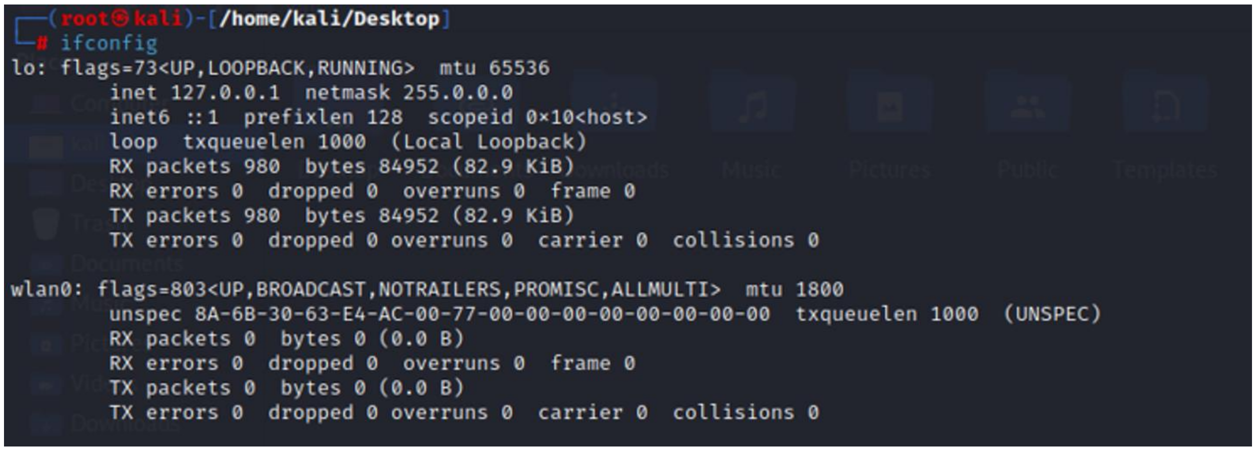
1. Airmon-ng

On wireless interfaces, monitor mode can be enabled using this script. Alternatively, it can be used to terminate network managers or switch back from managed to monitor mode. The status of the interfaces can be seen by entering the airmon-ng command without any arguments. It can also make a list of programs that could harm the wireless card's performance and delete them.IN the above screenshot we can see that we have started the air-mon and then kill the processes.

Uses: An ethical hacker records each of these packets to see if the router is exposed or not. It is also employed to ascertain whether the network is at risk. Every device has all the information required. It is also used to monitor the busy traffic.

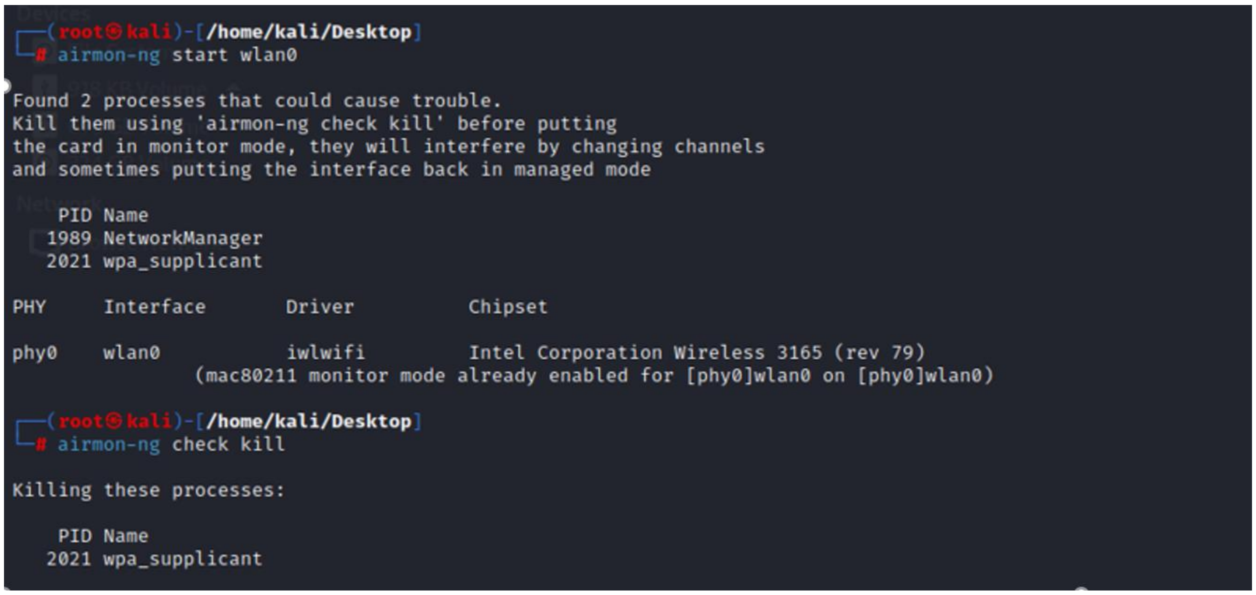
Test cases

1.



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# ifconfig
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 980  bytes 84952 (82.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 980  bytes 84952 (82.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=803<UP,BROADCAST,NOTRAILERS,PROMISC,ALLMULTI>  mtu 1800
        unspec 8A-6B-30-63-E4-AC-00-77-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2.

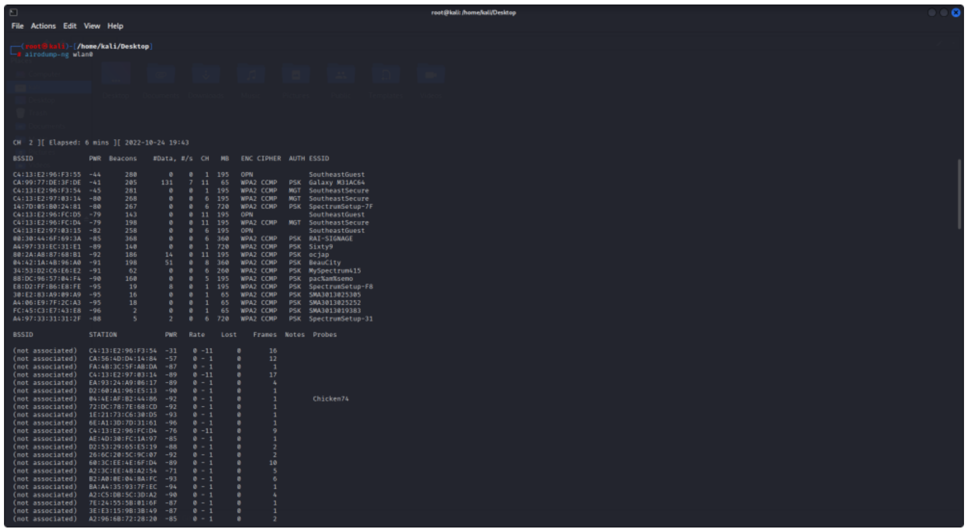

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# airmon-ng start wlan0

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
   1989 NetworkManager
   2021 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           iwlwifi         Intel Corporation Wireless 3165 (rev 79)
                (mac80211 monitor mode already enabled for [phy0]wlan0 on [phy0]wlan0)

┌──(root㉿kali)-[/home/kali/Desktop]
└─# airmon-ng check kill

Killing these processes:

