Aircrack-ng

Aircrack-ng

Description

Aircrack-ng is an 802.11 WEP and WPA/WPA2-PSK key cracking program.

Aircrack-ng can recover the WEP key once enough encrypted packets have been captured with airodump-ng. This part of the aircrack-ng suite determines the WEP key using two fundamental methods. The first method is via the PTW approach (Pyshkin, Tews, Weinmann). The default cracking method is PTW. This is done in two phases. In the first phase, aircrack-ng only uses ARP packets. If the key is not found, then it uses all the packets in the capture. Please remember that not all packets can be used for the PTW method. This Tutorial: Packets Supported for the PTW Attack page provides details. An important limitation is that the PTW attack currently can only crack 40 and 104 bit WEP keys. The main advantage of the PTW approach is that very few data packets are required to crack the WEP key.

The other, older method is the FMS/KoreK method. The FMS/KoreK method incorporates various statistical attacks to discover the WEP key and uses these in combination with brute forcing. It requires more packets than PTW, but on the other hand is able to recover the passphrase when PTW sometimes fail.

Additionally, the program offers a dictionary method for determining the WEP key.

For cracking WPA/WPA2 pre-shared keys, only a dictionary method is used. A "four-way handshake" is required as input. For WPA handshakes, a full handshake is composed of four packets. However, aircrack-ng is able to work successfully with just 2 packets. EAPOL packets (2 and 3) or packets (3 and 4) are considered a full handshake.

SSE2, AVX, AVX2, and AVX512 support is included to dramatically speed up WPA/WPA2 key processing. With the exception of AVX512, all other instructions are built-in Aircrack-ng, and it will automatically select the fastest available for the CPU. For non-x86 CPUs, SIMD improvements are present as well.

Screenshot

LEGEND
1 = Keybyte
2 = Depth of current key search
3 = Byte the IVs leaked
4 = Votes indicating this is correct

```
KEY FOUND! [ AE:5B:7F:3A:03:D0:AF:9B:F6:8D:A5:E2:C7 ]
```

How does it work?

The first method is the PTW method (Pychkine, Tews, Weinmann). The PTW method is fully described in the paper found on this web site [https://web.archive.org/web/20070714194826/http://www.cdc.informatik.tu-darmstadt.de/aircrack-ptw/]. In 2005, Andreas Klein presented another analysis of the RC4 stream cipher. Klein showed that there are more correlations between the RC4 keystream and the key than the ones found by Fluhrer, Mantin, and Shamir and these may be additionally used to break WEP. The PTW method extends Klein's attack and optimizes it for usage against WEP. It essentially uses enhanced FMS techniques described in the following section. One particularly important constraint is that it only works with arp request/reply packets and cannot be employed against other traffic.

The second method is the FMS/Korek method which incorporates multiple techniques. The Techniques Papers on the links page lists many papers which describe these techniques in more detail and the mathematics behind them.

In this method, multiple techniques are combined to crack the WEP key:

- FMS (Fluhrer, Mantin, Shamir) attacks - statistical techniques
- Korek attacks - statistical techniques
- Brute force

When using statistical techniques to crack a WEP key, each byte of the key is essentially handled individually. Using statistical mathematics, the possibility that a certain byte in the key is correctly guessed goes up to as much as 15% when the right initialization vector (IV) is captured for a particular key byte. Essentially, certain IVs "leak" the secret WEP key for particular key bytes. This is the fundamental basis of the statistical techniques.

By using a series of statistical tests called the FMS and Korek attacks, votes are accumulated for likely keys for each key byte of the secret WEP key. Different attacks have a different number of votes associated with them since the probability of each attack yielding the right answer varies mathematically. The more votes a particular potential key value accumulates, the more likely it is to be correct. For each key byte, the screen shows the likely secret key and the number of votes it has accumulated so far. Needless to say, the secret key with the largest number of votes is most likely correct but is not guaranteed. Aircrack-ng will subsequently test the key to confirm it.

Looking at an example will hopefully make this clearer. In the screenshot above, you can see, that at key byte 0 the byte 0xAE has collected some votes, 50 in this case. So, mathematically, it is more likely that the key starts with AE than with 11 (which is second on the same line) which is almost half as possible. That explains why the more data that is available, the greater the chances that aircrack-ng will determine the secret WEP key.

However the statistical approach can only take you so far. The idea is to get into the ball park with statistics then use brute force to finish the job. Aircrack-ng uses brute force on likely keys to actually determine the secret WEP key.

This is where the fudge factor comes in. Basically the fudge factor tells aircrack-ng how broadly to brute force. It is like throwing a ball into a field then telling somebody to ball is somewhere between 0 and 10 meters (0 and 30 feet) away. Versus saying the ball is somewhere between 0 and 100 meters (0 and 300 feet) away. The 100 meter scenario will take a lot longer to search then the 10 meter one but you are more likely to find the ball with the broader search. It is a trade off between the length of time and likelihood of finding the secret WEP key.

