Hands-on Session: Simple Attacks on Wi-Fi Networks

Yug Patel - CS23MTECH14019
Somya Jain - CS23MTECH12011
Manan Patel - CS23MTECH14006

Task-1: DoS attacks on a victim's Wi-Fi STA

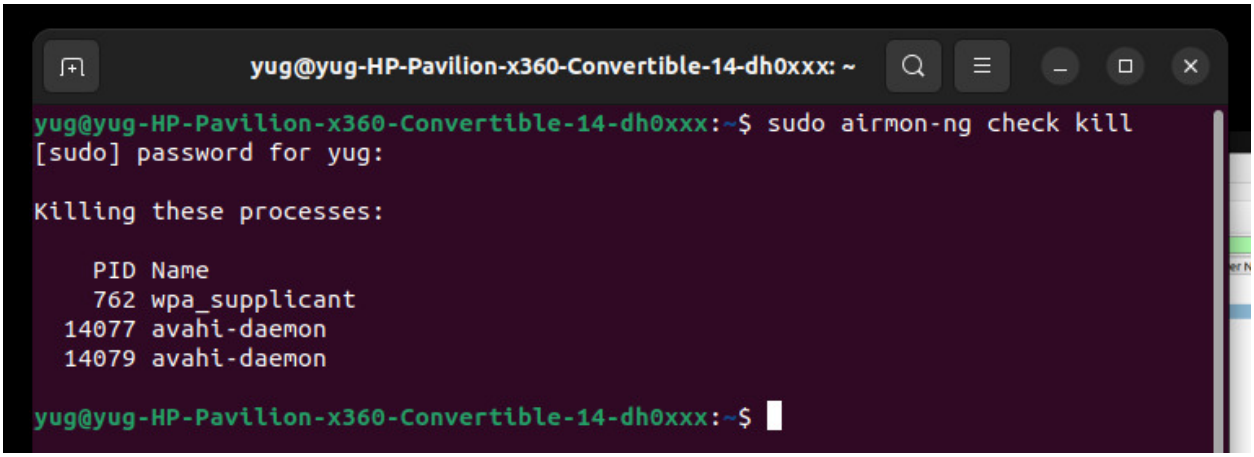
S1: Configure one STA (laptop or smartphone) as a client and connect it to IITH-Guest Wi-Fi AP

The Access Point (AP) is identified by the name 'Yug's 2.4 GHz' with the BSSID '3C:52:A1:97:89:5C'
The MAC address of the client is 20-C1-9B-58-DB-DA.

S2: Sniff traffic between STA and IITH-Guest Wi-Fi AP using a Wi-Fi sniffer (configure another laptop in monitor mode to listen to packets exchanged between STA and AP by using airmon-ng and airdump-ng tools. You can also use wireshark/tcpdump with appropriate filters on the sniffer laptop to observe the traffic once you keep the Wi-Fi radio of the sniffer laptop in monitor mode using airmon-ng or iw command)

Commands utilized for configuring the laptop in monitor mode:

1. **Verifying and terminating processes that could disrupt wireless network monitoring.**



2. **Inspecting the wireless interfaces of the laptop.**

```
yug@yug-HP-Pavilion-x360-Convertible-14-dh0xxx:~$ iwconfig
lo        no wireless extensions.

wlo1      IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

docker0   no wireless extensions.

hotspot   IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

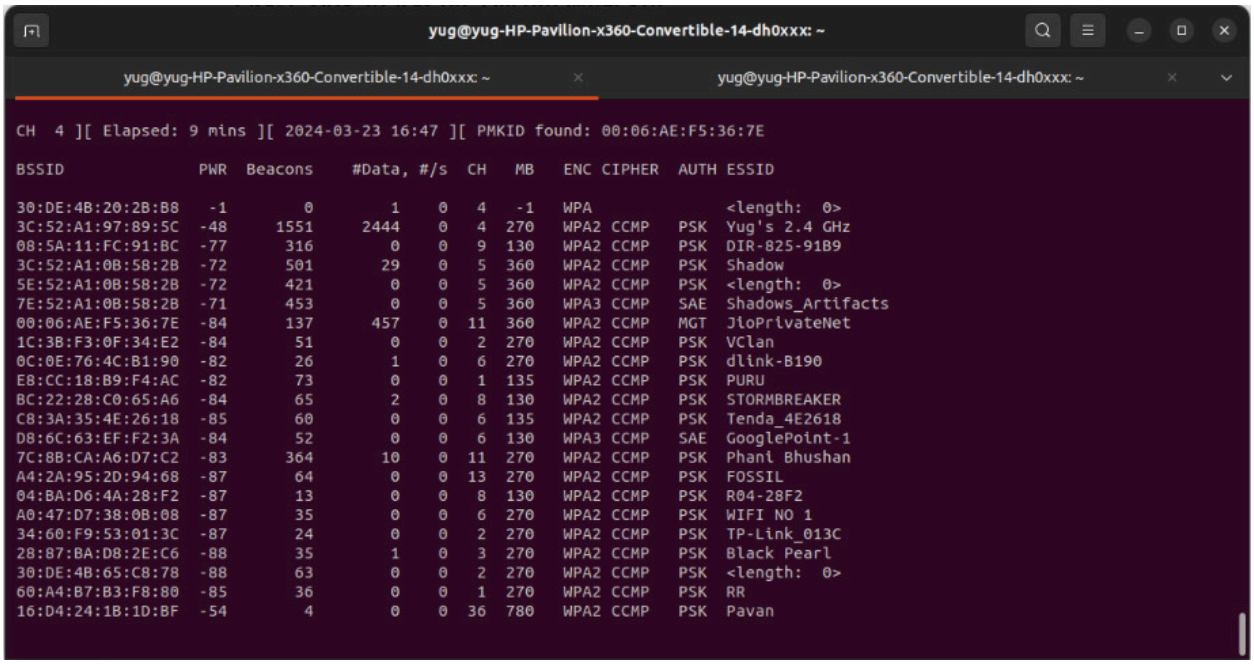
3. **Initiate a wireless interface in monitor mode:**

