**LINUX BASICS**

**CYBER SECURITY**

**apt-get update**

* update the sources where Kali can search and download programs from
* "apt-get" is the name of the application that allow us to download and install programs
* "update" is because i am saying that i want you to update the list of all the programs that i can install

**apt-get install terminator**

* this command installs the terminator
* “terminator” is the terminal program that allow us to have multiple terminal windows open in the same window
* You can split the terminator in 2, this is special property

**NETWORK BASICS**

MAC Address (Media Access Control)

* Permanent, physical, unique address.
* Assigned to network interfaces (any device that allow us to connect to a network) by the device manufacturer.
* Each network cards come with a specific address that is unique to this card.
* Why change the MAC address:
  + Increase anonymity.
  + Impersonate other devices.
  + Bypass filters.
  + Connect to networks that only specific devices with specific MAC addresses can connect to.

\*IP address is used in the internet to identify computers, and communicate between devices ın the internet.

Changing MAC address in Linux:

**ifconfig** : list all the network interfaces

* eth0 : virtual interface created by virtual box when we set Kali to use NAT network.
* ether : MAC address

**ifconfig eth0**(or any other interface) **down** ---> Disable the interface

**ifconfig eth0 hw ether**(we want to change hardware address – MAC) **00:11:22:33:44:55**(e.g.)

**ifconfig eth0 up** ---> Enable the interface

We are only changing the MAC address in memory.

\*lo is default interface created by Linux.

Devices on the same network communicate with each other using packets.

In the network, devices ensure that these packets go in the right direction using the MAC address.

Each packet has a source MAC and a destination MAC.

Each device only receives data that has the destination MAC as its own MAC address.

We can capture packets even if they do not have our MAC address as a destination address. To do this, we need to change the mode of operation of our wireless interface so that it operates in monitor mode.

**iwconfig** ---> to see the wireless interfaces only

You can see the wlan0 (wifi adapter), has some informations. Its mode is set to managed. What this means is basically this is the default mode of all wireless devices and it means this device will only capture packets that has the destination MAC as the MAC address of this device.

We want to capture all the packets that are within our range even if they are sent to the router or another device. SO TO DO THIS, we need to set the mode to monitor mode.

**ifconfig wlan0**(or any other interface) **down** ---> Disable the interface

**airmon-ng check kill**  --->to kill any process that could interfere with using my interface (adapter) in monitor mode. Airmon-ng is the name of the program. I say I want you to check all the processes that can interfere with monitor mode and if you find anything, kill. Not mandatory but running it will actually give you better results when you come to running the attacks that you will learn. It will actually kill the network manager that usually runs in top right corner. You lost your internet connection. This is not a problem because we will only need to be in monitor mode when we’re running pre-connection attacks. So we actually do not need internet connection to run any of the attacks that require monitor mode.

**iwconfig wlan0 mode monitor**

**ifconfig wlan0 up** ---> Enable the interface

**PRE-CONNECTION ATTACKS**

**PACKET SNIFFING**

In monitor mode we are able to capture all the wifi packets sent within our range even if the packet is not directed to our computer or even if we are not connected to the target network and even without knowing the key -password- of the target network.

**airodump-ng**

* program that we will use to capture the packets
* part of the **aircrack-ng** suit
* it’s a packet sniffer
* it’s a program designed to capture packets while you are in monitor mode
* allows us to see all the wireless networks around us, and show us detailed information about it’s MAC address, channel encryption, the clients connected to this network etc.

Running it:

1. **airodump-ng wlan0** ---> works unless you stop it

**BSSID** ---> MAC address of the target network

**PWR** ---> signal strength – power

**Beacons** ---> frames (name of the packets of datas) sent by the network in order to broadcast it’s existence. EVERY network even if it’s set to be hidden always sends this type of frames, basically broadcasting it’s existence, and telling all the wireless devices around it that “I exist, I have this BSSID, I work on this channel, I use this encryption, and my name is this (ESSID).”.