For example, if you tell aircrack-ng to use a fudge factor 2, it takes the votes of the most possible byte, and checks all other possibilities which are at least half as possible as this one on a brute force basis. The larger the fudge factor, the more possibilities aircrack-ng will try on a brute force basis. Keep in mind, that as the fudge factor gets larger, the number of secret keys to try goes up tremendously and consequently the elapsed time also increases. Therefore with more available data, the need to brute force, which is very CPU and time intensive, can be minimized.

In the end, it is all just "simple" mathematics and brute force!

For cracking WEP keys, a dictionary method is also included. For WEP, you may use either the statistical method described above or the dictionary method, not both at the same time. With the dictionary method, you first create a file

with either ascii or hexadecimal keys. A single file can only contain one type, not a mix of both. This is then used as input to aircrack-ng and the program tests each key to determine if it is correct.

The techniques and the approach above do not work for WPA/WPA2 pre-shared keys. The only way to crack these pre-shared keys is via a dictionary attack. This capability is also included in aircrack-ng.

With pre-shared keys, the client and access point establish keying material to be used for their communication at the outset, when the client first associates with the access point. There is a four-way handshake between the client and access point. airodump-ng can capture this four-way handshake. Using input from a provided word list (dictionary), aircrack-ng duplicates the four-way handshake to determine if a particular entry in the word list matches the results the four-way handshake. If it does, then the pre-shared key has been successfully identified.

It should be noted that this process is very computationally intensive and so in practice, very long or unusual pre-shared keys are unlikely to be determined. A good quality word list will give you the best results. Another approach is to use a tool like john the ripper to generate password guesses which are in turn fed into aircrack-ng.

Explanation of the Depth Field and Fudge Factor

The best explanation is an example. We will look at a specific byte. All bytes are processed in the same manner.

You have the votes like in the screen shot above. For the first byte they look like: AE(50) 11(20) 71(20) 10(12) 84(12)

The AE, 11, 71, 10 and 84 are the possible secret key for key byte 0. The numbers in parentheses are the votes each possible secret key has accumulated so far.

Now if you decide to use a fudge factor of 3. Aircrack-ng takes the vote from the most possible byte AE(50):

50 / 3 = 16.666666

Aircrack-ng will test (brute force) all possible keys with a vote greater than 16.6666, resulting in

AE, 11, 71

being tested, so we have a total depth of three:

0 / 3 AE(50) 11(20) 71(20) 10(12) 84(12)

When aircrack-ng is testing keys with AE, it shows 0 / 3, if it has all keys tested with that byte, it switches to the next one (11 in this case) and displays:

1 / 3 11(20) 71(20) 10(12) 84(12)

Usage

```
aircrack-ng [options] <capture file(s)>
```

You can specify multiple input files (either in .cap or .ivs format) or use file name wildcarding. See Other Tips for examples. Also, you can run both airodump-ng and aircrack-ng at the same time: aircrack-ng will auto-update when new IVs are available.

Options

Common options

Option	Param.	Description
-a	amode	Force attack mode (1 = static WEP, 2 = WPA/WPA2-PSK)

Option	Param.	Description
-e	essid	If set, all IVs from networks with the same ESSID will be used. This option is also required for WPA/WPA2-PSK cracking if the ESSID is not broadcasted (hidden)
-b	bssid	Long version --bssid. Select the target network based on the access point's MAC address
-p	nbcpu	On SMP systems: # of CPU to use. This option is invalid on non-SMP systems
-q	*none*	Enable quiet mode (no status output until the key is found, or not)
-C	MACs	Long version --combine. Merge the given APs (separated by a comma) into virtual one
-l	file name	(Lowercase L, ell) logs the key to the file specified. Overwrites the file if it already exists