```
yug@yug-HP-Pavilion-x360-Convertible-14-dh0xxx:~$ sudo airmon-ng start wlo1

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
  14091 avahi-daemon
  14092 avahi-daemon

PHY     Interface       Driver          Chipset

phy0    hotspot         rtw_8822be      Realtek Semiconductor Co., Ltd. RTL8822BE 802.11a/b/g/n/ac WiFi adapter
phy0    wlo1            rtw_8822be      Realtek Semiconductor Co., Ltd. RTL8822BE 802.11a/b/g/n/ac WiFi adapter
                (monitor mode enabled)
```

4. **Verifying the name of the created monitor interface.**

```
                (monitor mode enabled)

yug@yug-HP-Pavilion-x360-Convertible-14-dh0xxx:~$ iwconfig
lo        no wireless extensions.

wlo1      IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

docker0   no wireless extensions.

hotspot   IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

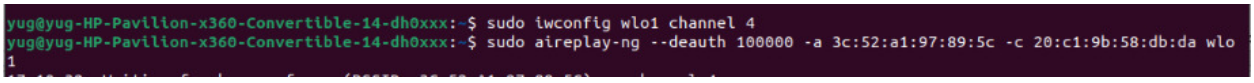
5. **To view all access points (APs) available in the vicinity:**



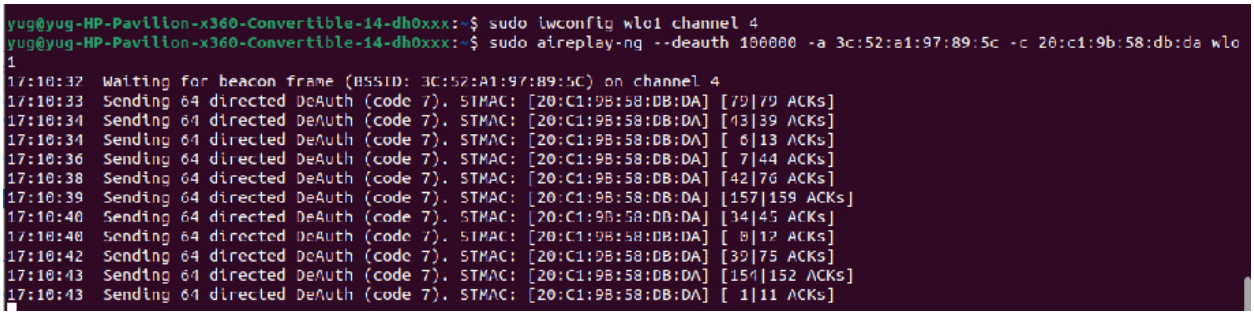
It is noted that the AP ‘Yug’s 2.4 GHz' with BSSID '3C:52:A1:97:89:5C' is utilizing channel number (CH) 4 for communication.

S3: Answer:

a. **The DOS attack is initiated on the client using the following command:**

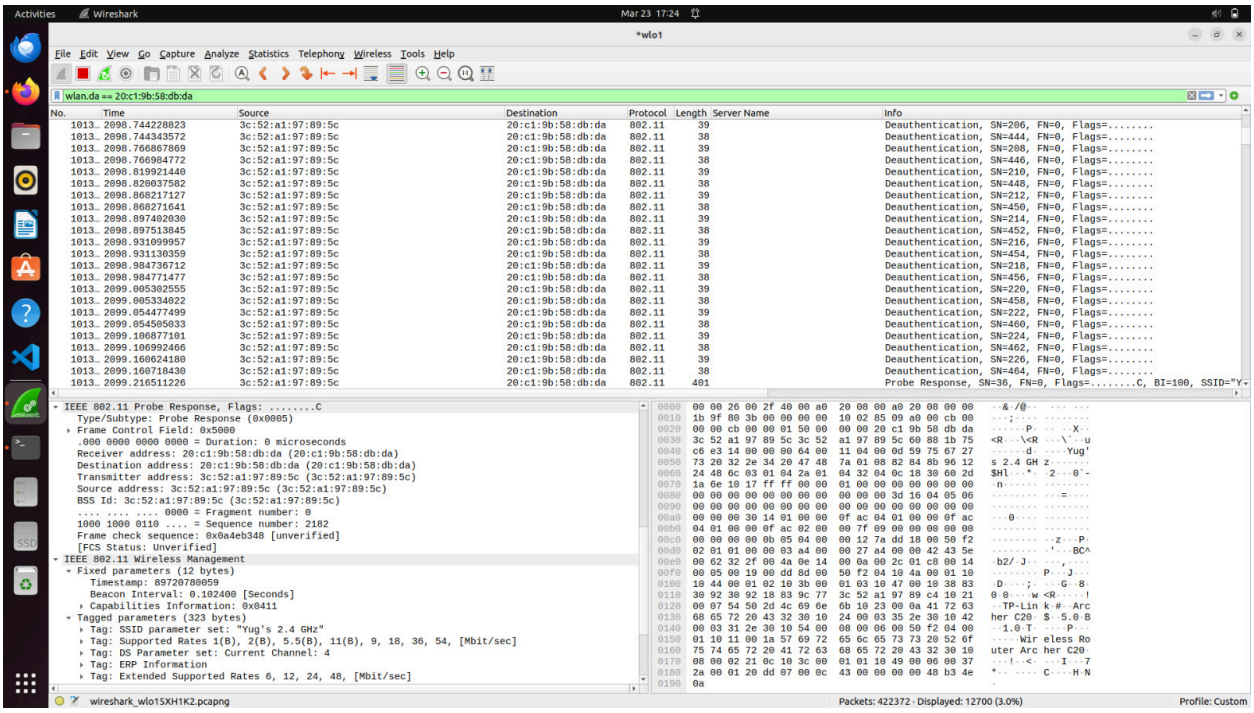


b. **In this scenario, the BSSID of the AP is** '3C:52:A1:97:89:5C'**, and the MAC address of the client is ‘**20-C1-9B-58-DB-DA’**.**