    PID Name
   2021 wpa_supplicant
```

2. Airodump-ng

Using Airodump-ng, raw 802.11 frames are captured during packet capture. In order to use them with aircrack-ng, it is especially well suited for gathering WEP IVs (Initialization Vector) or WPA handshakes. The GPS coordinates of the discovered access points can be recorded by airodump-ng if a GPS receiver is attached to the PC. Additionally, airodump-ng writes out several files containing the details of all access points and clients seen, which can be used for scripting, or creating custom tools

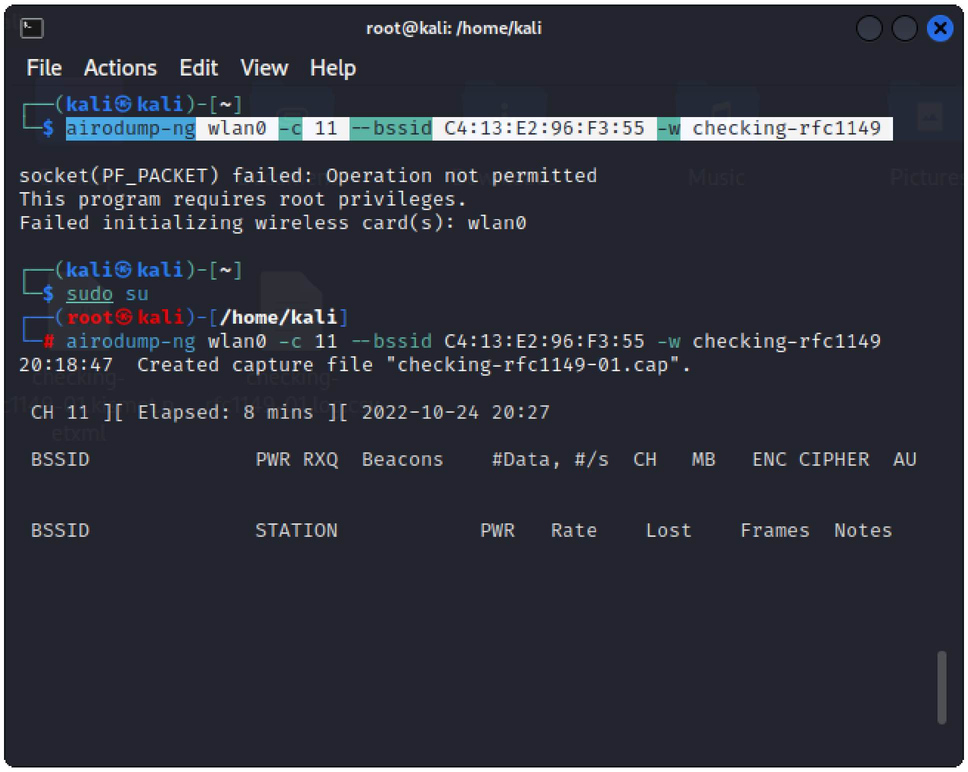
It is a packet capture tool that records and stores unprocessed data packets for further examination. Airodump-ng may also retrieve the access point coordinates if your computer is connected to a GPS receiver. You can begin capturing packets with airodump once monitor mode has been enabled with airmon-ng. All the nearby networks are listed using the Airdump-ng tool, which also provides vital information about each one. Since it is a packet sniffer, its primary purpose when we are in Monitor mode is to collect all the packets surrounding us. In order to gather valuable data, such as the mac address, channel name, encryption type, and the number of clients connected to the network, we can run it against all of the nearby networks before directing it at the target network. To just capture packets from a specific Wi-Fi network, we may also run it against a specific AP (access point).

Test case 1:



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# airodump-ng wlan0

 CH  2 ][ Elapsed: 6 mins ][ 2022-10-24 19:43

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

 C4:13:E2:96:F3:55  -44      280        0    0   1  195   OPN              SoutheastGuest
 CA:99:77:DE:3F:DE  -41      205      131    7  11   65   WPA2 CCMP   PSK  Galaxy M31AC64
 C4:13:E2:96:F3:54  -45      281        0    0   1  195   WPA2 CCMP   MGT  SoutheastSecure
 C4:13:E2:97:03:14  -80      268        0    0   6  195   WPA2 CCMP   MGT  SoutheastSecure
 14:7D:05:B0:24:81  -80      267        0    0   6  720   WPA2 CCMP   PSK  SpectrumSetup-7F
 C4:13:E2:96:FC:D5  -79      143        0    0  11  195   OPN              SoutheastGuest
 C4:13:E2:96:FC:D4  -79      198        0    0  11  195   WPA2 CCMP   MGT  SoutheastSecure
 C4:13:E2:97:03:15  -82      258        0    0   6  195   OPN              SoutheastGuest
 00:30:44:6F:69:3A  -85      368        0    0   6  360   WPA2 CCMP   PSK  RAI-SIGNAGE
 A4:97:33:EC:31:E1  -89      140        0    0   1  720   WPA2 CCMP   PSK  Sixty9
 80:2A:A8:87:68:B1  -92      186       14    0  11  195   WPA2 CCMP   PSK  ocjap
 04:42:1A:4B:96:A0  -91      198       51    0   8  360   WPA2 CCMP   PSK  BeauCity
 34:53:D2:C6:E6:E2  -91       62        0    0   6  260   WPA2 CCMP   PSK  MySpectrum415
 88:DC:96:57:04:F4  -90      160        0    0   5  195   WPA2 CCMP   PSK  pac%am%semo
 E8:D2:FF:B6:E8:FE  -95       19        8    0   1  195   WPA2 CCMP   PSK  SpectrumSetup-F8
 30:E2:83:A9:09:A9  -95       16        0    0   1   65   WPA2 CCMP   PSK  SMA3013025305
 A4:06:E9:7F:2C:A3  -95       18        0    0   1   65   WPA2 CCMP   PSK  SMA3013025252
 FC:45:C3:E7:43:E8  -96        2        0    0   1   65   WPA2 CCMP   PSK  SMA3013019383
 A4:97:33:31:31:2F  -88        5        2    0   6  720   WPA2 CCMP   PSK  SpectrumSetup-31

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 (not associated)   C4:13:E2:96:F3:54  -31    0 -11      0       16
 (not associated)   CA:56:4D:D4:14:84  -57    0 - 1      0       12
 (not associated)   FA:4B:3C:5F:AB:DA  -87    0 - 1      0        1
 (not associated)   C4:13:E2:97:03:14  -89    0 -11      0       17
 (not associated)   EA:93:24:A9:06:17  -89    0 - 1      0        4
 (not associated)   D2:60:A1:96:E5:13  -90    0 - 1      0        1
 (not associated)   04:4E:AF:B2:44:86  -92    0 - 1      0        1         Chicken74
 (not associated)   72:DC:78:7E:68:CD  -92    0 - 1      0        1
 (not associated)   1E:21:73:C6:30:D5  -93    0 - 1      0        1
 (not associated)   6E:A1:3D:7D:31:61  -96    0 - 1      0        1
 (not associated)   C4:13:E2:96:FC:D4  -76    0 -11      0        9
 (not associated)   AE:4D:30:FC:1A:97  -85    0 - 1      0        1
 (not associated)   D2:53:29:65:E5:19  -88    0 - 1      0        2
 (not associated)   26:6C:20:5C:9C:07  -92    0 - 1      0        2
 (not associated)   60:3C:EE:4E:6F:D4  -89    0 - 1      0       10
 (not associated)   A2:3C:EE:48:A2:54  -71    0 - 1      0        5
 (not associated)   B2:A0:0E:04:8A:FC  -93    0 - 1      0        6
 (not associated)   BA:A4:35:93:7F:EC  -94    0 - 1      0        1
 (not associated)   A2:C5:DB:5C:3D:A2  -90    0 - 1      0        4
 (not associated)   7E:24:55:5B:01:6F  -87    0 - 1      0        1
 (not associated)   3E:E3:15:9B:3B:49  -87    0 - 1      0        1
 (not associated)   A2:96:6B:72:28:20  -85    0 - 1      0        2
```