Static WEP cracking options

Option	Param.	Description
-c	*none*	Restrict the search space to alpha-numeric characters only (0x20 - 0x7F)
-t	*none*	Restrict the search space to binary coded decimal hex characters
-h	*none*	Restrict the search space to numeric characters (0x30-0x39) These keys are used by default in most Fritz!BOXes
-d	start	Long version --debug. Set the beginning of the WEP key (in hex), for debugging purposes
-m	maddr	MAC address to filter WEP data packets. Alternatively, specify -m ff:ff:ff:ff:ff:ff to use all and every IVs, regardless of the network
-n	nbits	Specify the length of the key: 64 for 40-bit WEP, 128 for 104-bit WEP, etc. The default value is 128
-i	index	Only keep the IVs that have this key index (1 to 4). The default behaviour is to ignore the key index
-f	fudge	By default, this parameter is set to 2 for 104-bit WEP and to 5 for 40-bit WEP. Specify a higher value to increase the bruteforce level: cracking will take more time, but with a higher likelyhood of success
-k	korek	There are 17 korek statistical attacks. Sometimes one attack creates a huge false positive that prevents the key from being found, even with lots of IVs. Try -k 1, -k 2, … -k 17 to disable each attack selectively
-x/-x0	*none*	Disable last keybytes brutforce
-x1	*none*	Enable last keybyte bruteforcing (default)
-x2	*none*	Enable last two keybytes bruteforcing
-X	*none*	Disable bruteforce multithreading (SMP only)
-s	*none*	Show the key in ASCII while cracking
-y	*none*	Experimental single bruteforce attack which should only be used when the standard attack mode fails with more than one million IVs
-z	*none*	Invokes the PTW WEP cracking method (Default in v1.x)
-P	number	Long version --ptw-debug. Invokes the PTW debug mode: 1 Disable klein, 2 PTW.
-K	*none*	Invokes the Korek WEP cracking method. (Default in v0.x)
-D	*none*	Long version --wep-decloak. Run in WEP decloak mode
-1	*none*	Long version --oneshot. Run only 1 try to crack key with PTW

Option	Param.	Description
-M	number	(WEP cracking) Specify the maximum number of IVs to use
-V	*none*	Long version --visual-inspection. Run in visual inspection mode (only with KoreK)

WEP and WPA-PSK cracking options

Option	Param.	Description
-w	words	Path to a wordlists or "-" without the quotes for standard in (stdin). Separate multiple wordlists by comma
-N	file	Create a new cracking session and save it to the specified file
-R	file	Restore cracking session from the specified file

WPA-PSK options

Option	Param.	Description
-E	file>	Create EWSA Project file v3
-j	file	Create Hashcat v3.6+ Capture file (HCCAPX)
-J	file	Create Hashcat Capture file
-S	*none*	WPA cracking speed test
-Z	sec	WPA cracking speed test execution length in seconds
-r	database	Utilizes a database generated by airolib-ng as input to determine the WPA key. Outputs an error message if aircrack-ng has not been compiled with sqlite support

SIMD Selection

Option	Param.	Description
--simd	optimization	Use user-specified SIMD optimization instead of the fastest one
--simd-list	*none*	Shows a list of the SIMD optimizations available

Other options

Option	Param.	Description
-H	*none*	Long version --help. Output help information
-u	*none*	Long form --cpu-detect. Provide information on the number of CPUs and features available such as MMX, SSE2, AVX, AVX2, AVX512

Usage Examples

WEP

The simplest case is to crack a WEP key. If you want to try this out yourself, here is a test file [https://download.aircrack-ng.org/wiki-files/other/test.ivs]. The key to the test file matches the screen image above, it does not match the following example.

aircrack-ng -K 128bit.ivs

Where:

- 128bit.ivs is the file name containing IVS.
- -K: Use KoreK attacks only

The program responds:

```
Opening 128bit.ivs
Read 684002 packets.

#  BSSID              ESSID                   Encryption

1  00:14:6C:04:57:9B                          WEP (684002 IVs)

Choosing first network as target.
```

If there were multiple networks contained in the file then you are given the option to select which one you want. By default, aircrack-ng assumes 128 bit encryption.

The cracking process starts and once cracked, here is what it looks like:

```
                              Aircrack-ng 1.4


                    [00:00:10] Tested 77 keys (got 684002 IVs)

KB    depth    byte(vote)
 0    0/ 1    AE( 199) 29(  27) 2D(  13) 7C(  12) FE(  12) FF(   6) 39(   5) 2C(   3) 00(   0) 08(   0)
 1    0/ 3    66(  41) F1(  33) 4C(  23) 00(  19) 9F(  19) C7(  18) 64(   9) 7A(   9) 7B(   9) F6(   9)
 2    0/ 2    5C(  89) 52(  60) E3(  22) 10(  20) F3(  18) 8B(  15) 8E(  15) 14(  13) D2(  11) 47(  10)
 3    0/ 1    FD( 375) 81(  40) 1D(  26) 99(  26) D2(  23) 33(  20) 2C(  19) 05(  17) 0B(  17) 35(  17)
 4    0/ 2    24( 130) 87( 110) 7B(  32) 4F(  25) D7(  20) F4(  18) 17(  15) 8A(  15) CE(  15) E1(  15)
 5    0/ 1    E3( 222) 4F(  46) 40(  45) 7F(  28) DB(  27) E0(  27) 5B(  25) 71(  25) 8A(  25) 65(  23)
 6    0/ 1    92( 208) 63(  58) 54(  51) 64(  35) 51(  26) 53(  25) 75(  20) 0E(  18) 7D(  18) D9(  18)
 7    0/ 1    A9( 220) B8(  51) 4B(  41) 1B(  39) 3B(  23) 9B(  23) FA(  23) 63(  22) 2D(  19) 1A(  17)
 8    0/ 1    14(1106) C1( 118) 04(  41) 13(  30) 43(  28) 99(  25) 79(  20) B1(  17) 86(  15) 97(  15)
 9    0/ 1    39( 540) 08(  95) E4(  87) E2(  79) E5(  59) 0A(  44) CC(  35) 02(  32) C7(  31) 6C(  30)
10    0/ 1    D4( 372) 9E(  68) A0(  64) 9F(  55) DB(  51) 38(  40) 9D(  40) 52(  39) A1(  38) 54(  36)
11    0/ 1    27( 334) BC(  58) F1(  44) BE(  42) 79(  39) 3B(  37) E1(  34) E2(  34) 31(  33) BF(  33)

         KEY FOUND! [ AE:66:5C:FD:24:E3:92:A9:14:39:D4:27:4B ]
```