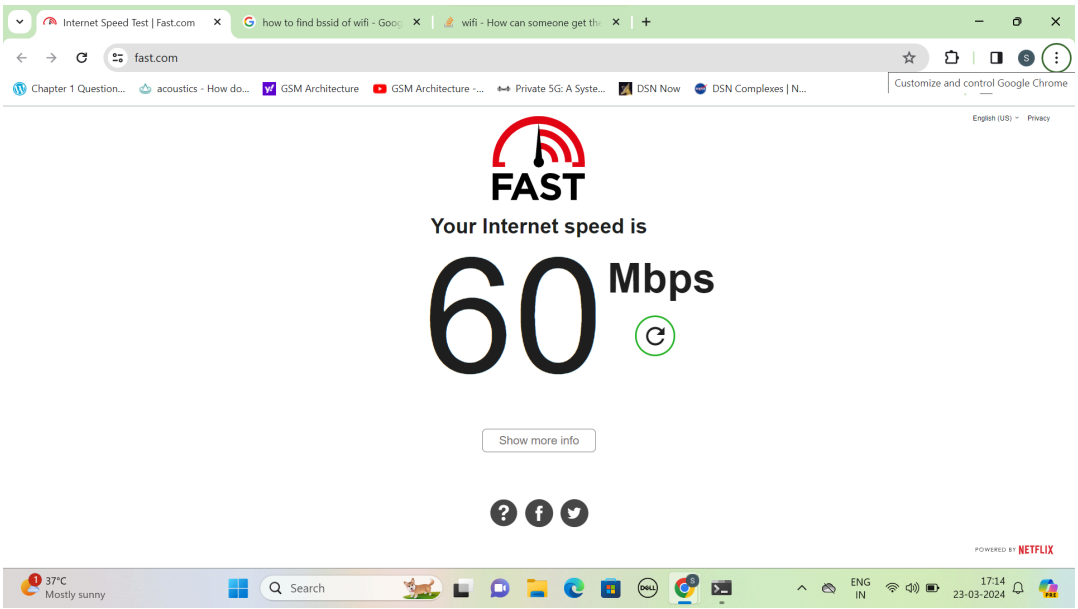


S4: Answer:

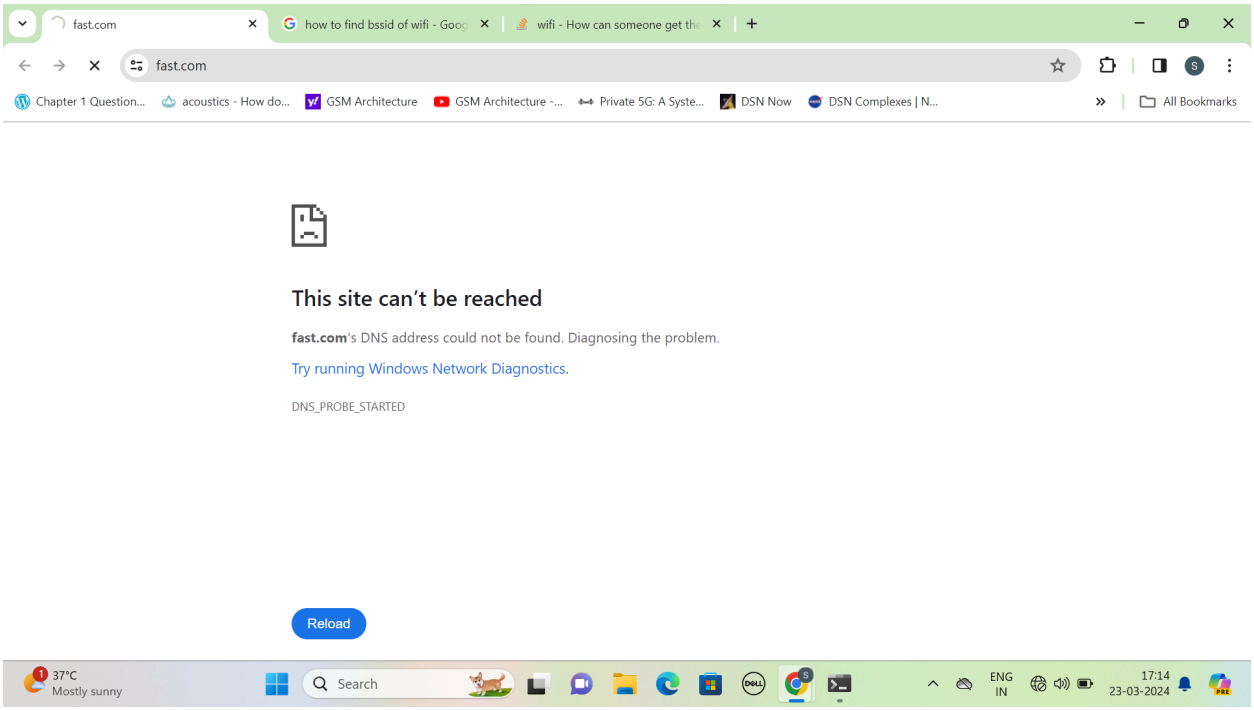
a. Wireshark traces



b. Victim's system initially



c. Impact on Victim's system



Note: We forgot to capture the .pcap file corresponding to the above mentioned MAC address. So the attached .pcap file is captured with another device.

