Aircrack-ng

Airbase-ng

Description

This documentation is still under development. There is quite a bit more work to be done on this documentation. Please post any comments or suggestions to this thread in the Forum [http://forum.aircrack-ng.org/index.php?topic=3247.0].

Airbase-ng is multi-purpose tool aimed at attacking clients as opposed to the Access Point (AP) itself. Since it is so versatile and flexible, summarizing it is a challenge. Here are some of the feature highlights:

- 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

The main idea is of the implementation is that it should encourage clients to associate with the fake AP, not prevent them from accessing the real AP.

A tap interface (atX) is created when airbase-ng is run. This can be used to receive decrypted packets or to send encrypted packets.

As real clients will most probably send probe requests for common/configured networks, these frames are important for binding a client to our softAP. In this case, the AP will respond to any probe request with a proper probe response, which tells the client to authenticate to the airbase-ng BSSID. That being said, this mode could possibly disrupt the correct functionality of many APs on the same channel.

WARNING: airbase-ng can easily disrupt Access Points around you. Where possible, use filters to minimize this possibility. Always act responsibly and do not disrupt networks which do not belong to you.

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)

-a BSSID Definition

If the BSSID is not explicitly specified by using “-a <BSSID>”, then the current MAC of the specified interface is used.

-i iface

If you specify an interface with this option then packets are also captured and processed from this interface in addition to replay interface.

-w WEP key

If WEP should be used as encryption, then the parameter "-w <WEP key>" sets the en-/decryption key. This is sufficient to let airbase-ng set all the appropriate flags by itself.

If the softAP operates with WEP encryption, the client can choose to use open system authentication or shared key authentication. Both authentication methods are supported by airbase-ng. But to get a keystream, the user can try to force the client to use shared key authentication. "-s" forces a shared key auth and "-S <len>" sets the challenge length.

-h MAC

This is the source MAC for the man-in-the-middle attack. The "-M" must also be specified.

-f allow/disallow

If this option is not specified, it defaults to "-f allow". This means the various client MAC filters (-d and -D) define which clients to accept.

By using the "-f disallow" option, this reverses selection and causes airbase to ignore the clients specified by the filters.

-W WEP Flag

This sets the beacon WEP flag. Remember that clients will normally only connect to APs which are the same as themselves. Meaning WEP to WEP, open to open.

The "auto" option is to allow airbase-ng to automatically set the flag based on context of the other options specified. For example, if you set a WEP key with -w, then the beacon flag would be set to WEP.

One other use of "auto" is to deal with clients which can automatically adjust their connection type. However, these are few and far between.

In practice, it is best to set the value to the type of clients you are dealing with.

-q Quiet Flag

This suppresses printing any statistics or status information.

-v Verbose Flag

This prints additional messages and details to assist in debugging.

-M MITM Attack

This option is not implemented yet. It is a man-in-the-middle attack between specified clients and BSSIDs.

-A Ad-Hoc Mode

This causes airbase-ng to act as an ad-hoc client instead of a normal Access Point.

In ad-hoc mode airbase-ng also sends beacons, but doesn't need any authentication/association. It can be activated by using “-A”. The soft AP will adjust all flags needed to simulate a station in ad-hoc mode automatically and generate a random MAC, which is used as CELL MAC instead of the BSSID. This can be overwritten by the “-a <BSSID>” tag. The interface MAC will then be used as source mac, which can be changed with “-h <sourceMAC>”.

-Y External Processing

The parameter “-Y” enables the “external processing” Mode. This creates a second interface “atX”, which is used to replay/modify/drop or inject packets at will. This interface must also be brought up with ifconfig and an external tool is needed to create a loop on that interface.

The packet structure is rather simple: the ethernet header (14 bytes) is ignored and right after that follows the complete ieee80211 frame the same way it is going to be processed by airbase-ng (for incoming packets) or before the packets will be sent out of the wireless card (outgoing packets). This mode intercepts all data packets and loops them through an external application, which decides what happens with them. The MAC and IP of the second tap interface doesn't matter, as real ethernet frames on this interface are dropped dropped anyway.