NOTE: The ASCII WEP key is displayed only when 100% of the hex key can be converted to ASCII.

This key can then be used to connect to the network.

Next, we look at cracking WEP with a dictionary. In order to do this, we need dictionary files with ascii or hexadecimal keys to try. Remember, a single file can only have ascii or hexadecimal keys in it, not both.

WEP keys can be entered in hexadecimal or ascii. The following table describes how many characters of each type is required in your files.

WEP key length in bits	Hexadecimal Characters	Ascii Characters
64	10	5
128	26	13
152	32	16
256	58	29

Example 64 bit ascii key: "ABCDE"
Example 64 bit hexadecimal key: "12:34:56:78:90" (Note the ":" between each two characters.)
Example 128 bit ascii key: "ABCDEABCDEABC"
Example 128 bit hexadecimal key: "12:34:56:78:90:12:34:56:78:90:12:34:56"

To WEP dictionary crack a 64 bit key:

aircrack-ng -w h:hex.txt,ascii.txt -a 1 -n 64 -e teddy wep10-01.cap

Where:

- -w h:hex.txt,ascii.txt is the list of files to use. For files containing hexadecimal values, you must put a "h:" in front of the file name.
- -a 1 says that it is WEP
- -n 64 says it is 64 bits. Change this to the key length that matches your dictionary files.
- -e teddy is to optionally select the access point. Your could also use the "-b" option to select based on MAC address
- wep10-01.cap is the name of the file containing the data. It can be the full packet or an IVs only file. It must contain be a minimum of four IVs.

Here is a sample of the output:

```
                              Aircrack-ng 1.4


                     [00:00:00] Tested 2 keys (got 13 IVs)

KB    depth   byte(vote)
 0    0/  0   00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0)
 1    0/  0   00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0)
 2    0/  0   00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0)
 3    0/  0   00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0)
 4    0/  0   00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0) 00(   0)

                      KEY FOUND! [ 12:34:56:78:90 ]
        Probability: 100%
```

Lets look at a PTW attack example. Remember that this method requires arp request/reply packets as input. It must be the full packet and not just the IVs, meaning that the "-- ivs" option cannot be used when running airodump-ng. As well, it only works for 64 and 128 bit WEP encryption.

Enter the following command:

```
 aircrack-ng -z ptw*.cap
```

Where:

- -z means use the PTW methodology to crack the wep key. *Note:* in v1.x, this is the default attack mode; use -K to revert to Korek.
- ptw*.cap are the capture files to use.

The systems responds:

```
Opening ptw-01.cap
Read 171721 packets.

#  BSSID              ESSID                  Encryption

1  00:14:6C:7E:40:80  teddy                  WEP (30680 IVs)

Choosing first network as target.
```

Then:

```
                              Aircrack-ng 1.4

                     [00:01:18] Tested 0/140000 keys (got 30680 IVs)

KB    depth   byte(vote)
 0    0/  1   12( 170) 35( 152) AA( 146) 17( 145) 86( 143) F0( 143) AE( 142) C5( 142) D4( 142) 50( 140)
 1    0/  1   34( 163) BB( 160) CF( 147) 59( 146) 39( 143) 47( 142) 42( 139) 3D( 137) 7F( 137) 18( 136)
 2    0/  1   56( 162) E9( 147) 1E( 146) 32( 146) 6E( 145) 79( 143) E7( 142) EB( 142) 75( 141) 31( 140)
```

```
3    0/  1   78( 158) 13( 156) 01( 152) 5F( 151) 28( 149) 59( 145) FC( 145) 7E( 143) 76( 142) 92( 142)
4    0/  1   90( 183) 8B( 156) D7( 148) E0( 146) 18( 145) 33( 145) 96( 144) 2B( 143) 88( 143) 41( 141)

                          KEY FOUND! [ 12:34:56:78:90 ]
      Decrypted correctly: 100%
```

WPA

Now onto cracking WPA/WPA2 passphrases. Aircrack-ng can crack either types.

aircrack-ng -w password.lst *.cap
Where:

- -w password.lst is the name of the password file. Remember to specify the full path if the file is not located in the same directory.
- *.cap is name of group of files containing the captured packets. Notice in this case that we used the wildcard * to include multiple files.