Task-2: Snoop into HTTP traffic of a victim Wi-Fi STA

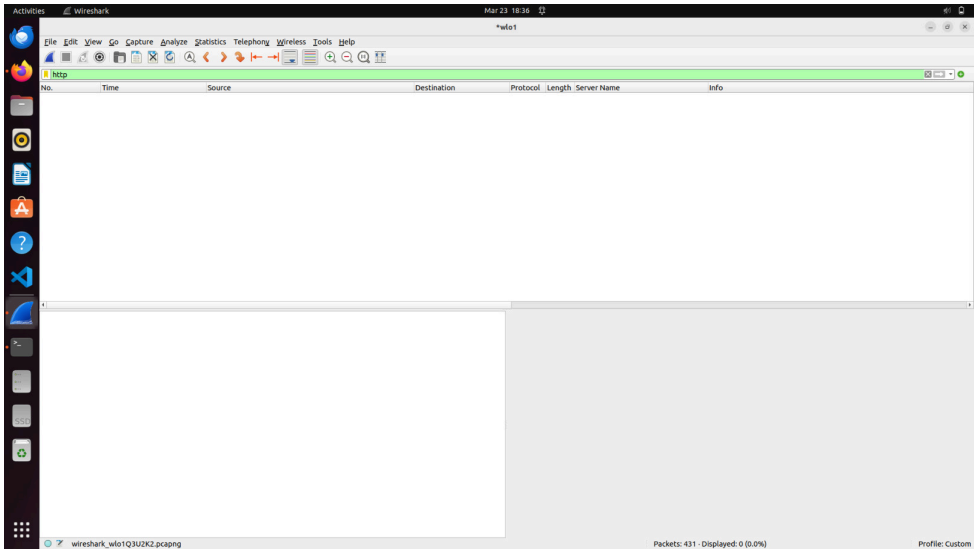
S1: Configure one STA (laptop or smartphone) as a client and connect it to IITH-Guest Wi-Fi AP

The Access Point (AP) is identified by the name 'Manan'

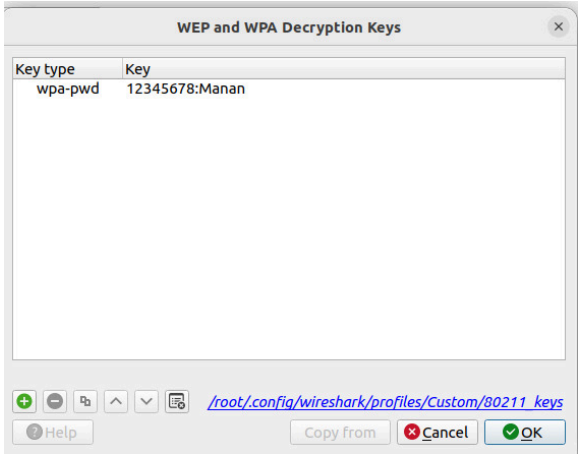
The MAC address of the client is 20-C1-9B-58-DB-DA.

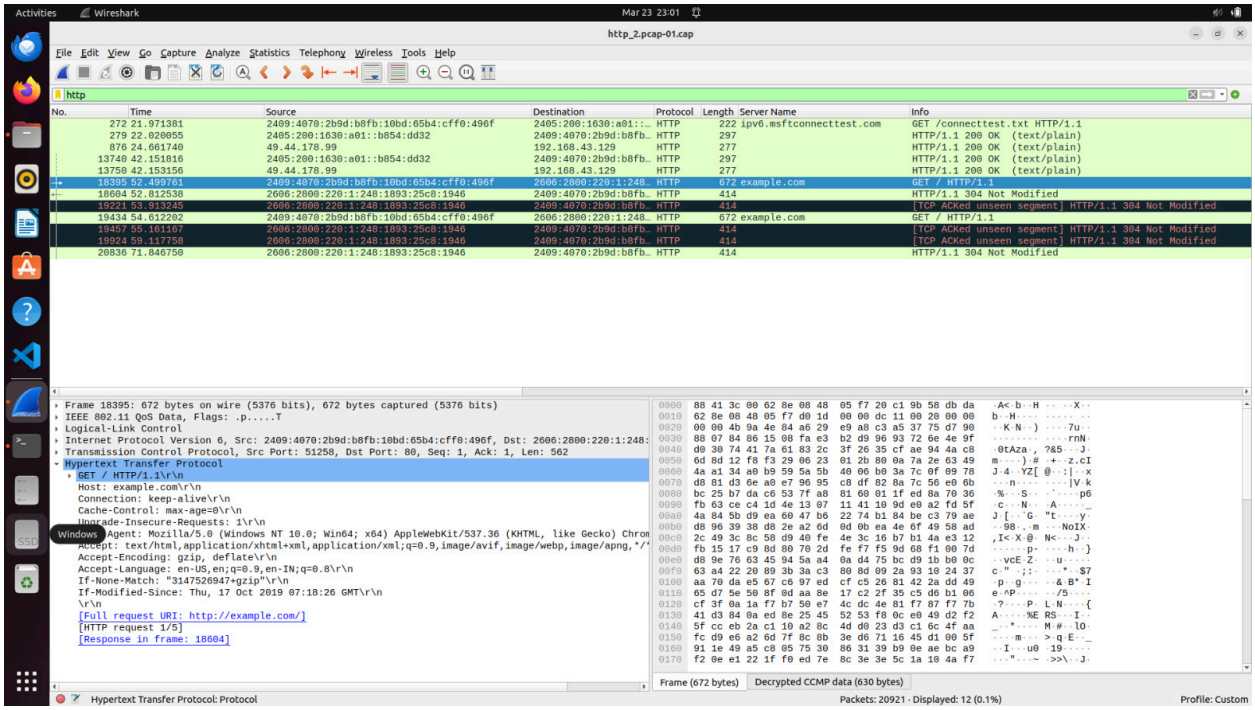
S2: Same task as above

S3: Open this pcap in wireshark to check whether you could see any HTTP traffic between victim STA and example.com



S4. Open wireshark again and key in 122345678. Now check for presence of any HTTP traffic due to automatic decryption of link-level encrypted L2 packets.



