**Abstract- *To demonstrate the working of Aircrack Suite for cracking WPA/WPA2 passwords with the help for custom made “word file” which consist a subset of all possible common passwords combinations of alphabets and numbers.***

**Introduction**

This project is based on cracking WPA/WPA2 networks which use pre-shared keys. Most wireless access points now use Wi-Fi Protected Access II with a pre-shared key for wireless security, known as WPA2-PSK. WPA2 uses a stronger encryption algorithm, AES, that's very difficult to crack but not impossible. These passwords are cracked using either dictionary attack or Brute force attack. The environment used for running the process is Ubuntu 17.04.

WPA/WPA2 supports many types of authentication beyond pre-shared keys. aircrack-ng can only crack pre-shared keys. So, the command airodump-ng is used to show the network as having the authentication type of PSK. There is another important difference between cracking WPA/WPA2 and WEP. This is the approach used to crack the WPA/WPA2 pre-shared key. Unlike WEP, where statistical methods can be used to speed up the cracking process, only plain brute force techniques can be used against WPA/WPA2. That is, because the key is not static, so collecting IVs like when cracking WEP encryption, does not speed up the attack.

The weakness in the WPA2-PSK system is that the encrypted password is shared in what is known as the 4-way handshake. When a client authenticates to the access point (AP), the client and the AP go through a 4-step process to authenticate the user to the AP. If we can grab the password at that time, we can then attempt to crack it.

**Software Requirement:**

* Ubuntu 17.04
* Aircrack-ng suite
* Wordlist *(consisting of common passwords)*

**Hardware Requirement:**

* Aircrack-ng supported Wifi card
* A working computer

**Implementation**

**Step 1: Start the wireless interface in monitor mode**

The purpose of this step is to put your card into what is called monitor mode. Monitor mode is the mode whereby your card can listen to every packet in the air. Normally your card will only “hear” packets addressed to you. By hearing every packet, we can later capture the WPA/WPA2 4-way handshake. As well, it will allow us to optionally deauthenticate a wireless client in a later step. The exact procedure for enabling monitor mode varies depending on the driver you are using. To determine the driver (and the correct procedure to follow), run the airmon-ng command.

**Airmon-ng**

This script can be used to enable monitor mode on wireless interfaces. It may also be used to go back from monitor mode to managed mode. Entering the airmon-ng command without parameters will show the interfaces status.

Usage: airmon-ng <start|stop> <interface> [channel] or airmon-ng <check|check kill>

Where:

* <start|stop> indicates if you wish to start or stop the interface. (Mandatory)
* <interface> specifies the interface. (Mandatory)
* [channel] optionally set the card to a specific channel.
* <check|check kill> “check” will show any processes that might interfere with the aircrack-ng suite. It is strongly recommended that these processes be eliminated prior to using the aircrack-ng suite. “check kill” will check and kill off processes that might interfere with the aircrack-ng suite.

**Step 2: Start airodump-ng to collect authentication handshake**

The purpose of this step is to run airodump-ng to capture the 4-way authentication handshake for the AP we are interested in.

Enter: airodump-ng -c 9 --bssid 00:14:6C:7E:40:80 -w psk ath0

Where:

* -c 9 is the channel for the wireless network
* --bssid 00:14:6C:7E:40:80 is the access point MAC address. This eliminates extraneous traffic.
* -w psk is the filename prefix for the file which will contain the IVs.
* ath0 is the interface name.

**Airodump-ng**

Airodump-ng is used for packet capturing of raw 802.11 frames and is particularly suitable for collecting WEP IVs (Initialization Vector) for the intent of using them with aircrack-ng. If you have a GPS receiver connected to the computer, airodump-ng is capable of logging the coordinates of the found access points. Additionally, airodump-ng writes out several files containing the details of all access points and clients seen.

Usage: airodump-ng <options> <interface>[,<interface>,...]

*Options:*

* --ivs : Save only captured IVs
* --gpsd : Use GPSd
* --write <prefix> : Dump file prefix
* -h : Hides known stations for --showack
* -r <file> : Read packets from that file
* -x <msecs> : Active Scanning Simulation
* --ignore-negative-one : Removes the message that says fixed channel <interface>: -1

*Filter options:*

* --encrypt <suite> : Filter APs by cipher suite
* --netmask <netmask> : Filter APs by mask
* --bssid <bssid> : Filter APs by BSSID
* --essid <essid> : Filter APs by ESSID
* --essid-regex <regex> : Filter APs by ESSID using a regular expression
* -a : Filter unassociated clients

**Step 3** **:** **Use aireplay-ng to deauthenticate the wireless client**

If you are patient, you can wait until airodump-ng captures a handshake when one or more clients connect to the AP. You only perform this step if you opted to actively speed up the process. The other constraint is that there must be a wireless client currently associated with the AP. If there is no wireless client currently associated with the AP, then you have to be patient and wait for one to connect to the AP so that a handshake can be captured. Needless to say, if a wireless client shows up later and airodump-ng did not capture the handshake, you can backtrack and perform this step.