The program responds:

```
 Opening wpa2.eapol.cap
 Opening wpa.cap
 Read 18 packets.

 #  BSSID              ESSID                  Encryption

 1  00:14:6C:7E:40:80  Harkonen               WPA (1 handshake)
 2  00:0D:93:EB:B0:8C  test                   WPA (1 handshake)

 Index number of target network ?
```

Notice in this case that since there are multiple networks we need to select which one to attack. We select number 2. The program then responds:

```
                            Aircrack-ng 1.4


                [00:00:03] 230 keys tested (73.41 k/s)


                        KEY FOUND! [ biscotte ]


    Master Key      : CD D7 9A 5A CF B0 70 C7 E9 D1 02 3B 87 02 85 D6
                      39 E4 30 B3 2F 31 AA 37 AC 82 5A 55 B5 55 24 EE

    Transcient Key  : 33 55 0B FC 4F 24 84 F4 9A 38 B3 D0 89 83 D2 49
                      73 F9 DE 89 67 A6 6D 2B 8E 46 2C 07 47 6A CE 08
                      AD FB 65 D6 13 A9 9F 2C 65 E4 A6 08 F2 5A 67 97
                      D9 6F 76 5B 8C D3 DF 13 2F BC DA 6A 6E D9 62 CD

    EAPOL HMAC      : 52 27 B8 3F 73 7C 45 A0 05 97 69 5C 30 78 60 BD
```

Now you have the passphrase and can connect to the network.

SIMD

Aircrack-ng is compiled with multiple optimizations based on CPU features we call crypto engines. CPU features are different based on the type of CPU.

On x86 (and 64 bit), typically SSE2, AVX and AVX2 are available (AVX512 can be compiled in but it should only be done if the current CPU supports it). On ARM, neon and ASIMD are usually available and on PowerPC, ASIMD and altivec. A generic optimization is always available no matter what architecture it is compiled on or for. A limited set of optimizations may be available depending on the OS/CPU/compilers available.

When running aircrack-ng, it will load the fastest optimization based on what your CPU supports. For package

maintainers, it is very useful as they don't have to target the one supporting all the CPU which would be the slowest.

In order to override, the option --simd can be used. Such as

```
aircrack-ng --simd=avx wpa.cap -w password.lst
```

In order to list all the available SIMD optimization, use --simd-list. Such as

```
aircrack-ng --simd-list
```

will display "avx2 avx sse2 generic" on x86.

Cracking session

Cracking can sometimes take a very long time and it is sometimes necessary to turn off the computer or put it to sleep for a while. In order to handle this kind of situation, a new set of option has been created.

It will create and/or update a session file saving the current status of the cracking (every 10 minutes) as well as all the options used, wordlists and capture files used. Multiple wordlists can be used and it works with WEP and WPA.

```
aircrack-ng --new-session current.session -w password.lst,english.txt wpa-01.cap
```

In order to restore the session, use --restore-session:

```
aircrack-ng --restore-session current.session
```

It will keep updating *current.session* every 10 minutes.

Limitations:

- The wordlist must be files. For now, they cannot be *stdin* or airolib-ng databases
- Session has to be restored from the same directory as when first using --new-session
- No new options can be added when restoring session

Usage Tips

General approach to cracking WEP keys

 Fix Me! *This needs updating for v1.x!*

Clearly, the simplest approach is just to enter "aircrack-ng captured-data.cap" and let it go. Having said that, there are some techniques to improve your chances of finding the WEP key quickly. There is no single magic set of steps. The following describes some approaches which tend to yield the key faster. Unless you are comfortable with experimentation, leave well enough alone and stick to the simple approach.

If you are capturing arp request/reply packets, then the fastest approach is to use "aircrack-ng -z <data packet capture files>". You can then skip the balance of this section since it will find the key very quickly assuming you have collected sufficient arp request/reply packets! *NOTE:* -z is the default attack mode in aircrack-ng v1.x; use -K to revert to the attack mode used in previous versions.

The overriding technique is capture as much data as possible. That is the single most important task. The number of initialization vectors (IVs) that you need to determine the WEP key varies dramatically by key length and access point. Typically you need 250,000 or more unique IVs for 64 bit keys and 1.5 million or more for 128 bit keys. Clearly a lot more for longer key bit lengths. Then there is luck. There will be times that the WEP key can be determined with as few as 50,000 IVs although this is rare. Conversely, there will be times when you will need mulitple millions of IVs to crack the WEP key. The number of IVs is extremely hard to predict since some access points are very good at eliminating IVs that lead the WEP key.