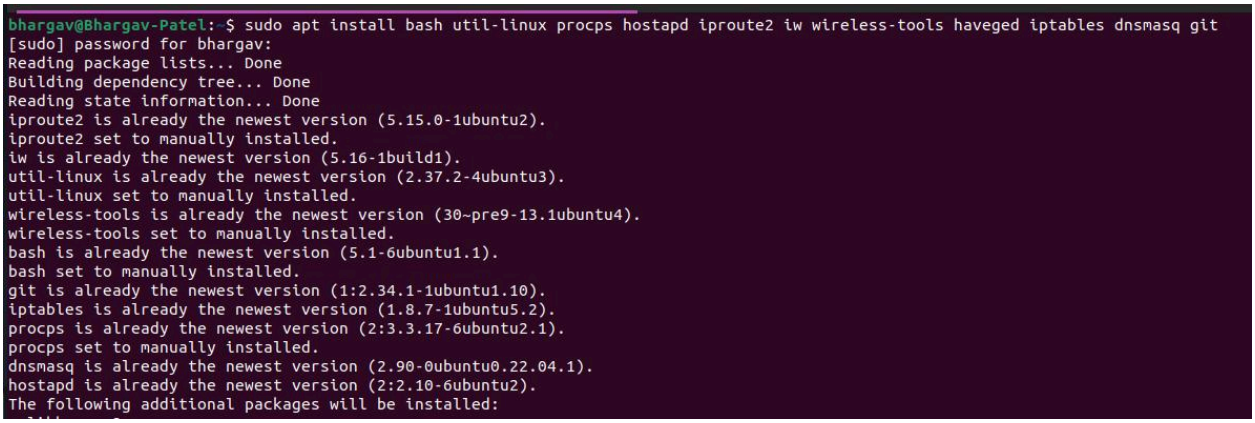

Activities
Wireshark
Mar 23 23:01
http_2.pcap-01.cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Server Name Info
272 21.971381 2409:4070:2b9d:b8fb:10bd:65b4:cff0:496f 2405:200:1630:a01::… HTTP 222 ipv6.msftconnecttest.com GET /connecttest.txt HTTP/1.1
279 22.020055 2405:200:1630:a01::b854:dd32 2409:4070:2b9d:b8fb… HTTP 297 HTTP/1.1 200 OK (text/plain)
876 24.661740 49.44.178.99 192.168.43.129 HTTP 277 HTTP/1.1 200 OK (text/plain)
13740 42.151816 2405:200:1630:a01::b854:dd32 2409:4070:2b9d:b8fb… HTTP 297 HTTP/1.1 200 OK (text/plain)
13750 42.153156 49.44.178.99 192.168.43.129 HTTP 277 HTTP/1.1 200 OK (text/plain)
18395 52.499761 2409:4070:2b9d:b8fb:10bd:65b4:cff0:496f 2606:2800:220:1:248… HTTP 672 example.com GET / HTTP/1.1
18604 52.812538 2606:2800:220:1:248:1893:25c8:1946 2409:4070:2b9d:b8fb… HTTP 414 HTTP/1.1 304 Not Modified
19221 53.813245 2606:2800:220:1:248:1893:25c8:1946 2409:4070:2b9d:b8fb… HTTP 414 [TCP ACKed unseen segment] HTTP/1.1 304 Not Modified
19434 54.612202 2409:4070:2b9d:b8fb:10bd:65b4:cff0:496f 2606:2800:220:1:248… HTTP 672 example.com GET / HTTP/1.1
19457 55.161167 2606:2800:220:1:248:1893:25c8:1946 2409:4070:2b9d:b8fb… HTTP 414 [TCP ACKed unseen segment] HTTP/1.1 304 Not Modified
19924 59.117758 2606:2800:220:1:248:1893:25c8:1946 2409:4070:2b9d:b8fb… HTTP 414 [TCP ACKed unseen segment] HTTP/1.1 304 Not Modified
20836 71.846750 2606:2800:220:1:248:1893:25c8:1946 2409:4070:2b9d:b8fb… HTTP 414 HTTP/1.1 304 Not Modified
Frame 18395: 672 bytes on wire (5376 bits), 672 bytes captured (5376 bits)
IEEE 802.11 QoS Data, Flags: .p.....T
Logical-Link Control
Internet Protocol Version 6, Src: 2409:4070:2b9d:b8fb:10bd:65b4:cff0:496f, Dst: 2606:2800:220:1:248:
Transmission Control Protocol, Src Port: 51258, Dst Port: 80, Seq: 1, Ack: 1, Len: 562
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
Host: example.com\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9,en-IN;q=0.8\r\n
If-None-Match: "3147526947+gzip"\r\n
If-Modified-Since: Thu, 17 Oct 2019 07:18:26 GMT\r\n
\r\n
[Full request URI: http://example.com/]
[HTTP request 1/5]
[Response in frame: 18604]
Windows
0000 88 41 3c 00 62 8e 08 48 05 f7 20 c1 9b 58 db da
0010 62 8e 08 48 05 f7 d0 1d 00 00 dc 11 00 20 00 00
0020 00 00 4b 9a 4e 84 a6 29 e9 a8 c3 a5 37 75 d7 90
0030 88 07 84 86 15 08 fa e3 b2 d9 96 93 72 6e 4e 9f
0040 d0 30 74 41 7a 61 83 2c 3f 26 35 cf ae 94 4a c8
0050 6d 8d 12 f8 f3 29 06 23 01 2b 80 0a 7a 2e 63 49
0060 4a a1 34 a0 b9 59 5a 5b 40 06 b0 3a 7c 0f 09 78
0070 d8 81 d3 6e a0 e7 96 95 c8 df 82 8a 7c 56 e0 6b
0080 bc 25 b7 da c6 53 7f a8 81 60 01 1f ed 8a 70 36
0090 fb 63 ce c4 1d 4e 13 07 11 41 10 9d e0 a2 fd 5f
00a0 4a 84 5b d9 ea 60 47 b6 22 74 b1 84 be c3 79 ae
00b0 d8 96 39 38 d8 2e a2 6d 0d 0b ea 4e 0f 49 58 ad
00c0 2c 49 3c 8c 58 d9 40 fe 4e 3c 16 b7 b1 4a e3 12
00d0 fb 15 17 c9 8d 80 70 2d fe f7 f5 9d 68 f1 00 7d
00e0 d8 9e 76 63 45 94 5a a4 0a d4 75 bc d9 1b b0 0c
00f0 63 a4 22 20 89 3b 3a c3 80 8d 09 2a 93 10 24 37
0100 aa 70 da e5 67 c6 97 ed cf c5 26 81 42 2a dd 49
0110 65 d7 5e 50 8f 0d aa 8e 17 c2 2f 35 c5 d6 b1 06
0120 cf 3f 0a 1a f7 b7 50 e7 4c dc 4e 81 f7 87 f7 7b
0130 41 d3 84 0a ed 8e 25 45 52 53 f8 0c e0 49 d2 f2
0140 5f cc eb 2a c1 10 a2 8c 4d d0 23 d3 c1 6c 4f aa
0150 fc d9 e6 a2 6d 7f 8c 8b 3e d6 71 16 45 d1 00 5f
0160 91 1e 49 a5 c8 05 75 30 86 31 39 b9 0e ae bc a9
0170 f2 0e e1 22 1f f0 ed 7e 8c 3e 3e 5c 1a 10 4a f7
Frame (672 bytes)
Decrypted CCMP data (630 bytes)
Hypertext Transfer Protocol: Protocol
Packets: 20921 · Displayed: 12 (0.1%)
Profile: Custom