This step sends a message to the wireless client saying that that it is no longer associated with the AP. The wireless client will then hopefully reauthenticate with the AP. The reauthentication is what generates the 4-way authentication handshake we are interested in collecting. This is what we use to break the WPA/WPA2 pre-shared key. Based on the output of airodump-ng in the previous step, you determine a client which is currently connected. You need the MAC address for the following.

Open another console session and enter:

aireplay-ng -0 1 -a 00:14:6C:7E:40:80 -c 00:0F:B5:FD:FB:C2 ath0

Where:

* -0 means deauthentication
* 1 is the number of deauths to send (you can send multiple if you wish)
* -a 00:14:6C:7E:40:80 is the MAC address of the access point
* -c 00:0F:B5:FD:FB:C2 is the MAC address of the client you are deauthing
* ath0 is the interface name

**Aireplay-ng**

Aireplay-ng is used to inject frames. The primary function is to generate traffic for the later use in aircrack-ng for cracking the WEP and WPA-PSK keys. There are different attacks which can cause deauthentications for the purpose of capturing WPA handshake data, fake authentications, Interactive packet replay, hand-crafted ARP request injection and ARP-request reinjection. With the packetforge-ng tool it's possible to create arbitrary frames. Most drivers needs to be patched to be able to inject, don't forget to read Installing drivers.

Usage of the attacks:

It currently implements multiple different attacks:

* Attack 0: Deauthentication
* Attack 1: Fake authentication
* Attack 2: Interactive packet replay
* Attack 3: ARP request replay attack
* Attack 4: KoreK chopchop attack
* Attack 5: Fragmentation attack
* Attack 6: Caffe-latte attack
* Attack 7: Client-oriented fragmentation attack
* Attack 8: WPA Migration Mode
* Attack 9: Injection test

Usage: aireplay-ng <options> <replay interface>

Filter options:

* -b bssid : MAC address, Access Point
* -d dmac : MAC address, Destination
* -s smac : MAC address, Source
* -m len : minimum packet length

**Step 4: Run aircrack-ng to crack the pre-shared key**

The purpose of this step is to actually crack the WPA/WPA2 pre-shared key. To do this, you need a dictionary of words as input. Basically, aircrack-ng takes each word and tests to see if this is in fact the pre-shared key. There is a small dictionary that comes with aircrack-ng - “password.lst”. This file can be found in the “test” directory of the aircrack-ng source code. The Wiki FAQ has an extensive list of dictionary sources. You can use ***John the Ripper (JTR)*** to generate your own list and pipe them into aircrack-ng. Using JTR in conjunction with aircrack-ng is beyond the scope of this project.

Open another console session and enter:

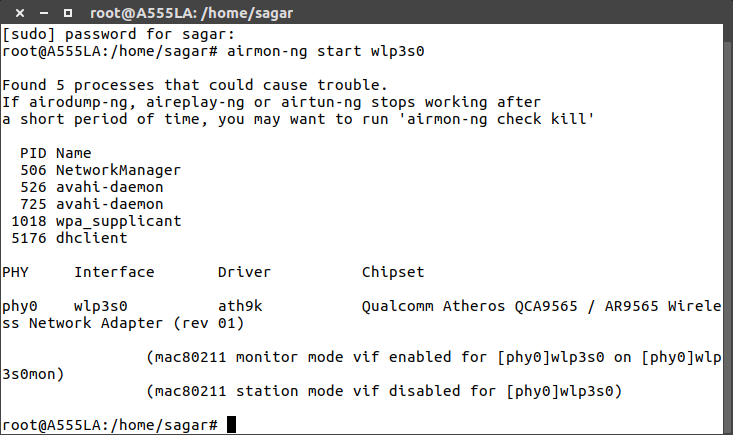
aircrack-ng -w password.lst -b 00:14:6C:7E:40:80 psk\*.cap

**Aircrack-ng**

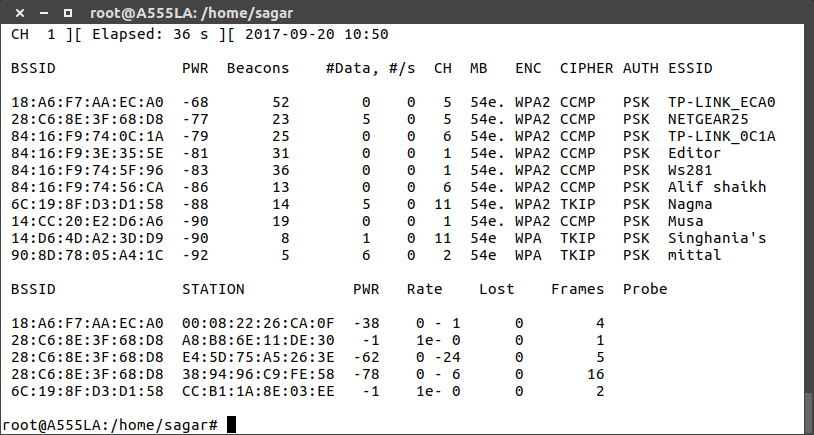
Aircrack-ng is an 802.11 WEP and WPA/WPA2-PSK key cracking program. Aircrack-ng can recover the WEP key once enough encrypted packets have been captured with airodump-ng. This part of the aircrack-ng suite determines the WEP key using two fundamental methods. The first method is via the PTW approach *(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. 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 second method is the FMS/KoreK method.