Generally, don't try to crack the WEP key until you have 200,000 IVs or more. If you start too early, aircrack tends to spend too much time brute forcing keys and not properly applying the statistical techniques. Start by trying 64 bit keys

"aircrack-ng -n 64 captured-data.cap". If they are using a 64 bit WEP, it can usually be cracked in less then 5 minutes (generally less then 60 seconds) with relatively few IVs. It is surprising how many APs only use 64 bit keys. If it does not find the 64 bit key in 5 minutes, restart aircrack in the generic mode: "aircrack-ng captured-data.cap". Then at each 100,000 IVs mark, retry the "aircrack-ng -n 64 captured-data.cap" for 5 minutes.

Once you hit 600,000 IVs, switch to testing 128 bit keys. At this point it is unlikely (but not impossible) that it is a 64 bit key and 600,000 IVs did not crack it. So now try "aircrack-ng captured-data.cap".

Once you hit 2 million IVs, try changing the fudge factor to "-f 4". Run for at least 30 minutes to one hour. Retry, increasing the fudge factor by adding 4 to it each time. Another time to try increasing the fudge factor is when aircrack-ng stops because it has tried all the keys.

All the while, keep collecting data. Remember the golden rule, "the more IVs the better".

Also check out the next section on how to determine which options to use as these can significantly speed up cracking the WEP key. For example, if the key is all numeric, then it can take as few as 50,000 IVs to crack a 64 bit key with the "-t" versus 200,000 IVs without the "-t". So if you have a hunch about the nature of the WEP key, it is worth trying a few variations.

How to determine which options to use

While aircrack-ng is running, you mostly just see the beginning of the key. Although the secret WEP key is unknown at this point, there may be clues to speed things up. If the key bytes have a fairly large number of votes, then they are likely 99.5% correct. So lets look at what you can do with these clues.

If the bytes (likely secret keys) are for example: 75:47:99:22:50 then it is quite obvious, that the whole key may consist only of numbers, like the first 5 bytes. So it MAY improve your cracking speed to use the -t option only when trying such keys. See Wikipedia Binary Coded Decimal [https://en.wikipedia.org/wiki/Binary-coded_decimal] for a description of what characters -t looks for.

If the bytes are 37:30:31:33:36 which are all numeric values when converted to Ascii, it is a good idea to use -h option. The FAQ entry Converting hex characters to ascii provides links to determine if they are all numeric.

And if the first few bytes are something like 74:6F:70:73:65, and upon entering them into your hexeditor or the links provided in the previous sentence, you see that they may form the beginning of some word, then it seems likely an ASCII key is used, thus you activate -c option to check only printable ASCII keys.

If you know the start of the WEP key in hexadecimal, you can enter with the "-d" parameter. Lets assume you know the WEP key is "0123456789" in hexadecimal then you could use "-d 01" or "-d 0123", etc.

Another option to try when having problems determining the WEP key, is the "-x2" option which causes the last two keybytes to be brute forced instead of the default of one.

How to convert the HEX WEP key to ASCII?

See the next entry.

How to use the key

If aircrack-ng determines the key, it is presented to you in hexadecimal format. It typically looks like:

```
KEY FOUND! [11:22:33:44:55]
```

The length will vary based on the WEP bit key length used. See the table above which indicates the number of hexadecimal characters for the various WEP key bit lenghts.

You may use this key without the ":" in your favorite client. This means you enter "1122334455" into the client and specify that the key is in hexadecimal format. Remember that most keys cannot be converted to ASCII format. If the HEX key is in fact valid ASCII characters, the ASCII will also be displayed.

If you wish to experiment a bit with converting HEX to ASCII, see this FAQ entry.

We do not specifically provide support or the details on how to configure your wireless card to connect to the AP. For linux, this page [https://web.archive.org/web/20080212235953/http://wirelessdefence.org/Contents/LinuxWirelessCommands.htm] has an excellent writeup. As well, search the internet for this information regarding linux and Windows systems. As well, see the documentation for your card's wireless client. If you are using linux, check the mailing lists and forums specific to the distribution.

Additionally, Aircrack-ng prints out a message indicating the likelihood that the key is correct. It will look something similar to "Probability: 100%". Aircrack-ng tests the key against some packets to confirm the key is correct. Based on these tests, it prints the probability of a correct key.

Also remember we do not support or endorse people accessing networks which do not belong to them.

How to convert the hex key back to the passphrase?

People quite often ask if the hexadecimal key found by aircrack-ng can be converted backwords to the original "passphrase". The simple answer is "NO".

To understand why this is so, lets take a look at how these passphrases are converted into the hexadecimal keys used in WEP.

Some vendors have a wep key generator which "translates" a passphrase into a hexadecimal WEP key. There are no standards for this. Very often they just pad short phrases with blanks, zeroes or other characters. However, usually the passphrases are filled with zeros up to the length of 16 bytes, and afterwards the MD5SUM of this bytestream will be the WEP Key. Remember, every vendor can do this in a slightly different way, and so they may not be compatible.