Task-3: MITM attacks on a Wi-Fi Network

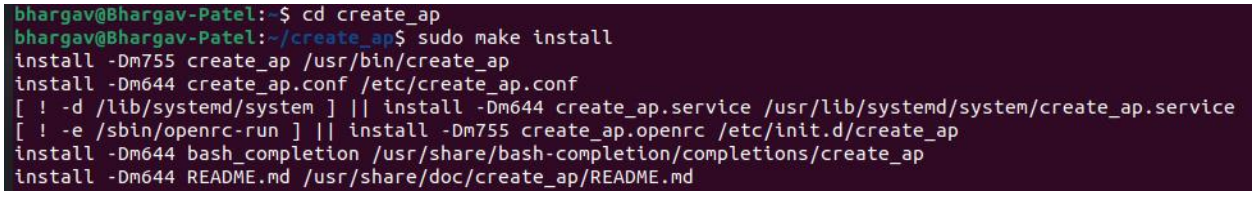
S1: MITM attacks on Wi-Fi networks by an open Wi-Fi network

The following steps were followed for creating an open WiFi Access Point named **"FreeWiFi"**:-

(1) We have used a tool named **create_ap** to create a virtual interface on the wireless card as a free open access point. So first installing the required dependencies for the tool.



(2) The below picture demonstrates the commands used to clone and install the networking utility tool.



(3) Creating a virtual interface named **“hotspot”** using iw command with BSSID as **12:34:56:78:ab:cd**. And then setting up the interface with IP **192.168.28.1.**

command for creating virtual interface:

sudo iw phy phy0 interface add hotspot type __ap

command for setting up that interface with an ip:

sudo ifconfig hotspot 192.168.28.1 up

```
root@Bhargav-Patel:/home/bhargav# iwconfig
lo        no wireless extensions.

enp46s0   no wireless extensions.

wlp45s0   IEEE 802.11  ESSID:"Yug's 2.4 GHz"
          Mode:Managed  Frequency:2.417 GHz  Access Point: 3C:52:A1:97:89:5C
          Bit Rate=300 Mb/s   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=70/70  Signal level=-25 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:53   Missed beacon:0

hotspot   IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on

ap0       IEEE 802.11  Mode:Master  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

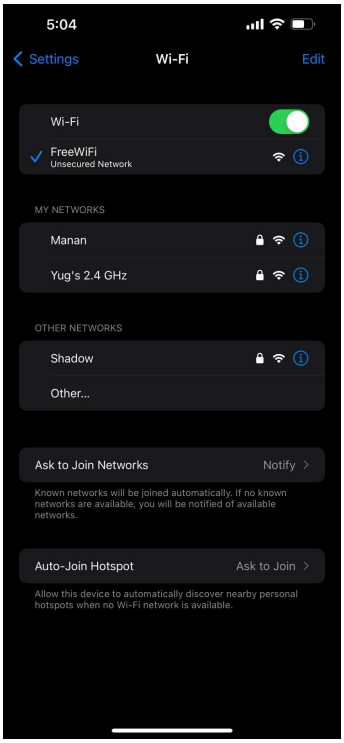
The hotspot interface is identified as one of the wireless extensions

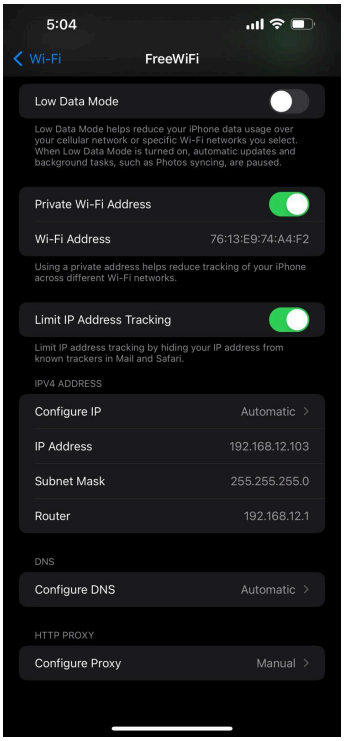
```
hotspot: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.28.1  netmask 255.255.255.0  broadcast 192.168.28.255
        ether 12:34:56:78:ab:cd  txqueuelen 1000  (Ethernet)
        RX packets 278  bytes 66518 (66.5 KB)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 134  bytes 40848 (40.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

*Networking parameter of the created in **“hotspot”** interface*

(4) Finally the open access point **“FreeWiFi”** is created using ***create_ap*** command. The mentioned command internally uses hostapd, udhcpd and iptables utilities and creates the access point.

```
bhargav@Bhargav-Patel:~$ sudo create_ap wlp45s0 hotspot FreeWiFi
Config dir: /tmp/create_ap.wlp45s0.conf.MTiNUhI0
PID: 12619
Network Manager found, set ap0 as unmanaged device... DONE
wlp45s0 is already associated with channel 2 (2417 MHz), fallback to channel 2
Creating a virtual WiFi interface... ap0 created.
Sharing Internet using method: nat
hostapd command-line interface: hostapd_cli -p /tmp/create_ap.wlp45s0.conf.MTiNUhI0/hostapd_ctrl
ap0: interface state UNINITIALIZED->ENABLED
ap0: AP-ENABLED
```