Test case 2



```
┌──(kali㉿kali)-[~]
└─$ airodump-ng wlan0 -c 11 --bssid C4:13:E2:96:F3:55 -w checking-rfc1149

socket(PF_PACKET) failed: Operation not permitted
This program requires root privileges.
Failed initializing wireless card(s): wlan0

┌──(kali㉿kali)-[~]
└─$ sudo su
┌──(root㉿kali)-[/home/kali]
└─# airodump-ng wlan0 -c 11 --bssid C4:13:E2:96:F3:55 -w checking-rfc1149
20:18:47  Created capture file "checking-rfc1149-01.cap".

 CH 11 ][ Elapsed: 8 mins ][ 2022-10-24 20:27

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


 BSSID              STATION            PWR   Rate    Lost    Frames  Notes
```

3. Airbase-ng

In contrast to the Access Point (AP), Airbase-ng is a versatile tool designed to attack customers. The implementation's major goal is to persuade users to connect to the fictitious access point rather than preventing them from using the genuine hotspot. Running airbase-ng results in the creation of a touch interface (atX). This can be used to send encrypted packets or to receive packets that have been decrypted. These frames are crucial for connecting real clients to our softAP since they are more likely to send probing requests for shared/configured networks. In such a scenario, the access point will respond to any probe request with a proper probe response, alerting the authentication client depending on the BSSID of the airbase.

- Implements the Caffe Latte WEP client attack
- Implements the Hirte WEP client attack
- Ability to cause the WPA/WPA2 handshake to be captured
- Ability to act as an ad-hoc Access Point
- Ability to act as a full Access Point
- Ability to filter by SSID or client MAC addresses
- Ability to manipulate and resend packets
- Ability to encrypt sent packets and decrypt received packets

Usage:

usage: airbase-ng <options> <replay interface>

Options

-a bssid : set Access Point MAC address

-i iface : capture packets from this interface

-w WEP key : use this WEP key to encrypt/decrypt packets

-h MAC : source mac for MITM mode

-f disallow : disallow specified client MACs (default: allow)

-W 0|1 : [don't] set WEP flag in beacons 0|1 (default: auto)

-q : quiet (do not print statistics)

-v : verbose (print more messages) (long --verbose)

-M : M-I-T-M between [specified] clients and bssids (NOT CURRENTLY IMPLEMENTED)

-A : Ad-Hoc Mode (allows other clients to peer) (long --ad-hoc)

-Y in|out|both : external packet processing

-c channel : sets the channel the AP is running on

-X : hidden ESSID (long --hidden)

-s : force shared key authentication

-S : set shared key challenge length (default: 128)

-L : Caffe-Latte attack (long --caffe-latte)

-N : Hirte attack (cfrag attack), creates arp request against wep client (long –cfrag)

-x nbpps : number of packets per second (default: 100)

-y : disables responses to broadcast probes

-0 : set all WPA,WEP,open tags. can't be used with -z & -Z

-z type : sets WPA1 tags. 1=WEP40 2=TKIP 3=WRAP 4=CCMP 5=WEP104

-Z type : same as -z, but for WPA2

-V type : fake EAPOL 1=MD5 2=SHA1 3=auto

-F prefix : write all sent and received frames into pcap file

-P : respond to all probes, even when specifying ESSIDs

-I interval : sets the beacon interval value in ms

-C seconds : enables beaconing of probed ESSID values (requires -P)

Filter options:

 --bssid <MAC> : BSSID to filter/use (short -b)

 --bssids <file> : read a list of BSSIDs out of that file (short -B)

 --client <MAC> : MAC of client to accept (short -d)

 --clients <file> : read a list of MACs out of that file (short -D)

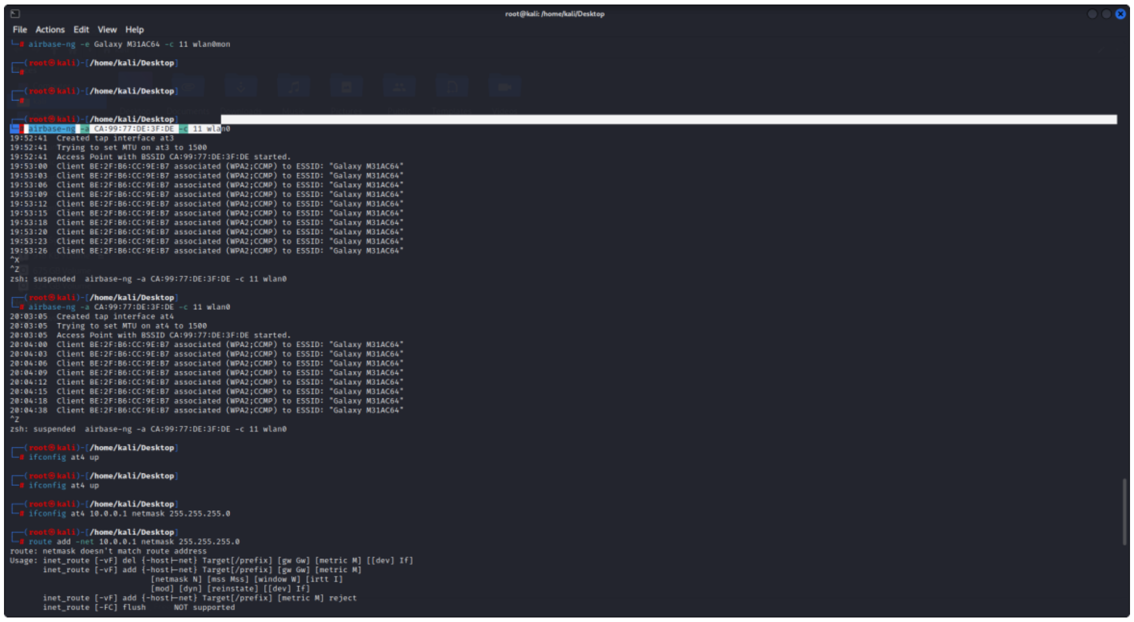
 --essid <ESSID> : specify a single ESSID (short -e)

 --essids <file> : read a list of ESSIDs out of that file (short -E)

Help:

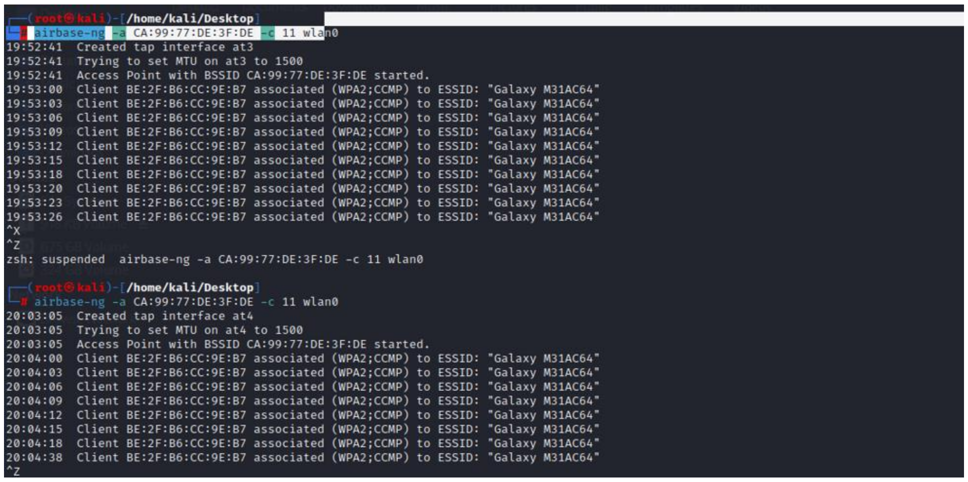
 --help: Displays the usage screen (short -H)