So there is no way to know the how long the original passphrase was. It could as short as one character. It all depends on the who developed the software.

Knowing all this, if you still wish to try to obtain the original passphrase, Latin SuD has a tool which attempts reverse the process. Click here [https://www.latinsud.com/wepconv.html] for the tool.

Nonetheless, these passphrases result in a WEP Key that is as easily cracked as every other WEP Key. The exact conversion method really does not matter in the end.

Keep in mind that wep passwords that look like "plain text" might either be ASCII or PASSPHRASE. Most (all) systems support ASCII and are the default, but some support passphrase and those which support it require users to specify whether it's ascii or a passphrase. Passphrases can be any arbitrary length. ASCII are usually limited to 5 or 13 (wep40 and wep104).

As a side note, Windows WZC only supports fixed length hex or ascii keys, so the shortest inputable key is 5 characters long. See the table above on this page regarding how many characters are needed for specific key lengths.

Sample files to try

There are a number of sample files that you can try with aircrack-ng to gain experience:

- wpa.cap [https://github.com/aircrack-ng/aircrack-ng/raw/master/test/wpa.cap]: This is a sample file with a wpa handshake. It is located in the "test" directory of the install files. The passphrase is "biscotte". Use the password file (password.lst) which is in the same directory.
- wpa2.eapol.cap [https://github.com/aircrack-ng/aircrack-ng/raw/master/test/wpa2.eapol.cap]: This is a sample file with a wpa2 handshake. It is located in the "test" directory of the install files. The passphrase is "12345678". Use the password file (password.lst) which is in the same directory.
- test.ivs [https://download.aircrack-ng.org/wiki-files/other/test.ivs]: This is a 128 bit WEP key file. The key is "AE:5B:7F:3A:03:D0:AF:9B:F6:8D:A5:E2:C7".
- ptw.cap [https://github.com/aircrack-ng/aircrack-ng/raw/master/test/wep_64_ptw.cap]: This is a 64 bit WEP key file suitable for the PTW method. The key is "1F:1F:1F:1F:1F".

- wpa-psk-linksys.cap [https://github.com/aircrack-ng/aircrack-ng/raw/master/test/wpa-psk-linksys.cap]: This is a sample file with a WPA1 handshake along with some encrypted packets. Useful for testing with airdecap-ng. The password is "dictionary".
- wpa2-psk-linksys.cap [https://github.com/aircrack-ng/aircrack-ng/raw/master/test/wpa2-psk-linksys.cap]: This is a sample file with a WPA2 handshake along with some encrypted packets. Useful for testing with airdecap-ng. The password is "dictionary".

Dictionary Format

Dictionaries used for WPA/WPA bruteforcing need to contain one passphrase per line.

The linux and Windows end of line format is slightly different. See this Wikipedia entry [https://en.wikipedia.org/wiki/Line_feed] for details. There are conversion tools are available under both linux and Windows which can convert one format to another. As well, editors are available under both operating systems which can edit both formats correctly. It is up to the reader to use an Internet search engine to find the appropriate tools.

However both types should work with the linux or Windows versions of aircrack-ng. Thus, you really don't need to convert back and forth.

Hexadecimal Key Dictionary

Although it is not part of aircrack-ng, it is worth mentioning an interesting piece of work is by SuD. It is basically a wep hex dictionary already prepared and the program to run it:

```
https://www.latinsud.com/pub/wepdict/
```

Tools to split capture files

There are times when you want to split capture files into smaller pieces. For example, files with a large number of IVs can sometimes cause the PTW attack to fail. In this case, it is worth splitting the file into smaller pieces and retrying the PTW attack.

So here are two tools to split capture files:

- https://www.badpenguin.co.uk/files/pcap-util [https://www.badpenguin.co.uk/files/pcap-util]
- https://www.badpenguin.co.uk/files/pcap-util2 [https://www.badpenguin.co.uk/files/pcap-util2]

Another technique is to use Wireshark / tshark. You can mark packets then same them to a separate file.

How to extract WPA handshake from large capture files

Sometimes you have a very large capture file and would like to extract the WPA/WPA2 handshake packets from it to a separate file. The can be done with "tshark" which is a command line version of the Wireshark suite. Installing the linux version of the Wireshark suite [https://www.wireshark.org] on your system should also install tshark.

The following command will extract all handshake and beacon packets from your pcap capture file and create a separate file with just those packets:

```
tshark -r <input file name> -R "eapol || wlan.fc.type_subtype == 0x08" -w <output file name>
```

Remember you must use a pcap file as input, not an IVs file.

Other Tips

To specify multiple capture files at a time you can either use a wildcard such as * or specify each file individually.

Examples:

- aircrack-ng -w password.lst wpa.cap wpa2.eapol.cap

- aircrack-ng *.ivs
- aircrack-ng something*.ivs

To specify multiple dictionaries at one time, enter them comma separated with no spaces.