There are 3 arguments for “-Y”: “in”, “out” and “both”, which specify the direction of frames to loop through the external application. Obviously “in” redirects only incoming (through the wireless NIC) frames, while outgoing frames aren't touched. “out” does the opposite, it only loops outgoing packets and “both” sends all both directions through the second tap interface.

There is a small and simple example application to replay all frames on the second interface. The tool is called “replay.py” and is located in “./test”. It's written in python, but the language doesn't matter. It uses pcapy to read the frames and scapy to potentially alter/show and reinject the frames. The tool as it is, simply replays all frames and prints a short summary of the received frames. The variable “packet” contains the complete ieee80211 packet, which can easily be dissected and modified using scapy.

This can be compared to ettercap filters, but is more powerful, as a real programming language can be used to build complex logic for filtering and packet customization. The downside on using python is, that it adds a delay of around 100ms and the cpu utilization is rather large on a high speed network, but its perfect for a demonstration with only a few lines of code.

-c Channel Flag

This is used to specify the channel on which to run the Access Point.

-X Hidden SSID Flag

This causes the Access Point to hide the SSID and to not broadcast the value.

-s Force Shared Key Authentication

When specified, this forces shared key authentication for all clients.

The soft AP will send an “authentication method unsupported” rejection to any open system authentication request if “-s” is specified.

-S Shared Key Challenge Length

“-S <len>” sets the challenge length, which can be anything from 16 to 1480. The default is 128 bytes. It is the number of bytes used in the random challenge. Since one tag can contain a maximum size of 255 bytes, any value above 255 creates several challenge tags until all specified bytes are written. Many clients ignore values different than 128 bytes so this option may not always work.

-L Caffe Latte Attack

Airbase-ng also contains the new caffe-latte attack, which is also implemented in aireplay-ng as attack “-6”. It can be used with “-L” or “–caffe-latte”. This attack specifically works against clients, as it waits for a broadcast arp request, which happens to be a gratuitous arp. See this [http://wiki.wireshark.org/Gratuitous_ARP] for an explanation of what a gratuitous arp [http://wiki.wireshark.org/Gratuitous_ARP] is. It then flips a few bits in the sender MAC and IP, corrects the ICV (crc32) value and sends it back to the client, where it came from. The point why this attack works in practice is, that at least windows sends gratuitous arps after a connection on layer 2 is established and a static ip is set, or dhcp fails and windows assigned an IP out of 169.254.X.X.

“-x <pps>” sets the number of packets per second to send when performing the caffe-latte attack. At the moment, this attack doesn't stop, it continuously sends arp requests. Airodump-ng is needed to capture the replys.

-N Hirte Attack (Fragmentation Attack)

This attack listens for an ARP request or IP packet from the client. Once one is received, a small amount of PRGA is extracted and then used to create an ARP request packet targeted to the client. This ARP request is actually made of up of multiple packet fragments such that when received, the client will respond.

This attack works especially well against ad-hoc networks. As well, it can be used against softAP clients and normal AP clients.

This option includes added compatibility with some clients. As well, random source IPs and MACs for cfrag attack are included to evade simple flood protection.

-x Number of Packets per Second

This sets the number of packets per second transmission rate (default: 100).

-y Disable Broadcast Probes

When using this option, the fake AP will not respond to broadcast probes. A broadcast probe is where the

specific AP is not identified uniquely. Typically, most APs will respond with probe responses to a broadcast probe. This flag will prevent this happening. It will only respond when the specific AP is uniquely requested.

-0 Set WPA/WEP Tags

This enables all WPA/WPA2/WEP Tags to be enabled in the beacons sent. It cannot be specified when also using -z or -Z

-z Set WPA Tag

This specifies the WPA beacon tags. The valid values are: 1=WEP40 2=TKIP 3=WRAP 4=CCMP 5=WEP104. It is recommended that you also set the WEP flag in the beacon with "-W 1" when using this parameter since some clients get confused without it.

-Z Set WPA2 Tag

This specifies the WPA2 beacon tags. The valid values are the same as WPA. It is recommended that you also set the WEP flag in the beacon with "-W 1" when using this parameter since some clients get confused without it.