Test case 1:



```
└─# airbase-ng -e Galaxy M31AC64 -c 11 wlan0mon

┌──(root㉿kali)-[/home/kali/Desktop]
└─#

┌──(root㉿kali)-[/home/kali/Desktop]
└─#

┌──(root㉿kali)-[/home/kali/Desktop]
└─# airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0
19:52:41  Created tap interface at3
19:52:41  Trying to set MTU on at3 to 1500
19:52:41  Access Point with BSSID CA:99:77:DE:3F:DE started.
19:53:00  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:03  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:06  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:09  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:12  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:15  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:18  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:20  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:23  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:26  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
^X
^Z
zsh: suspended  airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0

┌──(root㉿kali)-[/home/kali/Desktop]
└─# airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0
20:03:05  Created tap interface at4
20:03:05  Trying to set MTU on at4 to 1500
20:03:05  Access Point with BSSID CA:99:77:DE:3F:DE started.
20:04:00  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:03  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:06  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:09  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:12  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:15  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:18  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:38  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
^Z
zsh: suspended  airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0

┌──(root㉿kali)-[/home/kali/Desktop]
└─# ifconfig at4 up

┌──(root㉿kali)-[/home/kali/Desktop]
└─# ifconfig at4 up

┌──(root㉿kali)-[/home/kali/Desktop]
└─# ifconfig at4 10.0.0.1 netmask 255.255.255.0

┌──(root㉿kali)-[/home/kali/Desktop]
└─# route add -net 10.0.0.1 netmask 255.255.255.0
route: netmask doesn't match route address
Usage: inet_route [-vF] del {-host|-net} Target[/prefix] [gw Gw] [metric M] [[dev] If]
       inet_route [-vF] add {-host|-net} Target[/prefix] [gw Gw] [metric M]
                              [netmask N] [mss Mss] [window W] [irtt I]
                              [mod] [dyn] [reinstate] [[dev] If]
       inet_route [-vF] add {-host|-net} Target[/prefix] [metric M] reject
       inet_route [-FC] flush      NOT supported
```

Test case 2



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0
19:52:41  Created tap interface at3
19:52:41  Trying to set MTU on at3 to 1500
19:52:41  Access Point with BSSID CA:99:77:DE:3F:DE started.
19:53:00  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:03  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:06  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:09  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:12  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:15  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:18  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:20  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:23  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
19:53:26  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
^X
^Z
zsh: suspended  airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0

┌──(root㉿kali)-[/home/kali/Desktop]
└─# airbase-ng -a CA:99:77:DE:3F:DE -c 11 wlan0
20:03:05  Created tap interface at4
20:03:05  Trying to set MTU on at4 to 1500
20:03:05  Access Point with BSSID CA:99:77:DE:3F:DE started.
20:04:00  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:03  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:06  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:09  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:12  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:15  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:18  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
20:04:38  Client BE:2F:B6:CC:9E:B7 associated (WPA2;CCMP) to ESSID: "Galaxy M31AC64"
^Z
```

4. Aircrack-ng

A detector, packet sniffer, WEP and WPA/WPA2-PSK cracker, and analysis software are all included in the network software package known as Aircrack-ng for 802.11 wireless LANs. Aircrack-ng is a fork of the original Aircrack project.

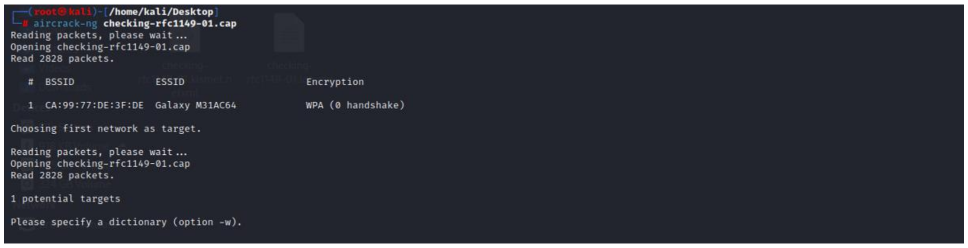
The Aircrack-ng toolkit searches Wi-Fi networks for security weaknesses. You may monitor WiFi security, gather data packets, and export them to text files for more in-depth analysis. To check the functionality of WiFi cards, it is feasible to capture and inject them. When cracking WEP and WPA/WPA2-PSK keys, Aircrack-ng employs a variety of methods. A complete set of tools for evaluating WiFi network security is called Aircrack-ng.

It focuses on different areas of WiFi security:

- Monitoring: Packet capture and export of data to text files for further processing by third party tools.
- Attacking: Replay attacks, deauthentication, fake access points and others via packet injection.
- Testing: Checking WiFi cards and driver capabilities (capture and injection).
- Cracking: WEP and WPA PSK (WPA 1 and 2).

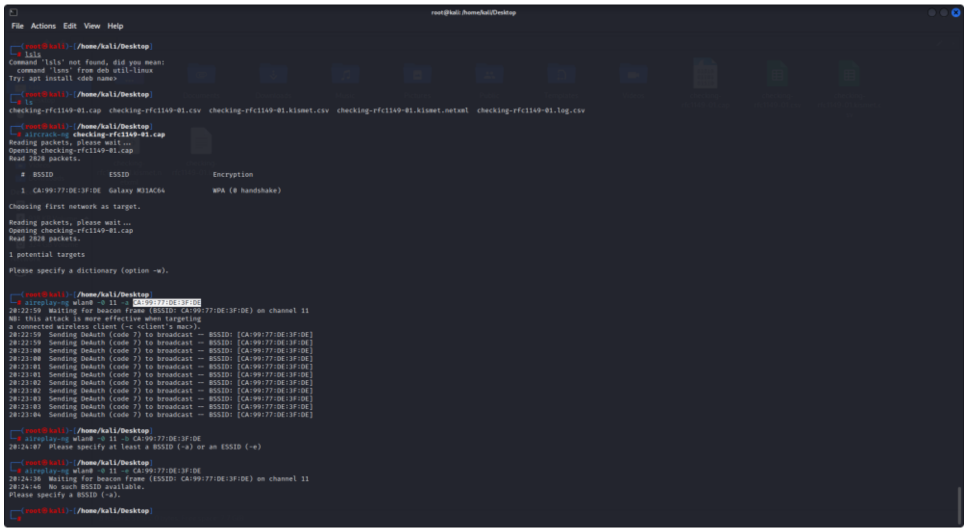
It works with any wireless network interface controller whose driver can sniff 802.11a, 802.11b, and 802.11g traffic and provides raw monitoring mode. The application is compatible with Linux, FreeBSD, macOS, OpenBSD, and Windows. The Linux version has also been transferred to the Android, Zaurus PDA, and Maemo platforms, and an iPhone proof of concept port has been produced.

Test case 1



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# aircrack-ng checking-rfc1149-01.cap
Reading packets, please wait ...
Opening checking-rfc1149-01.cap
Read 2828 packets.

   #  BSSID              ESSID                     Encryption

   1  CA:99:77:DE:3F:DE  Galaxy M31AC64            WPA (0 handshake)

Choosing first network as target.

Reading packets, please wait ...
Opening checking-rfc1149-01.cap
Read 2828 packets.

1 potential targets

Please specify a dictionary (option -w).
```