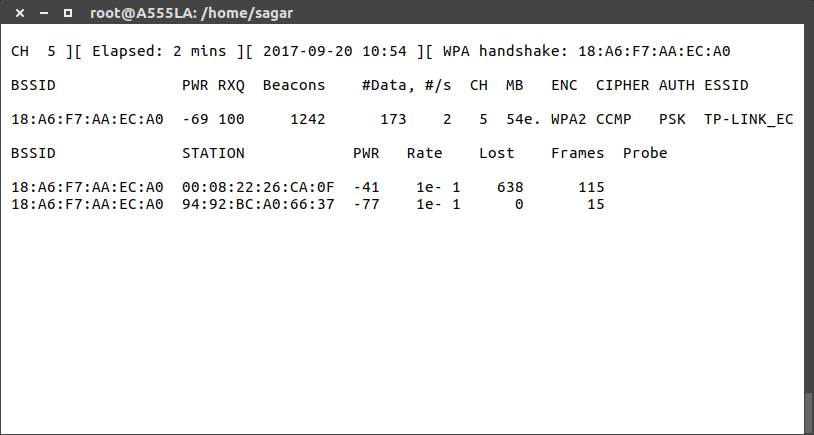
**Output:**

**Step 1: Start the wireless interface in monitor mode**

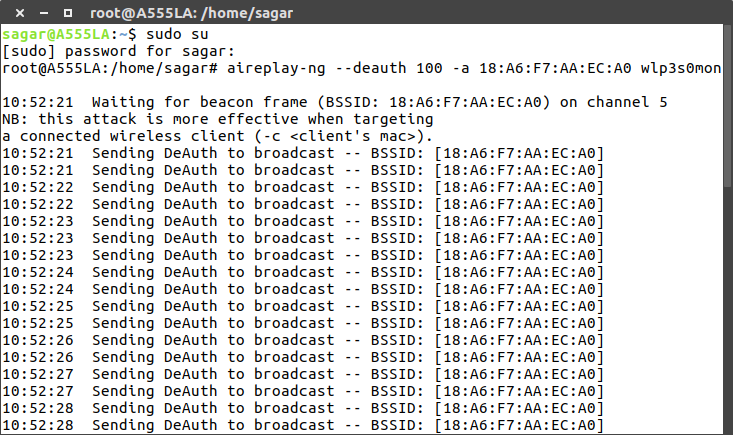


**Step 2: Start airodump-ng to collect authentication handshake**

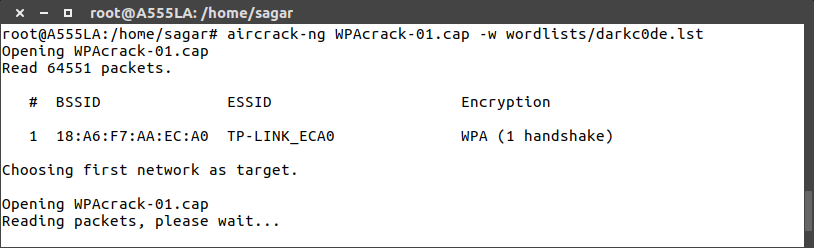
****

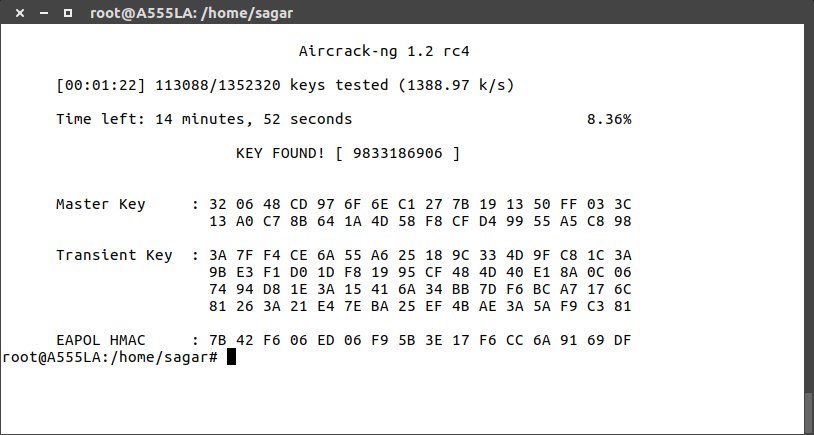
****

**Step 3** - **Use aireplay-ng to deauthenticate the wireless client**

****

**Step 4: Run aircrack-ng to crack the pre-shared key**

****

****

**Conclusion**

Hence, the custom word file for all possible common passwords has been created and with the help of aircrack suite the working of password cracking has been demonstrated successfully.

**References**

[1] Mathew Monte “Network Attacks and Exploitation: A Framework”, Wiley Publications 1st Edition

[2]Dileep Kumar G., Manoj Kumar Singh, M.K. Jayanthi “Network Security Attacks and Countermeasures”, 2nd Edition