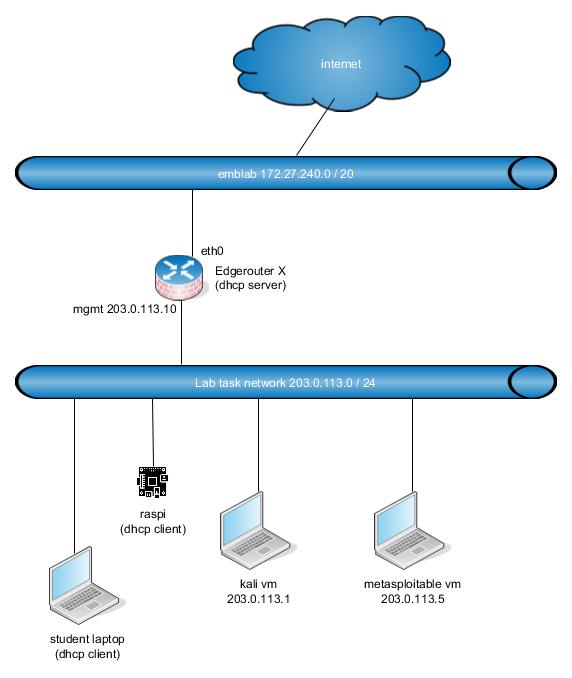
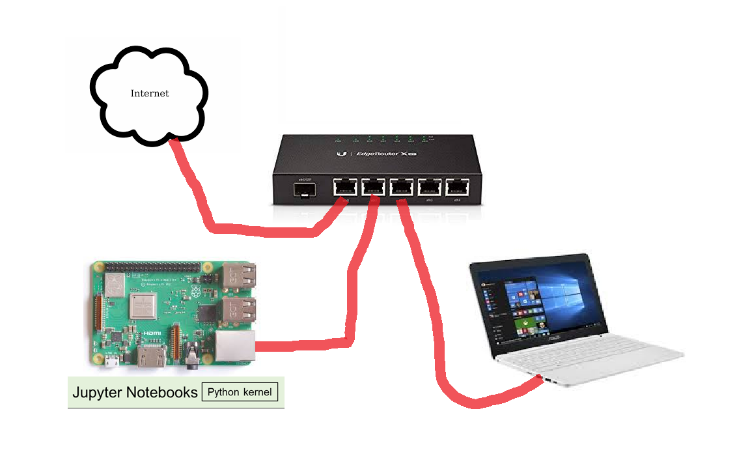
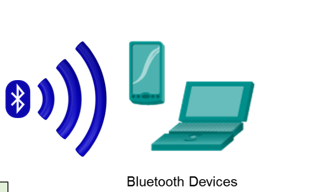
**In this lab (Part 3), please do not try to pair your Raspberry to anybody else’s devices without permission.**

**Lab - Sniffing Bluetooth with the Raspberry Pi**

Topology



1. Objectives

In this lab, students will become familiar with varying levels of security on different common devices that use Bluetooth/BLE (Bluetooth Low Energy). Students will configure a Raspberry Pi to detect and display information about the Bluetooth devices that are in its range. This is an awareness building lab that will help students understand the nature of wireless hacking processes and requirements and take on a “hacker mindset” to think like a “white hat” to help secure IoT Security devices.

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

1. Background/Scenario

Bluetooth offers several benefits and advantages. Bluetooth also has risks.

Bluetooth technology and associated devices are susceptible to general wireless networking threats, such as denial of service attacks and man in the middle (MITM) attacks. MITM attacks occur when the threat actors position themselves between the source and destination and intercept the communications. Some Bluetooth devices are designed to broadcast MAC, universal unique identifier (UUID) and service information at a predefined interval. Due to continuous advertisement, hackers can easily track the device and decode the broadcasting information using a sniffer. This lab requires that the target Bluetooth devices are locatable by the Raspberry Pi. These could be phones, laptops, personal fitness devices, and others.

**Note**: The labs in this course assume that the student has all of the necessary hardware to perform them. If you do not have the necessary hardware and wish to complete these labs, you may wish to purchase kits which contain all of the hardware for the challenge labs and additional hardware, which can be used to complete additional experiments beyond this course. Make sure to read the required resources for each challenge lab to understand what hardware is required.