**#DATA** ---> Number of data packets, or data frames, and these are the useful packets

**#/s** ---> Number of data packets that we collected in the past 10 seconds

**CH** ---> Channel that the network works on

**MB** ---> max speed supported by the network

**ENC** ---> Encryption type used by the network

**CIPHER** ---> cipher used in the network

**AUTH** ---> Authentication used on that network (PSE: pre shared key)

**ESSID** ---> names of the networks

***WiFi BANDS***

* Band of a network defines what frequency it can use to broadcast the signal. This means it also defines the frequency that the clients or the computers need to be able to support and use in order to be able to connect this network.
* Two main frequencies used in wifi networks are 2.4 and 5 GHz.
* Data can be sniffed from a certain band if the wireless adapter used supports that band.
* With airodump-ng command, you can’t see all the networks around you because your adapter may not be able to support. This command is only sniffing on 2.0 GHz frequency. More than 2 GHz is outside of adapter’s limit, it can’t see them.
* Problem with 5 GHz is that, there are a lot of wireless adapters that can see it and can communicate with it. But not many of them support monitor mode and packet injection.
* MOST COMMON WIFI BANDS ARE:
  + a ---> uses 5 GHz frequency only
  + b,g ---> both use 2.4 GHz frequency only
  + n ---> uses 5 and 2.4 GHz
  + ac ---> uses frequencies lower than 6 GHz

If your adapter supports 5 GHz, you still can’t see all the networks. Because you need to specifically tell airodump-ng that “i want you to listen on 5 GHz frequencies and 5GHz channels.”. To do that:

**airodump-ng --band a wlan0**

You can specify multiple bands using the band argument. This command will tell airodump-ng to capture data on both 2.4 and 5 GHz frequencies at the same time. This requires a powerful adapter and command works slower. Because when you are sniffing on 2 bands, you have a lot of channels that airodump-ng has to hop on all of them and discover clients and networks broadcasting on these channels.

**airodump-ng --band abg wlan0**

**Running airodump-ng just for the target, gather more information about it:**

**airodump-ng --bssid 00:11:22:33:44:55**(target MAC) **--channel 2**(target channel) **--write test**(this command will store all the data that will be gathered in a file named test) **wlan0**

YOU CAN SEE THE NETWORK INFORMATIONS THAT YOU WANT TO SNIFF DATA ON AND ALSO YOU CAN SEE NEW SECTION AT THE BOTTOM. THESE ARE THE CLIENTS OR THE DEVICES CONNECTED TO THE TARGET NETWORK.

* STATION: MAC addresses of the clients
* PWR: Signal strength of the devices
* Rate: Speed
* Lost: Amount of data lost
* Frames: Amount of frames or packets that we have captured
* Probe: We can see here if any of these devices are still probing for networks. Sometimes, when you are on airodump-ng against all networks, you’d still see the bottom section and you’d see that some devices are not connected and they’re literally trying or looking for networks. So you’d see the name of the networks that they’re looking for under the probe.

AFTER THIS COMMAND YOU WILL HAVE TEST (test-01) FILES IN YOUR CURRENT WORKING DIRECTORY THAT CONTAIN THE DATA THAT YOU JUST CAPTURED. THEY HAVE DIFFERENT EXTENSIONS. THE MAIN FILE WE ARE GONNA BE USING IS THE .cap FILE. THIS FILE CONTAINS THE DATA THAT WE CAPTURED DURING THE PERIOD THAT airodump-ng WAS WORKING.

.cap FILE SHOULD CONTAIN EVERYTHING THAT WAS SENT TO AND FROM MY TARGET NETWORK. SO IT SHOULD CONTAIN: URLs, CHAT MESSAGES, USERNAMES, PASSWORDS, OR ANYTHING THAT ANY OF THESE DEVICES (connected to target network, at the bottom) DID ON THE INTERNET.

Problem is, target network uses encryption. So all of the data between the router and the clients is encrypted.

We are gonna use the tool WIRESHARK to analyze the data.

**wireshark** ---> opens wireshark

In wireshark, open the .cap file.

We can’t read anything that we captured.