-V EAPOL Type

This specifies the valid EAPOL types. The valid values are: 1=MD5 2=SHA1 3=auto

-F File Name Prefix

This option causes airbase-ng to write all sent and received packets to a pcap file on disk. This is the file prefix (like airodump-ng -w).

-P All Probes

This causes the fake access point to respond to all probes regardless of the ESSIDs specified. Without -P, the old behavior of ignoring probes for non-matching ESSIDs will be used.

-I Beacon Interval

This sets the time in milliseconds between beacons being sent.

When using a list of ESSIDs, all ESSIDs will be broadcast with beacons. As extra ESSIDs are added, the beacon interval value is now adjusted based on the number of ESSIDs times the interval value (0x64 is default still). To support "fast" beaconing of a long list of ESSIDs, the -I parameter can be used to set a smaller interval. To get 0x64 interval for N beacons, set the -I parameter to 0x64/N. If this value goes below ~10 or so, the maximum injection rate will be reached and airbase-ng will not be able to reliable handle new clients. Since each card's injection rates are different, the -I parameters allows it to be tuned to a specific setup and injection speed based on the number of beacons.

-C Seconds

The -P option must also be specified in order to use this option. The wildcard ESSIDs will also be beaconed this number of seconds. A good typical value to use is "-C 60".

When running in the default mode (no ESSIDs) or with the -P parameter, the -C option can be used to enable beacon broadcasting of the ESSIDs seen by the directed probes. This allows one client which is probing for a network to result in a beacon for the same network for a brief period of time (the -C parameter, which is the number of seconds to broadcast new probe requests). This works well when some clients are sending directed probes, while others listen passively for beacons. A client which does directed probes results in a beacon which wakes up the passive client and causes the passive client to join the network as well. This is especially useful with Vista clients (which listens passively for beacons in many cases) which share the same WiFi? network as Linux/Mac OS X clients which send directed probes.

Beacon Frames

The beacon frame contains the ESSID in case exactly one ESSID is specified, if several are set, the ESSID will be hidden in the beacon frame with a length of 1. If no ESSID is set, the beacon will contain "default" as ESSID, but accept all ESSIDs in association requests. If the ESSID should be hidden in the beacon frame all the time (read: for no or one specified ESSID), the "-X" flag can be set.

Control Frame Handling

Control frames (ack/rts/cts) are never sent by the code, but sometimes read (the firmware should handle that). Management and data frames can always be sent, no need to authenticate before association or even sending of data frames. They can be sent right away. Real clients will still authenticate and associate and the softAP should send the correct answers, but airbase-ng doesn't care to check the properties and simply allows all stations to connect (with respect to the filtered ESSIDs and client MACs). So an authentication cannot fail (except if SKA is forced). Same for the association phase. The AP will never send deauthentication or disassociation frames on normal operation mode.

It has been implemented in a way to maximizes the compatibility and the chances to keep a station connected.

Filtering

There are rich filtering capabilities.

To limit the supported ESSIDs, you can specify "-e <ESSID>" to add an ESSID to the list of allowed ESSIDs, or use "-E <ESSIDfile>" to read a list of allowed ESSIDs out of this file (one ESSID per line).

The same can be done for client MACs (sort of a MAC-filter). "-d <MAC>" adds a single MAC to the list, "-D <MACfile>" adds all MACs out of the <MACfile> to that list (again, one MAC per line).

The MAC list can be used to allow only the clients on this list and block all others (default), or to block the specified ones and allow all others. This is controlled by "-f allow" or "-f disallow". "allow" creates a whitelist (default in case "-f" is not set), whereas "disallow" a blacklist builds (the second case).

Tap Interface

Each time airbase is run, a tap interface (atX) is created. To use it, run "ifconfig atX up" where X is the actual

interface number.

This interface has many uses:

- If an encryption key is specified with "-w", then incoming packets will be decrypted and presented on the interface.
- Packets sent to this interface will be transmitted. Additionally, they will be encrypted if the "-w" option is used.

Usage Examples

Here are usage examples. You only require a single wireless device even though two cards were used in some of the examples.