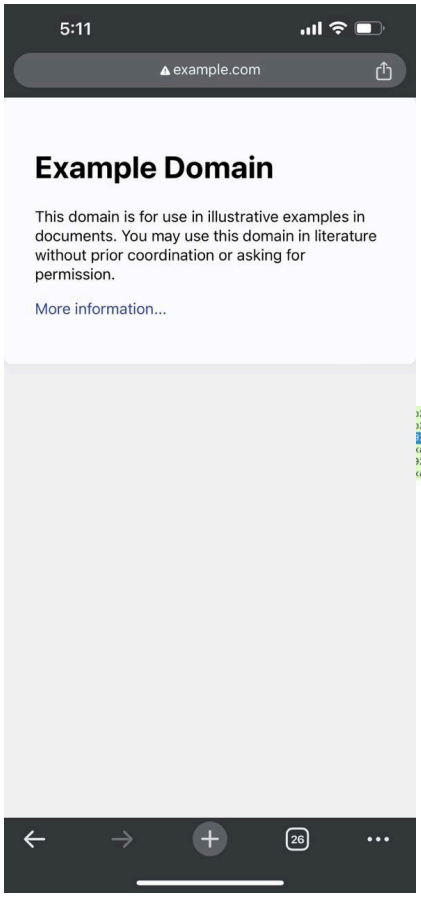


The images aside shows how a victim client with gets connected to open access point and also receives IP from the access point.

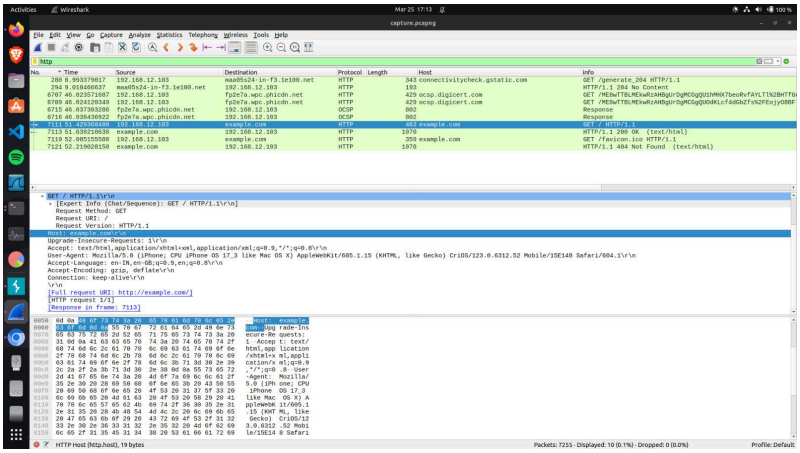
(5) Voila!! We can see the vitim client with MAC 76:13:e9:74:a4:f2 is conned to the rouge open access point.

```
ap0: STA 76:13:e9:74:a4:f2 IEEE 802.11: authenticated
ap0: STA 76:13:e9:74:a4:f2 IEEE 802.11: associated (aid 1)
ap0: AP-STA-CONNECTED 76:13:e9:74:a4:f2
```

S2: Let the victim client visit example.com over http and show that MITM attacker observes (passive attacker) into http traffic between the victim and remote webserver.



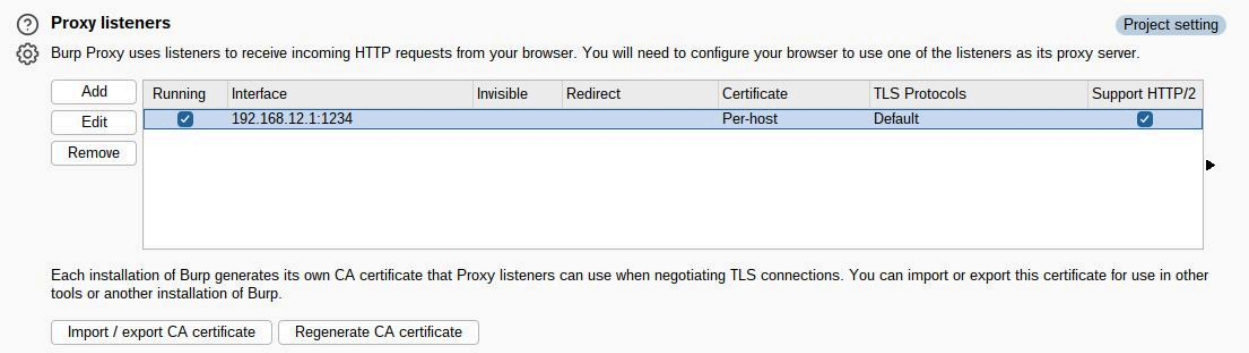
The victim visits example.com over port 80



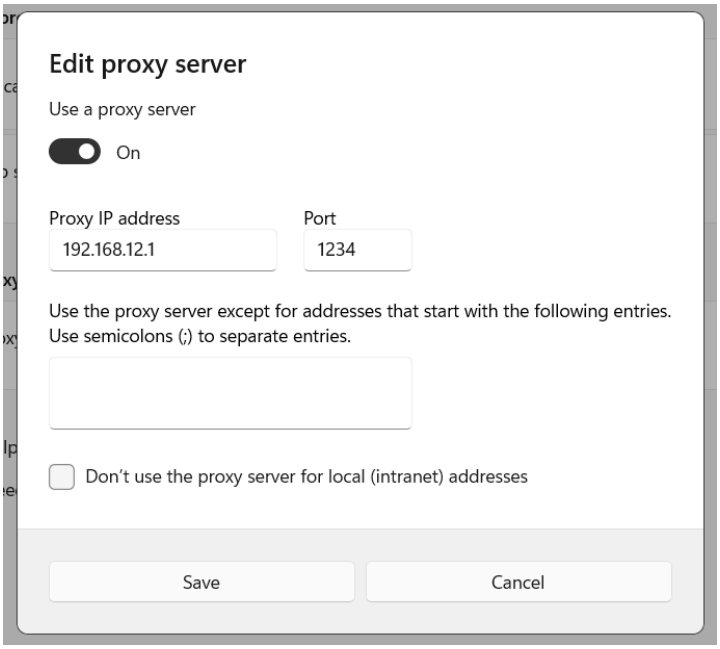
Entire wireshark capture

S3: Active MITM attacker: Show how MITM attacker could modify HTTP responses from example.com by injecting custom HTML code or javascript.