Test case 2



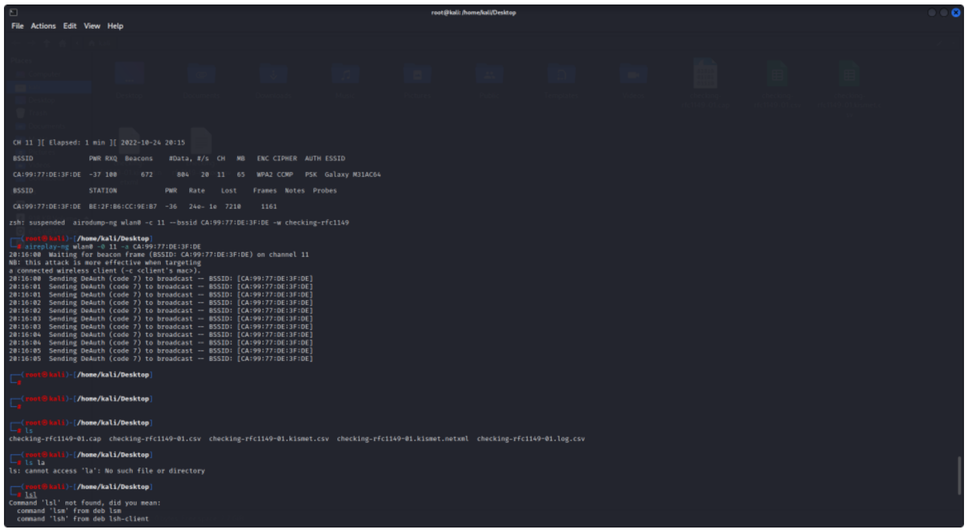
```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# lsls
Command 'lsls' not found, did you mean:
  command 'lsns' from deb util-linux
Try: apt install <deb name>

┌──(root㉿kali)-[/home/kali/Desktop]
└─# ls
checking-rfc1149-01.cap  checking-rfc1149-01.csv  checking-rfc1149-01.kismet.csv  checking-rfc1149-01.kismet.netxml  checking-rfc1149-01.log.csv

┌──(root㉿kali)-[/home/kali/Desktop]
└─# aircrack-ng checking-rfc1149-01.cap
Reading packets, please wait ...
Opening checking-rfc1149-01.cap
Read 2828 packets.

   #  BSSID              ESSID                     Encryption

   1  CA:99:77:DE:3F:DE  Galaxy M31AC64            WPA (0 handshake)

Choosing first network as target.

Reading packets, please wait ...
Opening checking-rfc1149-01.cap
Read 2828 packets.

1 potential targets

Please specify a dictionary (option -w).


┌──(root㉿kali)-[/home/kali/Desktop]
└─# aireplay-ng wlan0 -0 11 -a CA:99:77:DE:3F:DE
20:22:59  Waiting for beacon frame (BSSID: CA:99:77:DE:3F:DE) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
20:22:59  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:22:59  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:00  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:00  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:01  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:01  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:02  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:02  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:03  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:03  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:23:04  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]

┌──(root㉿kali)-[/home/kali/Desktop]
└─# aireplay-ng wlan0 -0 11 -b CA:99:77:DE:3F:DE
20:24:07  Please specify at least a BSSID (-a) or an ESSID (-e)

┌──(root㉿kali)-[/home/kali/Desktop]
└─# aireplay-ng wlan0 -0 11 -e CA:99:77:DE:3F:DE
20:24:36  Waiting for beacon frame (ESSID: CA:99:77:DE:3F:DE) on channel 11
20:24:46  No such BSSID available.
Please specify a BSSID (-a).

┌──(root㉿kali)-[/home/kali/Desktop]
└─#
```

5. Aireplay-ng

It uses Aireplay-ng to inject frames. The main purpose is to produce traffic that will subsequently be used by aircrack-ng to decrypt the WEP and WPA-PSK keys. Deauthentication can be caused by a variety of techniques, including forged authentications, interactive packet replay, specially constructed ARP request injection, and ARP-request reinjection. The utility packet forge-ng allows for the creation of arbitrary frames. It is intended to do injection in order to generate fake traffic that can be utilized for WEP cracking. Wireless rogue traffic is produced using Aireplay-ng. To crack WEP and WPA keys, it can be used in conjunction with aircrack-ng. Aireplay-primary NG's function is to inject frames. With aireplay-ng, you can carry out a variety of strong attacks, including the deauthentication attack, which aids in capturing WPA handshake data, and the fake authentication attack, which involves injecting packets into the network access point while pretending to be a legitimate user in order to create and capture new IVs. The following list also includes other kinds of assaults:

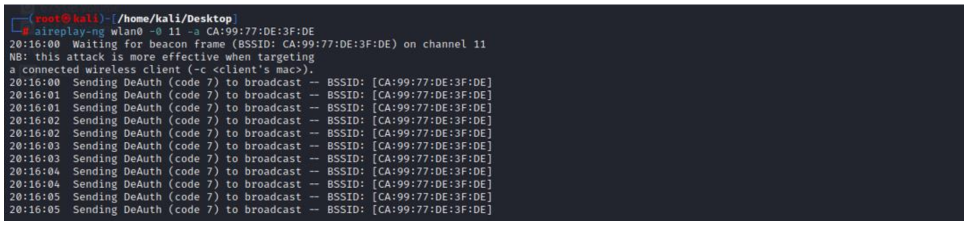
- Interactive packet replay attack
- ARP request replay attack
- KoreK chopchop attack
- Cafe-latte attack
- Fragmentation attack

Test case 1



```
 CH 11 ][ Elapsed: 1 min ][ 2022-10-24 20:15

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

 CA:99:77:DE:3F:DE  -37 100      672      804   20  11   65   WPA2 CCMP   PSK  Galaxy M31AC64

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 CA:99:77:DE:3F:DE  BE:2F:B6:CC:9E:B7  -36   24e- 1e  7210     1161

zsh: suspended  airodump-ng wlan0 -c 11 --bssid CA:99:77:DE:3F:DE -w checking-rfc1149

┌──(root㉿kali)-[/home/kali/Desktop]
└─# aireplay-ng wlan0 -0 11 -a CA:99:77:DE:3F:DE
20:16:00  Waiting for beacon frame (BSSID: CA:99:77:DE:3F:DE) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
20:16:00  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:01  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:01  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:02  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:02  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:03  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:03  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:04  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:04  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:05  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:05  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]

┌──(root㉿kali)-[/home/kali/Desktop]
└─#

┌──(root㉿kali)-[/home/kali/Desktop]
└─#

┌──(root㉿kali)-[/home/kali/Desktop]
└─# ls
checking-rfc1149-01.cap  checking-rfc1149-01.csv  checking-rfc1149-01.kismet.csv  checking-rfc1149-01.kismet.netxml  checking-rfc1149-01.log.csv

┌──(root㉿kali)-[/home/kali/Desktop]
└─# ls la
ls: cannot access 'la': No such file or directory

┌──(root㉿kali)-[/home/kali/Desktop]
└─# lsl
Command 'lsl' not found, did you mean:
  command 'lsm' from deb lsm
  command 'lsh' from deb lsh-client
```