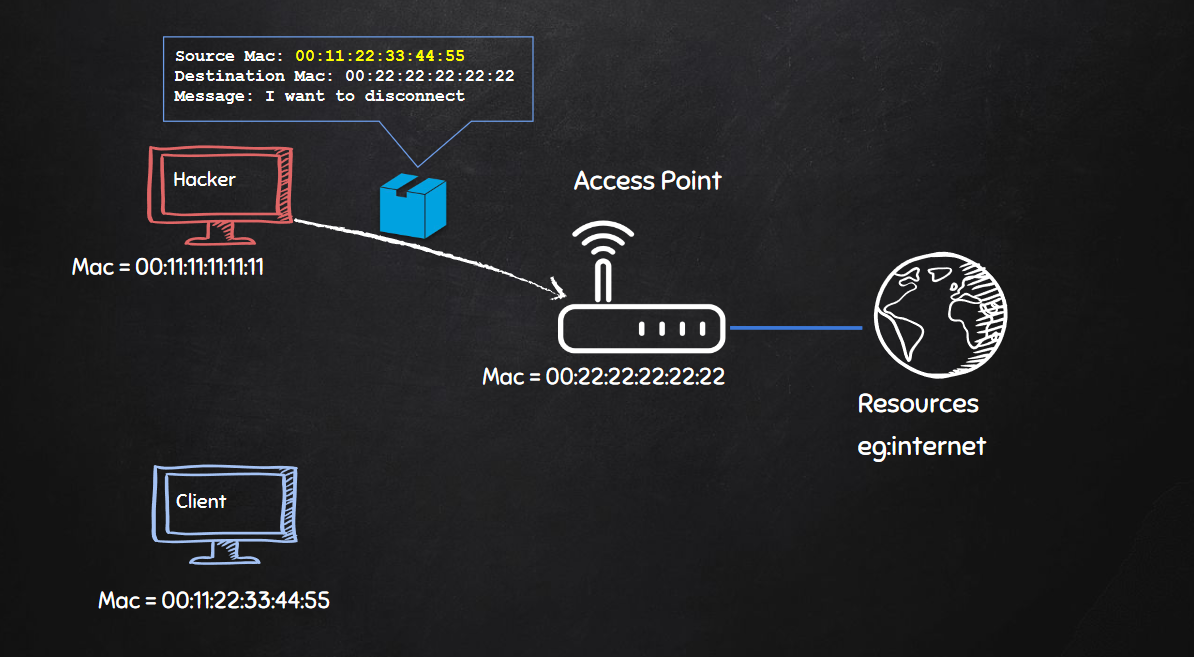
You can see here the name of the device’s (which is connected to the network) manufacturer, and it’s MAC address.

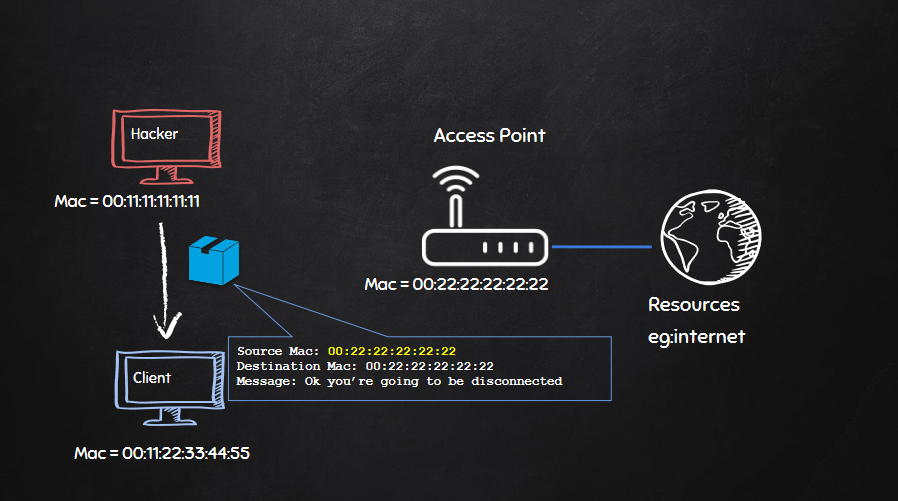
You can see router or device in source part.

***DEAUTHENTICATION ATTACK***

Disconnect any client from any network.

* Works on encrypted networks (WEP, WPA&WPA2)
* No need to know the network key, password.
* No need to connect to the network.





1. We changed our MAC address to the MAC address of targetted client to pretend to be the client that we want to disconnect. Then tell the router that “I want to disconnect from you”.
2. Then we are gonna pretend to be the router by changing our MAC address to the router’s MAC address and tell the client that “You are requested to be disconnected, so I am gonna disconnect you ”.

