NAME

airbase-ng - multi-purpose tool aimed at attacking clients as opposed to the Access Point (AP) itself

SYNOPSIS

airbase-ng [options] <interface name>

DESCRIPTION

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.

OPTIONS

-H, --help

Shows the help screen.

-a <bssid>

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

-i <iface>

Also capture and process from this interface in addition to the 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 <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-ng to ignore the clients specified by the filters.

-W <0|1>
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 This suppresses printing any statistics or status information.

-v This prints additional messages and details to assist in debugging.

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

-A, --ad-hoc
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 <in|out|both>
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 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 possibly 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 utilizations is rather large on a high speed network, but its perfect for a demonstration with only a few lines of code.

-c <channel>
This is used to specify the channel on which to run the Access Point.

-X, --hidden
This causes the Access Point to hide the SSID and to not broadcast the value.

-s When specfiied, 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 It sets the shared key 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
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 for an explanation of what a 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 replies.

-N, --cfrag
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.

-x <nbpps>
This sets the number of packets per second that packets will be sent (default: 100).

-y 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 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 <type>
This specifies the WPA beacon tags. The valid values are: 1=WEP40 2=TKIP 3=WRAP 4=CCMP 5=WEP104.

-Z <type>
same as -z, but for WPA2

-V <type>
This specifies the valid EAPOL types. The valid values are: 1=MD5 2=SHA1 3=auto

-F <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
: This causes the fake access point to respond to all probes regardless of the ESSIDs specified.

-I <interval>
: This sets the time in milliseconds between each beacon.

-C <seconds>
: The wildcard ESSIDs will also be beaconed this number of seconds. A good typical value to use is "-C 60" (require -P).

-n <hex>
: ANonce (nonce from the AP) to use instead of a randomized one. It must be 64 hexadecimal characters.

Filter options:

--bssid <MAC>, -b <MAC>
: BSSID to filter/use.

--bssids <file>, -B <file>
: Read a list of BSSIDs out of that file.

--client <MAC>, -d <MAC>
: MAC of client to accept.

--clients <file>, -D <file>
: Read a list of client's MACs out of that file.

--essid <ESSID>, -e <ESSID>
: Specify a single ESSID. For SSID containing special characters, see https://www.aircrack-ng.org/doku.php?id=faq#how_to_use_spaces_double_quote_and_single_quote_etc_in_ap_names

--essids <file>, -E <file>
: Read a list of ESSIDs out of that file. It will use the same BSSID for all AP which can generate some interesting output in Airodump-ng like: http://www.chimplabs.com/blog/2015/09/24/unintentional-fun-with-aircrack-ng-at-derbycon-5-0/

AUTHOR

This manual page was written by Thomas d'Otreppe. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU General Public License, Version 2 or any later version published by the Free Software Foundation On Debian systems, the complete text of the GNU General Public License can be found in /usr/share/common-licenses/GPL.

SEE ALSO

aireplay-ng(8)
airmon-ng(8)
airodump-ng(8)
airodump-ng-oui-update(8)
airserv-ng(8)
airtun-ng(8)
besside-ng(8)
easside-ng(8)
tkiptun-ng(8)
wesside-ng(8)
aircrack-ng(1)
airdecap-ng(1)
airdecloak-ng(1)
airolib-ng(1)
besside-ng-crawler(1)

buddy-ng(1)
ivstools(1)
kstats(1)
makeivs-ng(1)
packetforge-ng(1)
wpaclean(1)
airventriloquist(8)