Test case 2



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# aireplay-ng wlan0 -0 11 -a CA:99:77:DE:3F:DE
20:16:00  Waiting for beacon frame (BSSID: CA:99:77:DE:3F:DE) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
20:16:00  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:01  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:01  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:02  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:02  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:03  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:03  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:04  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:04  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:05  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
20:16:05  Sending DeAuth (code 7) to broadcast -- BSSID: [CA:99:77:DE:3F:DE]
```

6. Airolib-ng

To store and manage essid and password lists, generate their Pairwise Master Keys (PMKs), and use them in WPA/WPA2 cracking, use the airolib-ng program from the aircrack-ng package. The application employs the portable SQLite3 database, which is available on most platforms, as its storage method. The administration, memory, and disk overhead, along with platform availability, were all taken into account while choosing the SQLite3 database.

Calculating the pairwise master key, from which the private transient key (PTK) is obtained, is a step in the WPA/WPA2 cracking process. We may calculate the frame message identity code (MIC) for a particular packet using the PTK, and if the MIC matches that of the packet, then the PTK was accurate and the PMK was also correct.

Computing the PMK is still required, yet we can:

- Precompute it for later and/or shared use.

- Use distributed machines to generate the PMK and use their value elsewhere.

OPERATION

--stats

Output information about the database.

--sql

Execute specified SQL statement.

--clean [all]

Clean the database from old junk. When specifying 'all', it will also reduce

filesize if possible and run an integrity check.

--batch

Start batch-processing all combinations of ESSIDs and passwords.

--verify [all]

Verify a set of randomly chosen PMKs. If 'all' is given, all invalid PMK in the

database will be deleted.

--import [essid|passwd]

Import a flat file as a list of ESSIDs or passwords.

import cowpatty

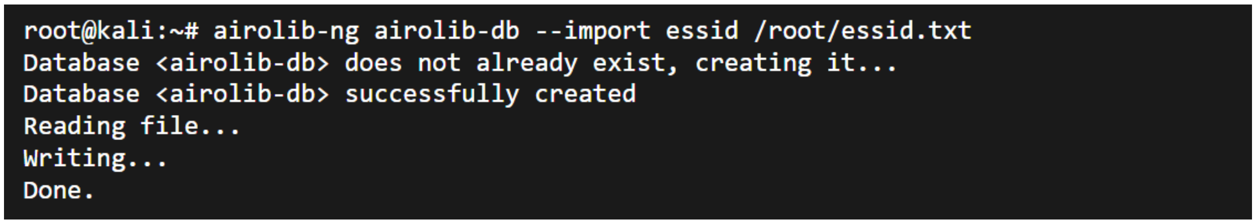
Import a coWPAtty file.

--export cowpatty

Export to a cowpatty file

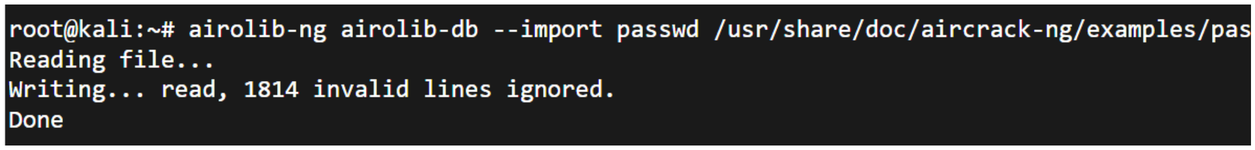
Test case 1:

To import a file containing the ESSIDs of the network(s) you are targeting, use the command line option -import essid /root/essid.txt and provide the database name to use (airolib-db). The database will be created if it doesn't already exist.



```
root@kali:~# airolib-ng airolib-db --import essid /root/essid.txt
Database <airolib-db> does not already exist, creating it...
Database <airolib-db> successfully created
Reading file...
Writing...
Done.
```

Any wordlists you want to utilize to calculate PMK should be imported.



```
root@kali:~# airolib-ng airolib-db --import passwd /usr/share/doc/aircrack-ng/examples/pas
Reading file...
Writing... read, 1814 invalid lines ignored.
Done
```

7. Airserv-ng

A client-server TCP network connection is used by Airserv-ng, a wireless card server, to enable different wireless application programs to independently use a wireless card. The server contains all operating system- and wireless card driver-specific code. Due to this, each wireless application is no longer required to incorporate the intricate wireless card and driver logic. Additionally, it supports a variety of operating systems.

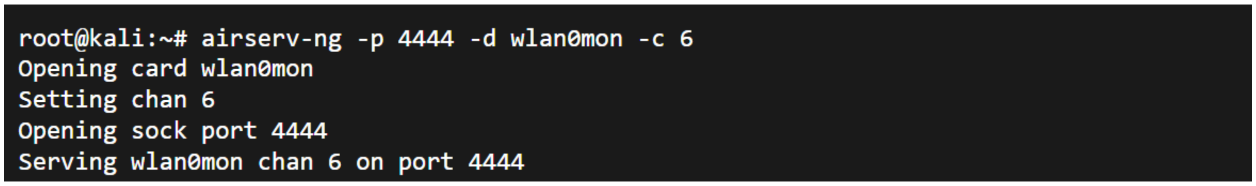
When the server is launched, it waits for client connections on a certain IP address and TCP port. This IP address and port are then used by the wireless application to connect to the server. Instead of utilizing the network interface, you specify "server IP address> colon "port number>" when using the aircrack-ng suite methods. One such instance is 127.0.0.1:666.

This allows for a number of interesting possibilities:

- By eliminating the wireless card/driver complexity, software developers can concentrate on the application functionality. This will lead to a larger set of applications being available. It also dramatically reduces the maintenance effort.
- Remote sensors are now easy to implement. Only a wireless card and airserv-ng are required to be running on the remote sensor. This means that small embedded systems can easily be created.
- You can mix and match operating systems. Each piece can run on a different operating system. The server and each of the applications can potentially run under a different operating system.
- Some wireless cards do not allow multiple applications to access them at once. This constraint is now eliminated with the client-server approach.
- By using TCP networking, the client and server can literally be in different parts of the world. As long as you have network connectivity, then it will work.

Test case :

Start a server instance utilizing the wlan0mon interface on channel 6 at a given port (-p 4444).