This will allow us to successfully disconnect, or deauthenticate any client from any network.

*HOW TO DO THIS?*

**aireplay-ng --deauth 10000000**(number of deauthentication packets that I want to send, large number so that it keeps sending these packets to both the router, and the target device, therefore I’ll disconnect my target device for a very long time. Only way to get it back to connect is to hit ctrl+c) **-a [MAC of the router] -c [MAC address of the target] wlan0**(wireless adapter in monitor mode)

In very rare cases this command fails unless airodump-ng is running against the target network. So first we run airodump-ng:

**airodump-ng --bssid [MAC of the router] --channel 5**(5 in this case) **wlan0**

(There is no need to “write”)

**GAINING ACCESS**

**WEP (WIRED EQUIVALENT PRIVACY) CRACKING**

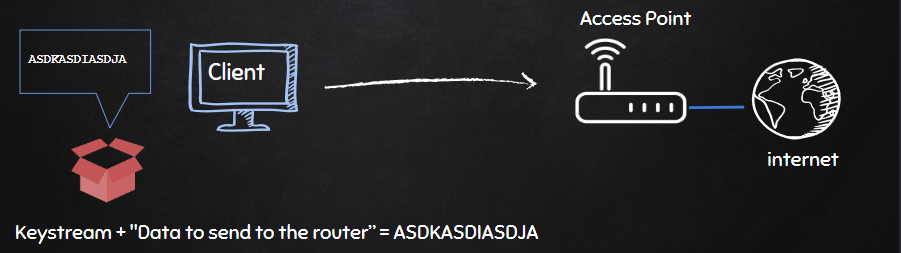
* Old but still used in some networks.
* Can be cracked easily.
* Uses an algorithm called RC4 to encrypt its data.
* Client encrypts data using a key.
* Encrypted packet sent in the air.
* Router decrypts packet using the key.
* THE SAME HAPPENS IF THE ROUTER WANTS TO SEND STH BACK TO THE CLIENT.

(Anybody who captures the packet in the middle can’t see the content of the packet because he/she doesn’t have the key.)

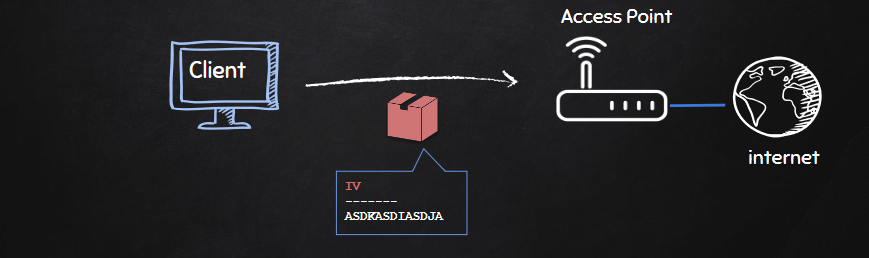


The way RC4 works is actually fine, the problem is with the way that WEP implements this algorithm.

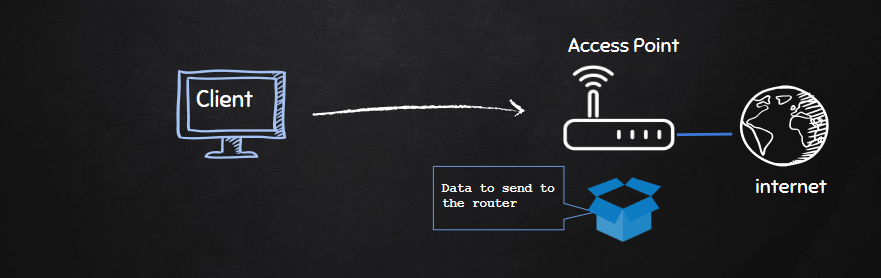
* Each packet is encrypted using a unique key stream (64 bit or 128 bit).
* In order to encrypt the packet, WEP tries to generate a unique key for each packet. So literally each packet that sent into the air, WEP tries to create a new unique key for it.
* Random initialization vector (IV) is used to generate the keys streams.
* The initialization vector is only 24 bits. It is random for each packet.
* IV + key (password) = key stream ---> The initialization vector is added to the password of the network (to the actual key that people use to connect to the network). Then the key stream that was generated is used to encrypt the packet.