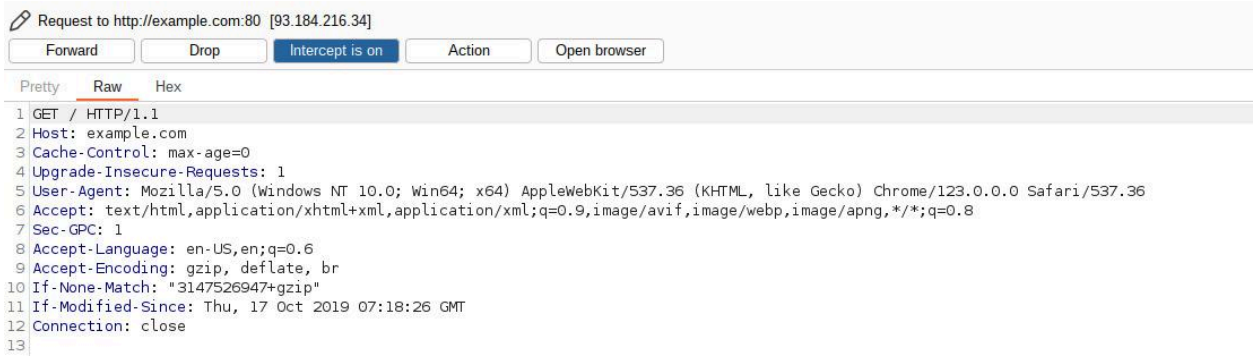
We have used burp-suite as a proxy server which captures http requests and responses. Using which we enabled the active attack by manipulating the response from example.com.



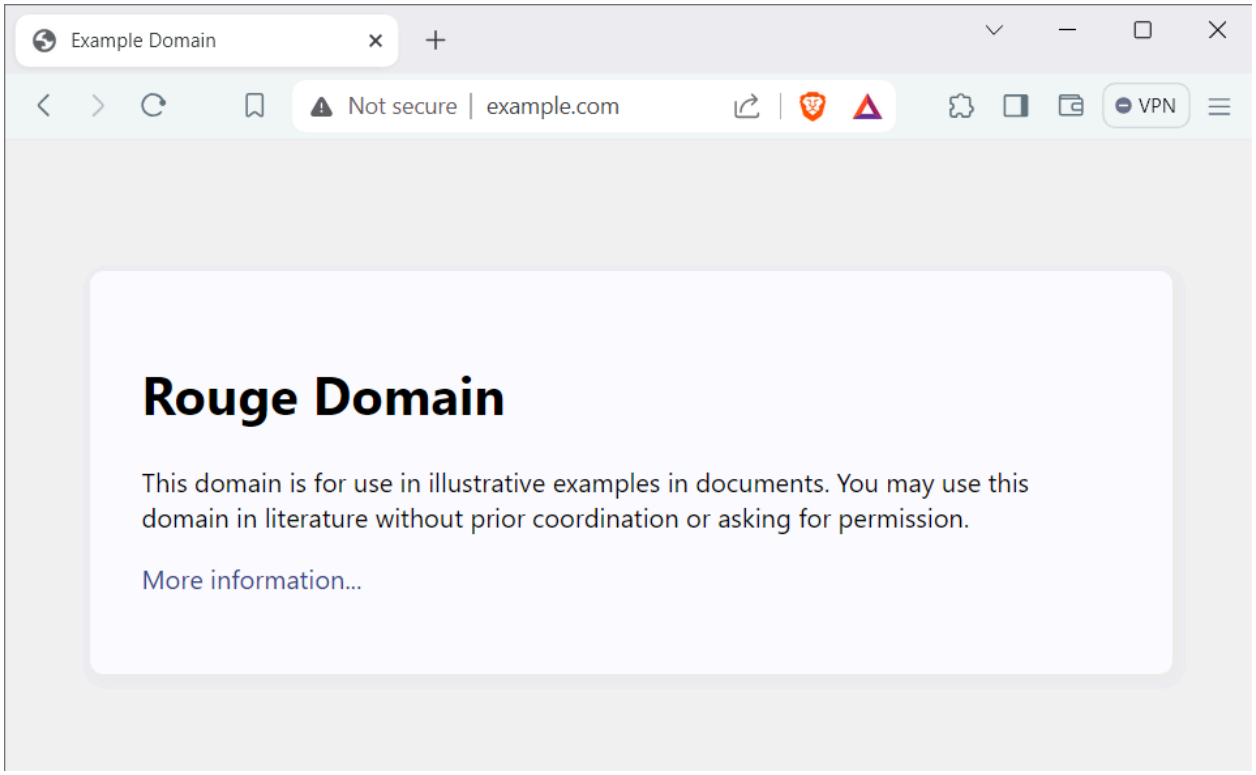
Burp-suite is listing on 192.168.12.1:1234



Victim's end is configured to hit on that proxy



Burp-suite captures the request to the 'example.com'



Modified response from burp-suite

Note: We forgot to capture the screenshot for changes we made in brup-suite for the response, later we tried with many different browsers but all were caching the same, we didn't catch the response.

References:

- https://thecybersecurityman.com/2018/08/11/creating-an-evil-twin-or-fake-access-point-using-aircrack-ng-and-dnsmasq-part-2-the-attack/
- https://anooppoommen.medium.com/create-a-wifi-hotspot-on-linux-29349b9c582d
- https://witestlab.poly.edu/blog/conduct-a-simple-man-in-the-middle-attack-on-a-wifi-hotspot/
- https://askubuntu.com/questions/318973/how-do-i-create-a-wifi-hotspot-sharing-wireless-internet-connection-single-adap/324785#324785
- https://wiki.archlinux.org/title/software_access_point#Wireless_client_and_software_AP_with_a_single_Wi-Fi_device
- https://w1.fi/hostapd/
- https://wiki.archlinux.org/title/Network_configuration/Wireless
- https://www.howtogeek.com/214080/how-to-turn-your-windows-pc-into-a-wi-fi-hotspot/