```
root@kali:~# airserv-ng -p 4444 -d wlan0mon -c 6
Opening card wlan0mon
Setting chan 6
Opening sock port 4444
Serving wlan0mon chan 6 on port 4444
```

8. Airtun-ng

A virtual tunnel interface designer is airtun-ng. You need the encryption key and the bssid for the network you want to monitor in order to execute wIDS data collection. All of the traffic for that network is decrypted by Airtun-ng before being sent to a conventional IDS system like Snort.

If you have the complete encryption key, traffic injection can be totally bidirectional. If you have the PRGA gained from chopchop or fragmentation assaults, it is outgoing unidirectional. The main benefit of airtun-ng over the other injection tools in the aircrack-ng suite is that you may produce, inject, or sniff packets using any other tool after using airtun-ng.

Additionally, Airtun-ng features repeater and tcpreplay-like features. A wireless device's detected traffic can be replayed using the repeater function, which also offers the ability to filter the traffic using a bssid and network mask before replaying the remaining traffic. The repeater function's interface is given by the -i at0 flag. You can keep using the tun interface while repeating this. Additionally, a pcap file read function enables you to replay packet captures in pcap format exactly as you did when you first collected them. For wifi, this is effectively tcpreplay functionality.

Usage

Usage: airtun-ng

-x nbpps : maximum number of packets per second (optional)

-a bssid : set Access Point MAC address (mandatory). In WDS Mode this sets the

Receiver

-i iface : capture packets from this interface (optional)

-y file : read PRGA from this file (optional / one of -y or -w must be defined)

-w wepkey : use this WEP-KEY to encrypt packets (optional / one of -y or -w must be defined)

-p pass : use this WPA passphrase to decrypt packets (use with -a and -e)

-e essid : target network SSID (use with -p)

-t tods : send frames to AP (1) or to client (0) or tunnel them into a WDS/Bridge (2)

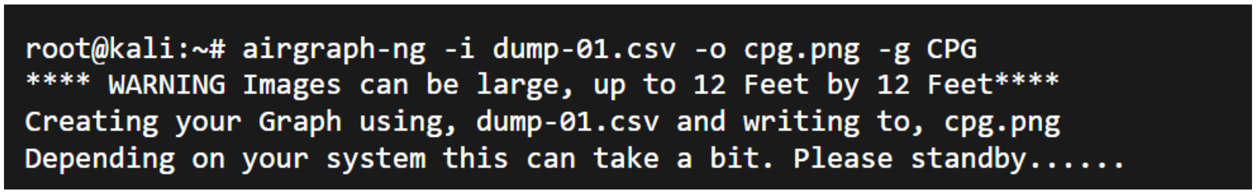
-r file : read frames out of pcap file (optional)

-h MAC : source MAC address

Usage:

CPG chart

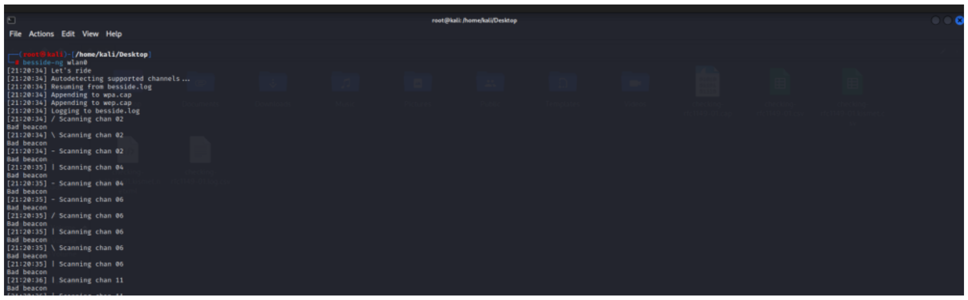
Choose the graph type, the output file to create (-o cpg.png), and the input file to use (-i dump-01.csv) (-g CAG).



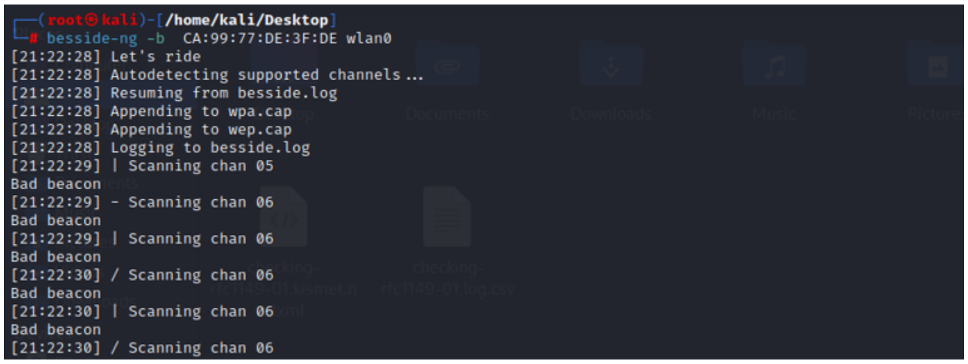
```
root@kali:~# airgraph-ng -i dump-01.csv -o cpg.png -g CPG
**** WARNING Images can be large, up to 12 Feet by 12 Feet****
Creating your Graph using, dump-01.csv and writing to, cpg.png
Depending on your system this can take a bit. Please standby......
```

9. Besside-ng

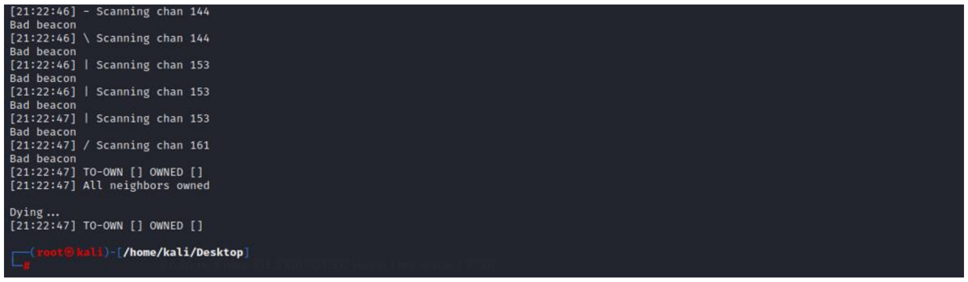
A similar utility to Wesside-ng, Besside-ng also supports WPA encryption. Which will automatically decrypt all nearby WEP networks and record WPA handshakes. In order to try and crack the password, WPA handshakes can be uploaded to the internet cracking service at Darkircop.org (Beside-ng Companion), which also offers helpful statistics based on user-submitted capture files on the viability of WPA cracking. Beside-ng needs the aircrack-ng SVN version and a wireless interface with functional injection.



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# besside-ng wlan0
[21:20:34] Let's ride
[21:20:34] Autodetecting supported channels ...
[21:20:34] Resuming from besside.log
[21:20:34] Appending to wpa.cap
[21:20:34] Appending to wep.cap
[21:20:34] Logging to besside.log
[21:20:34] / Scanning chan 02
Bad beacon
[21:20:34] \ Scanning chan 02
Bad beacon
[21:20:34] - Scanning chan 02
Bad beacon
[21:20:35] | Scanning chan 04
Bad beacon
[21:20:35] - Scanning chan 04
Bad beacon
[21:20:35] - Scanning chan 06
Bad beacon
[21:20:35] / Scanning chan 06
Bad beacon
[21:20:35] | Scanning chan 06
Bad beacon
[21:20:35] \ Scanning chan 06
Bad beacon
[21:20:35] | Scanning chan 06
Bad beacon
[21:20:36] | Scanning chan 11
Bad beacon
```