Examples:

- aircrack-ng -w password.lst,secondlist.txt wpa2.eapol.cap
- aircrack-ng -w firstlist.txt,secondlist.txt,thirdlist.txt wpa2.eapol.cap

Aircrack-ng comes with a small dictionary called password.lst. The password.lst file is located in the "test" directory of the source files. This FAQ entry has a list of web sites where you can find extensive wordlists (dictionaries). Also see this thread [https://forum.aircrack-ng.org/index.php?topic=1373] on the Forum.

Determining the WPA/WPA2 passphrase is totally dependent on finding a dictionary entry which matches the passphrase. So a quality dictionary is very important. You can search the Internet for dictionaries to be used. There are many available.

The tutorials page has the following tutorial How to crack WPA/WPA2? which walks you through the steps in detail.

As you have seen, if there are multiple networks in your files you need to select which one you want to crack. Instead of manually doing a selection, you can specify which network you want by essid or bssid on the command line. This is done with the -e or -b parameters.

Another trick is to use John the Ripper to create specific passwords for testing. Lets say you know the passphrase is the street name plus 3 digits. Create a custom rule set in JTR and run something like this:

```
john --stdout --wordlist=specialrules.lst --rules | aircrack-ng -e test -a 2 -w - /root/capture/wpa.cap
```

Remember that valid passwords are 8 to 63 characters in length. Here is a handy command to ensure all passwords in a file meet this criteria:

```
awk '{ if ((length($0) > 7) && (length($0) < 64)){ print $0 }}' inputfile
```

or

```
grep -E '^.{8,63}$' < inputfile
```

Usage Troubleshooting

Error message "Please specify a dictionary (option -w)"

This means you have misspelt the file name of the dictionary or it is not in the current directory. If the dictionary is located in another directory, you must provide the full path to the dictionary.

Error message "fopen(dictionary)failed: No such file or directory"

This means you have misspelt the file name of the dictionary or it is not in the current directory. If the dictionary is located in another directory, you must provide the full path to the dictionary.

Negative votes

There will be times when key bytes will have negative values for votes. As part of the statistical analysis, there are safeguards built in which subtract votes for false positives. The idea is to cause the results to be more accurate. When you get a lot of negative votes, something is wrong. Typically this means you are trying to crack a dynamic key such as WPA/WPA2 or the WEP key changed while you were capturing the data. Remember, WPA/WPA2 can only be cracked via a dictionary technique. If the WEP key has changed, you will need to start gathering new data and start over again.

"An ESSID is required. Try option -e" message

You have successfully captured a handshake then when you run aircrack-ng, you get similar output:

```
Opening wpa.cap
Read 4 packets.

        #   BSSID                ESSID                ENCRYPTION
        1   00:13:10:F1:15:86                         WPA (1) handshake
Choosing first network as target.

An ESSID is required. Try option -e.
```

Solution: You need to specify the real essid, otherwise the key cannot be calculated, as the essid is used as salt when generating the pairwise master key (PMK) out of the pre-shared key (PSK).

So just use -e "<REAL_ESSID>" instead of -e "" and aircrack-ng should find the passphrase.

The PTW method does not work

One particularly important constraint is that it only works against arp request/reply packets. It cannot be used against any other data packets. So even if your data capture file contains a large number of data packets, if there insufficient arp request/reply packets, it will not work. Using this technique, 64-bit WEP can be cracked with as few as 20,000 data packets and 128-bit WEP with 40,000 data packets. As well, it requires the full packet to be captured. Meaning you cannot use the "-- ivs" option when running airodump-ng. It also only works for 64 and 128 bit WEP encryption.

Error message "read(file header) failed: Success"

If you get the error message - "read(file header) failed: Success" or similar when running aircrack-ng, there is likely an input file with zero (0) bytes. The input file could be a .cap or .ivs file.

This is most likely to happen with wildcard input of many files such as:

```
aircrack-ng -z -b XX:XX:XX:XX:XX:XX *.cap
```

Simply delete the files with zero bytes and run the command again.

WPA/WPA2 Handshake Analysis Fails

Capturing WPA/WPA2 handshakes can be very tricky. A capture file may end up containing a subset of packets from various handshake attempts and/or handshakes from more then one client. Currently aircrack-ng can sometimes fail to parse out the handshake properly. What this means is that aircrack-ng will fail to find a handshake in the capture file even though one exists.

If you are sure your capture file contains a valid handshake then use Wireshark or an equivalent piece of software and manually pull out the beacon packet plus a set of handshake packets.

There is an open GitHub issue [https://github.com/aircrack-ng/aircrack-ng/issues/651] to correct this incorrect behavior.

---

aircrack-ng.txt · Last modified: 2019/09/18 22:39 by mister_x