Simple

Using "airbase-ng <iface>" is enough to setup an AP without any encryption. It will accept connections from any MAC for every ESSID, as long as the authentication and association is directed to the BSSID.

You really cannot do much in this scenario. However, it will present a list of clients which are connecting plus the encryption method and the SSIDs.

Hirte Attack in Access Point mode

This attack obtains the wep key from a client. It depends on receiving at least one ARP request or IP packet from the client after it has associated with the fake AP.

Enter:

```
airbase-ng -c 9 -e teddy -N -W 1 rausb0
```

Where:

- -c 9 specifies the channel
- -e teddy filters a single SSID
- -N specifies the Hirte attack
- -W 1 forces the beacons to specify WEP
- rausb0 specifies the wireless interface to use

The system responds:

```
18:57:54  Created tap interface at0
18:57:55  Client 00:0F:B5:AB:CB:9D associated (WEP) to ESSID: "teddy"
```

On another console window run:

```
airodump-ng -c 9 -d 00:06:62:F8:1E:2C -w cfrag wlan0
```

Where:

- -c 9 specifies the channel
- -d 00:06:62:F8:1E:2C filters the data captured to fake AP MAC (this is optional)
- -w specifies the file name prefix of the captured data
- wlan0 specifies the wireless interface to capture data on

Here is what the window looks like when airbase-ng has received a packet from the client and has successfully started the attack:

```
CH  9 ][ Elapsed: 8 mins ][ 2008-03-20 19:06

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

 00:06:62:F8:1E:2C  100  29      970    14398   33   9  54  WEP  WEP         teddy

 BSSID              STATION            PWR   Rate  Lost  Packets  Probes

 00:06:62:F8:1E:2C  00:0F:B5:AB:CB:9D   89   2-48     0   134362
```

At this point you can start aircrack-ng in another console window to obtain the wep key. Alternatively, use the "-F <file name prefix> option with airbase-ng to directly write a capture file instead of using airodump-ng.

Hirte Attack in Ad-Hoc mode

This attack obtains the wep key from a client. It depends on receiving at least one ARP request or IP packet from the client after it has associated with the fake AP.

Enter:

```
airbase-ng -c 9 -e teddy -N -W 1 -A rausb0
```

Where:

- -c 9 specifies the channel
- -e teddy filters a single SSID
- -N specifies the Hirte attack
- -W 1 forces the beacons to specify WEP
- -A specifies ad-hoc mode
- rausb0 specifies the wireless interface to use

The rest will be the same as the AP mode.

Caffe Latte Attack in Access Point mode

This attack obtains the WEP key from a client. It depends on receiving at least one gratuitous ARP request from the client after it has associated with the fake AP.

Enter:

```
airbase-ng -c 9 -e teddy -L -W 1 rausb0
```

Where:

- -c 9 specifies the channel

- -e teddy filters a single SSID
- -L specifies the Caffe Latte attack
- -W 1 forces the beacons to specify WEP
- rausb0 specifies the wireless interface to use

The rest is the same as the Hirte client attack.

Shared Key Capture

This is an example of capturing the PRGA from a client shared key association.

Enter:

```
airbase-ng -c 9 -e teddy -s -W 1 wlan0
```

Where:

- -c 9 specifies the channel
- -e teddy filters a single SSID
- -s forces shared key authentication
- -W 1 forces the beacons to specify WEP
- wlan0 specifies the wireless interface to use

The system responds:

```
15:08:31  Created tap interface at0
15:13:38  Got 140 bytes keystream: 00:0F:B5:88:AC:82
15:13:38  SKA from 00:0F:B5:88:AC:82
15:13:38  Client 00:0F:B5:88:AC:82 associated to ESSID: "teddy"
```

The last three lines only appear when the client associates with the fake AP.