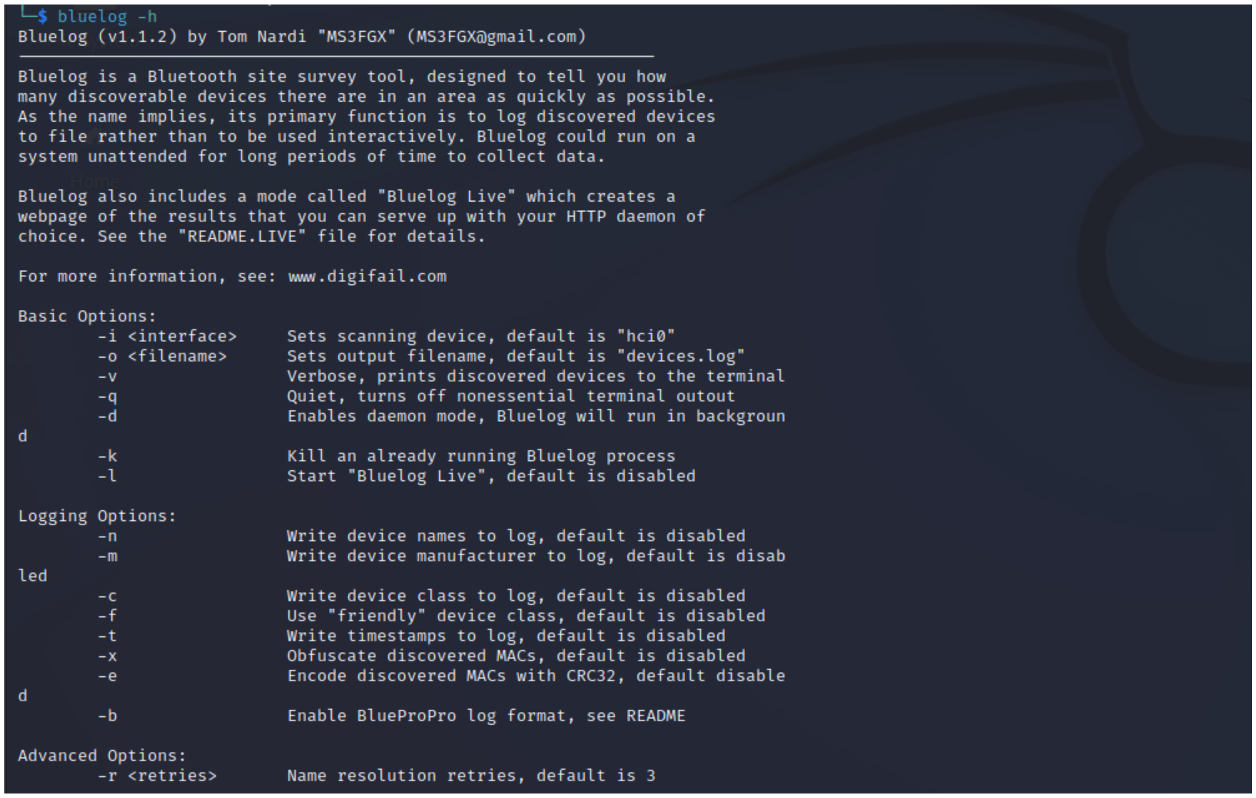
```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# besside-ng -b  CA:99:77:DE:3F:DE wlan0
[21:22:28] Let's ride
[21:22:28] Autodetecting supported channels ...
[21:22:28] Resuming from besside.log
[21:22:28] Appending to wpa.cap
[21:22:28] Appending to wep.cap
[21:22:28] Logging to besside.log
[21:22:29] | Scanning chan 05
Bad beacon
[21:22:29] - Scanning chan 06
Bad beacon
[21:22:29] | Scanning chan 06
Bad beacon
[21:22:30] / Scanning chan 06
Bad beacon
[21:22:30] | Scanning chan 06
Bad beacon
[21:22:30] / Scanning chan 06
```



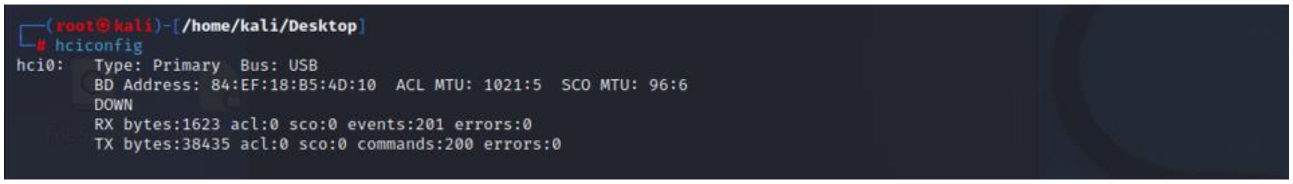
```
[21:22:46] - Scanning chan 144
Bad beacon
[21:22:46] \ Scanning chan 144
Bad beacon
[21:22:46] | Scanning chan 153
Bad beacon
[21:22:46] | Scanning chan 153
Bad beacon
[21:22:47] | Scanning chan 153
Bad beacon
[21:22:47] / Scanning chan 161
Bad beacon
[21:22:47] TO-OWN [] OWNED []
[21:22:47] All neighbors owned

Dying ...
[21:22:47] TO-OWN [] OWNED []

┌──(root㉿kali)-[/home/kali/Desktop]
└─#
```

10. Bluelog

A Bluetooth scanner called Bluelog is made to provide information on the number of discoverable devices in a location as rapidly as feasible. It is designed to be used as a site survey tool to count the potential Bluetooth targets in the immediate vicinity. The sole purpose of the Linux Bluetooth scanner Bluelog is to log devices that are in discoverable mode. It is designed to be used as a site survey tool to count the number of Bluetooth devices that may be found in a specific area.



```
└─$ bluelog -h
Bluelog (v1.1.2) by Tom Nardi "MS3FGX" (MS3FGX@gmail.com)
─────────────────────────────────────────────────────────────
Bluelog is a Bluetooth site survey tool, designed to tell you how
many discoverable devices there are in an area as quickly as possible.
As the name implies, its primary function is to log discovered devices
to file rather than to be used interactively. Bluelog could run on a
system unattended for long periods of time to collect data.

Bluelog also includes a mode called "Bluelog Live" which creates a
webpage of the results that you can serve up with your HTTP daemon of
choice. See the "README.LIVE" file for details.

For more information, see: www.digifail.com

Basic Options:
        -i <interface>     Sets scanning device, default is "hci0"
        -o <filename>      Sets output filename, default is "devices.log"
        -v                 Verbose, prints discovered devices to the terminal
        -q                 Quiet, turns off nonessential terminal outout
        -d                 Enables daemon mode, Bluelog will run in backgroun
d
        -k                 Kill an already running Bluelog process
        -l                 Start "Bluelog Live", default is disabled

Logging Options:
        -n                 Write device names to log, default is disabled
        -m                 Write device manufacturer to log, default is disab
led
        -c                 Write device class to log, default is disabled
        -f                 Use "friendly" device class, default is disabled
        -t                 Write timestamps to log, default is disabled
        -x                 Obfuscate discovered MACs, default is disabled
        -e                 Encode discovered MACs with CRC32, default disable
d
        -b                 Enable BlueProPro log format, see README

Advanced Options:
        -r <retries>       Name resolution retries, default is 3
```



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# hciconfig
hci0:   Type: Primary  Bus: USB
        BD Address: 84:EF:18:B5:4D:10  ACL MTU: 1021:5  SCO MTU: 96:6
        DOWN
        RX bytes:1623 acl:0 sco:0 events:201 errors:0
        TX bytes:38435 acl:0 sco:0 commands:200 errors:0
```

11. BlueMaho

It serves as the user interface (GUI) for a toolkit for evaluating the security of Bluetooth devices. It is open source, freeware, written in Python, and makes use of wxPython. It can be used to test BT devices for known vulnerabilities, however the main task is to test for undiscovered flaws. It can also produce lovely statistics. A freeware open source program called BlueMaho is used to test the security of Bluetooth devices and can be used to identify both known and undiscovered flaws. It was created in Python. Penetration testers frequently use it to assess the security. Martin Herfurt is the author, and plate form is Linux.