Before sending the gibberish (encrypted packet) into the air, WEP will also append the initialization vector. This is the 24 bit random number, that we mentioned, it creates in order to make sure that each packet has a unique key. The reason why it adds the initialization vector to the packet is because once the router receives the packet, it needs to be able to decrypt it, and to decrypt it, it needs the key and the IV. The router already has the key, so there is no need to send that. Therefore we just need to send the IV.



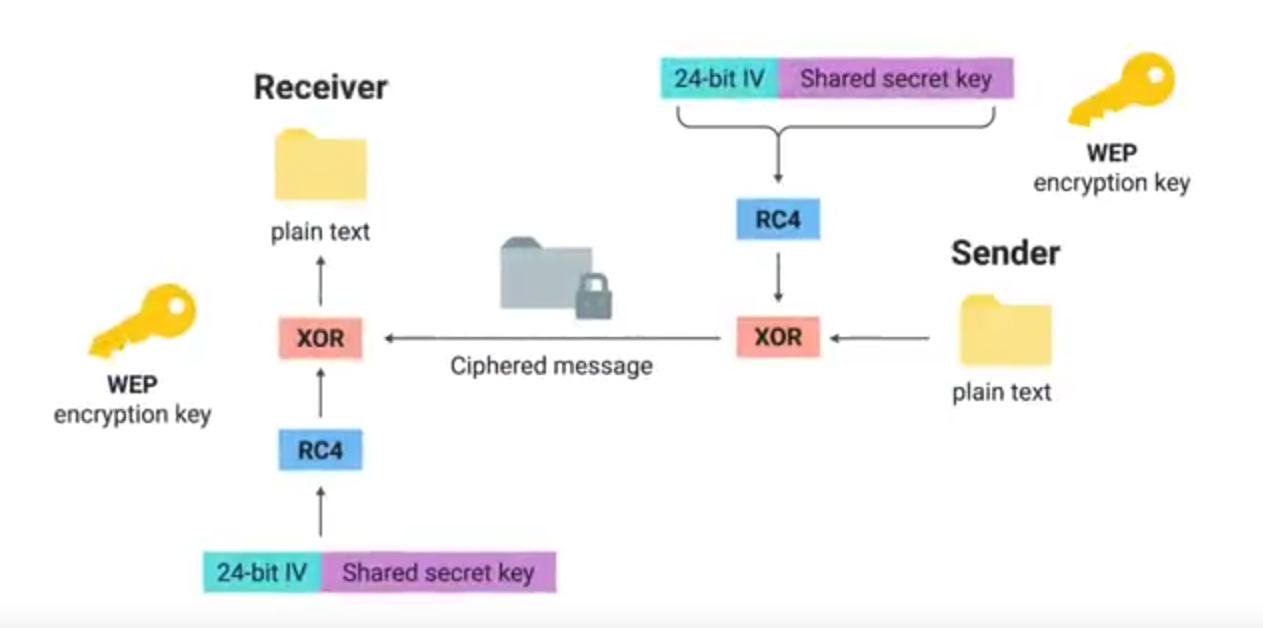
Then router can generate a key stream and then use that key stream to transform the gibberish into its original form and read the packet.



IV is too small (only 24 bits) and IV is sent in plain text. If someone captures the packet, they won’t be able to read the “Data to send to the router.” but they will be able to read the IV in plain text.

Result:

* IV’s will repeat on busy networks.
* This makes WEP vulnerable to statistical attacks.
* Repeated IVs can be used (with a tool named aircrack-ng) to determine the key stream once we have enough repeated IVs and from that aircrack-ng will also be able to crack WEP, and give us the key to the network break the encryption.



+

IV

In our packet, we have plain text. Almost every packet includes 0xaa value so we know there is 0xaa in our cipher text as a plain text.

Also we have IV of this packet (we can choose it in the packet because it repeats in about every 5000 packets) and we have cipher text which is created by XORing the keystream and the plain text.

Then we can use the aircrack-ng to found the keystream. We do it by XORing the ciphertext and plaintext and we get keystream.

When we found the keystream we can figure out the password because keystream is the combination of IV and the password and we know the IV.