On another console window run:

```
airodump-ng -c 9 wlan0
```

Where:

- -c 9 specifies the channel
- wlan0 specifies the wireless interface to use

Here is what the window looks like with a successful SKA capture. Notice “140 bytes keystream: 00:C0:CA:19:F9:65” in the top right-hand corner:

```
CH  9 ][ Elapsed: 9 mins ][ 2008-03-12 15:13 ][ 140 bytes keystream: 00:C0:CA:19:F9:65

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

 00:C0:CA:19:F9:65   87  92     5310        0    0   9  54  WEP  WEP    SKA  teddy

 BSSID              STATION            PWR   Rate  Lost  Packets  Probes

 00:C0:CA:19:F9:65  00:0F:B5:88:AC:82   83   0- 1     0     4096  teddy
```

Alternatively, use the ”-F <file name prefix> option with airbase-ng to directly write a capture file instead of

using airodump-ng.

WPA Handshake Capture

This is an example of how to capture the WPA handshake.

Enter:

```
airbase-ng -c 9 -e teddy -z 2 -W 1 rausb0
```

Where:

- -c 9 specifies the channel
- -e teddy filters a single SSID
- -z 2 specifies TKIP
- -W 1 set WEP flag because some clients get confused without it.
- rausb0 specifies the wireless interface to use

The -z type will have to be changed depending on the cipher you believe the client will be using. TKIP is typical for WPA.

The system responds:

```
10:17:24  Created tap interface at0
10:22:13  Client 00:0F:B5:AB:CB:9D associated (WPA1;TKIP) to ESSID: "teddy"
```

The last line only appears when the client associates.

On another console window run:

```
airodump-ng -c 9 -d 00:C0:C6:94:F4:87 -w cfrag wlan0
```

- -c 9 specifies the channel
- -d 00:C0:C6:94:F4:87 filters the data captured to fake AP MAC. It is MAC of card running the fake AP. This is optional.
- -w specifies the file name of the captured data
- wlan0 specifies the wireless interface to capture data on

When the client connects, notice the “WPA handshake: 00:C0:C6:94:F4:87” in the top right-hand corner of the screen below:

```
CH  9 ][ Elapsed: 5 mins ][ 2008-03-21 10:26 ][ WPA handshake: 00:C0:C6:94:F4:87

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

00:C0:C6:94:F4:87  100  70     1602       14    0   9  54  WPA  TKIP   PSK  teddy

BSSID              STATION            PWR   Rate  Lost  Packets  Probes

00:C0:C6:94:F4:87  00:0F:B5:AB:CB:9D   86   2- 1     0       75
```

Alternatively, use the “-F <file name prefix> option with airbase-ng to directly write a capture file instead of using airodump-ng.

Running “aircrack-ng cfrag-01.cap” proves it captured a valid WPA handshake:

```
Opening cfrag-01.cap
Read 114392 packets.

#  BSSID              ESSID                     Encryption

1  00:C0:C6:94:F4:87  teddy                     WPA (1 handshake)
```

WPA2 Handshake Capture

Capturing a WPA2 is basically identical to the example above regarding WPA. The typical difference is to specify -Z 4 (CCMP cipher) instead of -z 2.

Enter:

```
airbase-ng -c 9 -e teddy -Z 4 -W 1 rausb0
```

The balance is the same as the WPA handshake capture.

softAP

GURU EXPERTS ONLY: This functionality requires extremely advanced linux and networking knowledge. Do not post questions to the forum regarding this section. If you cannot debug this functionality on your own then you should not be using it!

A new tap interface “atX” will be created, which acts as the “wired side” to the AP. In order to use the AP, this new interface must be brought up with ifconfig and needs an IP. The assigned MAC is automatically set to the BSSID [by default the wireless interface MAC]. Once an IP is assigned and the client uses a static IP out of the same subnet, there is a working Ethernet connection between the AP and the client. Any daemon can be assigned to that interface, for example a dhcp and dns server. Together with kernel ip_forwarding and a proper iptable rule for masquerading, the softAP acts as a wireless router. Any tool, which operates on ethernet can be bound to this interface.

This forum posting [http://forum.aircrack-ng.org/index.php?topic=3983.msg23110#msg23110] provides an example of the commands needed to setup the softAP. This forum posting [http://forum.aircrack-ng.org/index.php?topic=4495.msg25342#msg25342] provides IPTables troubleshooting tip.