1. **Required Resources**

* Raspberry Pi 3 Model B or later (with PL-App)
* 8GB Micro SD card (minimum required)
* PC with IoTSec Kali VM
* Network connectivity between PC and Raspberry Pi
* Bluetooth devices (PC, Smartphone, Smartband, Bluetooth LED Control…)

1. Configuring the Raspberry Pi as a Bluetooth Sniffer
   * 1. Set up the topology.
        1. Start the IoT Security lab topology with Kali VM and Raspberry Pi connected physically via an Ethernet cable.
        2. Log into Kali VM with the username **kali** and password **kali**.
        3. Determine the IP address of your Raspberry Pi as was done in previous labs, if necessary.
        4. Open a web browser in the Kali VM or your host PC and navigate to the IP address for your Raspberry Pi.
     2. Verify and activate the Bluetooth interface/services on the Raspberry Pi.
        1. Within the PL-App, open a terminal window and enter the **rfkill list** command to check if the Bluetooth interface is blocked by software:

(pl-app) root@pi:/home/pi/notebooks# **rfkill list**

0: phy0: Wireless LAN

Soft blocked: no

Hard blocked: no

1: hci0: Bluetooth

Soft blocked: no

Hard blocked: no

* + - 1. If the Bluetooth Interface hci0 is blocked, use the command **rfkill unblock bluetooth** to unlock it.
      2. Use the **systemctl status bluetooth.service** command to show the status of the Bluetooth service.

(pl-app) root@pi:/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 Tue 2018-05-22 00:00:02 UTC; 10h ago

Docs: man:bluetoothd(8)

Main PID: 522 (bluetoothd)

Status: "Running"

CGroup: /system.slice/bluetooth.service

└─522 /usr/lib/bluetooth/bluetoothd

<output omitted>

* + - 1. If the Bluetooth service status is not “active (running)”, use the command **systemctl start bluetooth.service** to start it.

(pl-app) root@pi:/home/pi/notebooks**# systemctl start bluetooth.service**

* + - 1. Use the **hciconfig hci0** command to display the status of the Bluetooth interface.

(pl-app) root@pi:/home/pi/notebooks# hciconfig hci0

hci0: Type: Primary Bus: UART

BD Address: B8:27:EB:72:CC:11 ACL MTU: 1021:8 SCO MTU: 64:1

UP RUNNING

RX bytes:6770 acl:46 sco:0 events:230 errors:0

TX bytes:3819 acl:46 sco:0 commands:90 errors:0

* + - 1. If the hci0 interface status is down, use the command **hciconfig hci0 up** to activate it.

(pl-app) root@pi:/home/pi/notebooks# hciconfig hci0 up

1. Using the CLI to Detect and Display Information about Bluetooth Devices
   * + 1. From the command line, use the command **bluetoothctl** to launch the Bluetooth tool. The prompt will change to **[bluetooth]** and will be displayed. It will display some information about the Bluetooth controller of your Raspberry Pi and probably some information about Bluetooth devices connected or available for pairing:

(pl-app) root@pi:/home/pi/notebooks# bluetoothctl

[NEW] Controller B8:27:EB:72:CC:11 raspberrypi [default]

[bluetooth]#

* + - 1. Enter the **scan on** command to start searching for nearby Bluetooth devices and **scan off** to stop it. You should see the MAC addresses and sometimes the device name of all Bluetooth devices discovered by the Raspberry Pi:

[bluetooth]# **scan on**

Discovery started

[CHG] Controller B8:27:EB:72:CC:11 Discovering: yes

[NEW] Device A4:C1:38:CC:F5:42 Triones-A4C138CCF542

[NEW] Device DA:B6:4A:F2:A8:BA Band

[NEW] Device 28:98:7B:E1:CE:D8 GT-S700

[bluetooth]# **scan off**

Take note of the MAC address (and if available the name) of some discovered Bluetooth devices:

DC:8B:28:54:2D:BE DESKTOP-T3QKUHD

78:DD:08:A5:76:63 LENOVO-THINK

A8:76:50:33:84:85 Pedro's A52s

* + - 1. Use the **info** command followed by the MAC address of a discovered device to obtain detailed information about the device:

[bluetooth]# **info** **DA:B6:4A:F2:A8:BA**

Device DA:B6:4A:F2:A8:BA

Name: Band

Alias: Band

Paired: no

Trusted: no

Blocked: no

Connected: no

LegacyPairing: no

UUID: Anhui Huami Information.. (0000fee0-0000-1000-8000-00805f9b34fb)

ManufacturerData Key: 0x0157

Take note of the information displayed and perform a web search for the meaning of some fields displayed and report below a summary of the main information displayed.