*To crack WEP we need to (break into a BUSY network):*

* Capture a large number of packets/IVs. Because IVs are short, they will be repeated.

(using **airodump-ng**)

* Analyse the captured IVs and crack the key.

(using **aircrack-ng**)

We have adapter in monitor mode again.

**airodump-ng wlan0** ---> find the network using WEP

**airodump-ng --bssid [MAC of target router] --channel [channel of target router] --write basic\_wep wlan0**

You will see that data and frames are increasing really fast. You see under the #Data column the number of useful packets that contain a different IV that we can use in order to crack the key. The higher the number is, the more likely we will be able to crack the key.

**aircrack-ng basic\_wep-01.cap** ---> Find the key in the formats: MAC type / ASCII

ASCII is not always there so we’re gonna use MAC type.

Suppose that key is ---> 41:73:32:33:70

Remove the colons ---> 4173323370

Connect from your host machine. You can connect from Kali but when we enabled monitor mode, we killed a lot of processes and sometimes even after you restart these processes, getting connecting to your target will be a little buggy so it’s best to literally just restart Kali and connect again. So just to save all of this time, we’re gonna connect from host machine.

Just connect to the target network with using the 4173323370 as password.

So all we had to do is capture enough data and then run aircrack-ng to crack the encryption and give me the key.

*If the network is NOT BUSY:*

* It would take some time to capture enough IVs.
* We should force the Access Point to generate new packets with new IVs.

PROBLEM

* APs only communicate with connected clients.
* We can’t communicate with it.
* We can’t even start the attack

SOLUTION (FAKE AUTHENTICATION)

* We need to tell this network that we want to communicate with it (Associate with the AP before launching the attack.); because by default, APs ignore any requests they get unless the device has connected to this network or associated (not connecting) with it.
* With association we are just telling the target network “look, I want to communicate with you, don’t ignore my requests”. This is same with when you just click on the network when you want to connect to it. You still haven’t put the password, you’re just telling the target network “I want to communicate with you, please don’t ignore me”.

**airodump-ng --bssid [Target router MAC] --channel [Target router channel] --write arpreplay wlan0**

To associate with this network:

**aireplay-ng --fakeauth 0**(we only want to do this once) **-a [Target router MAC]** **-h [Wireles adapter MAC] wlan0**

MAC of wireless adapter is first 12 digits of the unspec field in **ifconfig**. Usually you see the address after the ether but when you enable monitor mode it’ll show up like so. Don’t forget to replace minuses with colons.

After the last command, I am associated with the target network and if I send it anything, it will accept and communicate with me. AGAIN, I am not connected to the network, I still can’t use the internet, I am literally just associated with the network so I can communicate with it.

We can start communicating with it, and it won’t ignore us.

Now we can go and start injecting packets into the traffic to force the access point to generate new packets with new IVs. This will increase the number of data quickly.

***ARP REQUEST REPLAY ATTACK***

* Wait for an ARP packet.
* Capture it, and replay it (retransmit it).
* This causes the AP to produce another packet with a new IV.
* Keep doing this till we have enough IVs to crack the key.

Once we have enough data, enough IVs, we can run aircrack-ng.

In short, we are gonna inject packets into the traffic.

The airodump-ng is running and we have associated with the network.

Now we will run the ARP replay attack in order to inject packets into the traffic and force the router to generate new packets and increase the number of data.

**aireplay-ng --arpreplay -b [Target router MAC]** **-h [Wireles adapter MAC] wlan0**

Our wireless adapter is gonna wait for an ARP packet after this command. Once there is an ARP packet transmitted in this network, adapter is gonna capture it and retransmit it. Once it does that, the AP will be forced to generate a new packet with a new IV and we’ll keep doing this, forcing the AP to continually generate new packets with new IVs.

**aireplay-ng --fakeauth 0**(time to wait between association attempts**)** **-a [Target router MAC]** **-h [Wireles adapter MAC] wlan0** ---> to play it safe

**aircrack-ng arpreplay-01.cap**

This may require more data packets if network uses 128 bit key.

We said 0 because we only want to do this once.

**WPA AND WPA2 (Wi-Fi PROTECTED ACCESS) CRACKING**

* Only difference between them is the encryption used to ensure message integrity.
* Both can be cracked using the same methods.
* Each packet is encrypted using a unique temporary key.
* Packets contain no useful information.
* WPA uses TKIP.
* WPA2 uses CCMP.