Here are some links that may find useful in getting bridging operational. In the madwifi-project.org one, just use at0 where ath0 is referenced.

- http://madwifi-project.org/wiki/UserDocs/TransparentBridge [http://madwifi-project.org/wiki/UserDocs/TransparentBridge]
- http://wiki.ubuntuusers.de/WLAN/MadWifi#Einfache-Methode [http://wiki.ubuntuusers.de/WLAN/MadWifi#Einfache-Methode]

Usage Tips

How Does the Caffe Latte Attack Work?

Here are some links:

- Cafe Latte attack [http://www.airtightnetworks.net/knowledgecenter/wep-caffelatte.html]
- The Caffe Latte Attack: How It Works—and How to Block It [http://www.esecurityplanet.com/prevention/article.php/3716656]

In addition to the descriptions above, airbase-ng sends the last 100 packets 100 times to attempt to increase the effectiveness of the attack.

How Does the Hirte Attack Work?

This is a client attack which can use any IP or ARP packet. The following describes the attack in detail.

The basic idea is to generate an ARP request to be sent back to the client such that the client responds.

The attack needs either an ARP or IP packet from the client. From this, we need to generate an ARP request. The ARP request must have the target IP (client IP) at byte position 33 and the target MAC should be all zeroes. However the target MAC can really be any value in practice.

The source IP is in the packet received from the client is in a known position - position 23 for ARP or 21 for IP. ARP is assumed if the packet is 68 or 86 bytes in length plus a broadcast destination MAC address. Otherwise it is assumed to be an IP packet.

In order to send a valid ARP request back to the client, we need to move the source IP to position 33. Of course you can't simply move bytes around, that would invalidate the packet. So instead, we use the concept of packet fragmentation to achieve this. The ARP request is sent to the client as two fragments. The first fragment length is selected such that the incoming source IP is moved to position 33 when the fragments are ultimately reassembled by the client. The second fragment is the original packet received from the client.

In the case of an IP packet, a similar technique is used. However due to the more limited amount of PRGA available, there are three fragments plus the original packet used.

In all cases, bit flipping is used to ensure the CRC is correct. Additionally, bit flipping is used to ensure the source MAC of the ARP contained within the fragmented packet is not multicast.

SoftAP with Internet connection and MITM sniffing

This forum thread [http://forum.aircrack-ng.org/index.php?topic=7172.0] provides a tutorial for SoftAP with Internet connection and MITM sniffing.

Usage Troubleshooting

Driver Limitations

Some drivers like r8187 don't capture packets transmitted by itself. The implication of this is that the softAP will not show up in airodump-ng. You can get around this by using two wireless cards, one to inject and one to capture. Alternatively, you can use the rtl8187 driver.

The madwifi-ng currently does not support the Caffe-Latte or Hirte attacks. The root cause is deep within the madwifi-ng driver. The driver does not properly synchronize speeds with the client and thus the client never receives the packets. If you need to use these attacks, try using the ath5k driver.

Broken SKA error message

You receive “Broken SKA: <MAC address> (expected: ??, got ?? bytes)” or similar. When using the ”-S“ option with values different then 128, some clients fail. This message indicates the number of bytes actually received was different that the number requested. Either don't use the option or try different values of ”-S“ to see which one eliminates the error.

"write failed: Message too long" / "wi_write(): Illegal seek" error messages

See this GitHub issue [https://github.com/aircrack-ng/aircrack-ng/issues/469] for a workaround. The issue explains the root cause and how to adjust the MTU to avoid the problem.

Error creating tap interface: Permission denied

See the following FAQ entry.

Related Commands

”-D“ is a new option that has been added to aireplay-ng. By default, aireplay-ng listens for beacons from the specified AP and fails if it does not hear any beacons. The ”-D“ option disables this requirement.

aireplay-ng -6 (Cafe Latte attack)

Example: aireplay-ng -6 -h 00:0E:D2:8D:7D:0A -D rausb0

aireplay-ng -7 (Hirte attack)

Example: aireplay-ng -7 -h 00:0E:D2:8D:7D:0A -D rausb0