Device A8:76:50:33:84:85 (public)

Name: Pedro's A52s

Alias: Pedro's A52s

Class: 0x005a020c

Icon: phone

Paired: no

Trusted: no

Blocked: no

Connected: no

LegacyPairing: no

UUID: OBEX Object Push (00001105-0000-1000-8000-00805f9b34fb)

UUID: Audio Source (0000110a-0000-1000-8000-00805f9b34fb)

UUID: A/V Remote Control Target (0000110c-0000-1000-8000-00805f9b34fb)

UUID: A/V Remote Control (0000110e-0000-1000-8000-00805f9b34fb)

UUID: Headset AG (00001112-0000-1000-8000-00805f9b34fb)

UUID: PANU (00001115-0000-1000-8000-00805f9b34fb)

UUID: NAP (00001116-0000-1000-8000-00805f9b34fb)

UUID: Handsfree Audio Gateway (0000111f-0000-1000-8000-00805f9b34fb)

UUID: SIM Access (0000112d-0000-1000-8000-00805f9b34fb)

UUID: Phonebook Access Server (0000112f-0000-1000-8000-00805f9b34fb)

UUID: PnP Information (00001200-0000-1000-8000-00805f9b34fb)

UUID: Message Access Server (00001132-0000-1000-8000-00805f9b34fb)

**In Part 3, please do not try to pair your Raspberry to anybody else’s devices without permission.**

1. Using the CLI to Pair with a Sniffed Device
   * + 1. Use the command **paired-devices** to view the devices actually paired with the Raspberry Pi.

[bluetooth]# paired-devices

Report below the MAC address and name of the Bluetooth devices actually paired with your Raspberry Pi.

R: There are no pared devices yet.

* + - 1. Using the command **pair** followed by the MAC address, for example, **pair CA:4C:FB:5E:00:BA**, try to pair via Bluetooth the Raspberry Pi to some previously discovered devices.

[bluetooth]# pair CA:4C:FB:5E:00:BA

Attempting to pair with CA:4C:FB:5E:00:BA

<Output omitted>

[CHG] Device CA:4C:FB:5E:00:BA Paired: yes

Pairing successful

Are you able to pair without permission from the owner to some Bluetooth devices? Please, report below your findings:

R: No, had to give permission and confirm the right code.

* + - 1. Use the command **paired-devices** to view the devices now paired with the Raspberry Pi.

[bluetooth]# paired-devices

Device CA:4C:FB:5E:00:BA K800BA

Record the MAC address and name of the Bluetooth devices actually paired with your Raspberry Pi.

R: Device A8:76:50:33:84:85 Pedro's A52s

* + - 1. Using the command **connect** followed by the MAC address, for example, **connect CA:4C:FB:5E:00:BA**, try to connect via Bluetooth the Raspberry Pi to some previously paired devices.

[bluetooth]# **connect CA:4C:FB:5E:00:BA**

Attempting to connect to CA:4C:FB:5E:00:BA

[CHG] Device CA:4C:FB:5E:00:BA Connected: yes

Connection successful

[CHG] Device CA:4C:FB:5E:00:BA ServicesResolved: yes

[K800BA]#

Are you able to connect (without permission from the owner) to some Bluetooth devices? Write your findings below:

R: No, it throws an error.

* + - 1. Enter **quit** to exit the Bluetooth tool.

[bluetooth]# **quit**

1. Clean Up
   * + 1. Enter **systemctl stop bluetooth.service** to disable the Bluetooth service.

(pl-app) root@Pi-3:/home/pi/notebooks# **systemctl stop bluetooth.service**

* + - 1. Verify that the Bluetooth service has been disabled with the **systemctl status bluetooth.service** command.

(pl-app) root@Pi-3:/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 Tue 2018-11-27 21:47:30 UTC; 1min 10s ago

Docs: man:bluetoothd(8)

Process: 843 ExecStart=/usr/lib/bluetooth/bluetoothd (code=exited, status=0/SUCCESS)

Main PID: 843 (code=exited, status=0/SUCCESS)

Status: "Quitting"

<output omitted>