If a feature is enabled and misconfigured (uses PIN), can be exploited to recover the key without having to crack the actual encryption. This feature is WPS (Wi-Fi PROTECTED SETUP). It was designed to simplify the process of connecting printers and such devices. You can actually see a WPS button on most wireless-enabled printers. If this button is pressed and then you press the WPS button on the router, you’ll notice that the printer will connect to the router without you having to enter the key. This way, the authentication is done using an eight-digit PIN. We are just exploiting a feature (WPS) that can be enabled on WPA/WPA2 encryptions.

* WPS is a feature that can be used with WPA & WPA2.
* Allows clients to connect without the password.
* Authentication is done using an 8 digit pin.
  + 8 Digits is very small.
  + We can try all possible pins in relatively short time.
  + Then the WPS pin can be used to compute the actual password.

PS: This only works if the router is configured not to use PBC (Push Button Authentication).

It needs to be configured to use a normal PIN authentication and not a Push Button Authentication. If PBC is used, then the router will refuse any PINs that we try unless the WPS button is pressed on the router.

The only difference between WPA and WPA2 is the algorithm used to encrypt the information but both encryptions work in the same way.

***CRACKING IF WPS ENABLED***

**wash --interface wlan0** ---> shows all the networks around me that have WPS enabled

* Vendor: Vendor of the hardware used in the AP.
* Lck: Tell us whether WPS is locked or not. Sometimes WPS is lockes after a number of failed attempts. If “No”, we can actually go ahead and try to guess the PIN.
* WPS: Version of WPS.
* dBm: Signal strength.
* Ch: Channel.

If our target network shows up, so there’s a good chance that this attack will work against it. The only reason it might fail is if the target uses PBC. The only way to know if it uses PBC is trying.

**reaver --bssid [Router MAC] --channel [Router channel] --interface wlan0 -vvv**(To show us as much information as possible. If it fails or things go wrong, we’ll be able to know what’s happening.) **--no-associate**(To tell reaver not to associate with the target network because we will already manually do that. Reaver can do it for you if you wish but it fails a lot.)

Bruteforce the PIN, once it found the PIN, it will use it to compute the actual WPA key.

If you have a bug (so many send\_packet), they still haven’t fixed this bug in the latest version of reaver. So it’s better to use older one.

1. Download the old reaver version. **--->** [**https://ufile.io/lro4nkdv**](https://ufile.io/lro4nkdv)
2. Go to the downloads (or to where you download the old version).
3. **chmod +x reaver --->** This will make reaver an executable file.
4. **./reaver --bssid [Router MAC] --channel [Router channel] --interface wlan0 -vvv** **--no-associate**

**aireplay-ng --fakeauth 30 -a [Router MAC] -h [Adapter MAC] wlan0**

We did this because when I run attack, the network will start accepting the PINs and not ignore me. We associate with the target network every 30 seconds.

We did these after the reaver because otherwise, aireplay-ng will fail to associate with my network because we say no associate in reaver.

We said please don’t ignore us so that reaver can bruteforce the PIN.

THEN REAVER WILL FIND THE PIN AND CONNECT TO THE NETWORK. THEN SEND QUERY TO THE ROUTER FOR THE ACTUAL PASSWORD.

***CRACKING IF WPS DISABLED OR ENABLED BUT CONFIGURED TO USE PBC***

Keys are unique, temporary and much longer than what they were in WEP.

We can’t use packets to crack the key because they have no useful data.

Only packets that can aid with the cracking process are the handshake packets. These are 4 packets sent when a client connects to the network.

1. **airodump-ng wlan0** ---> to copy the MAC
2. **airodump-ng --bssid [Router MAC] --channel [Router channel] --write wpa\_handshake wlan0**

Now you’ll wait until a new client connects to network because you need handshake packets.

You can do deatuhentication attack to speed it up.

1. **aireplay-ng --deauth 4 -a [Router MAC] -c [Client MAC] wlan0**

When you do this, you can see handshake will be captured in the airodump-ng screen.

Then you can quit the airodump-ng.

The handshake that we captured does not contain data the helps recalculate the key.

It contains data that can be used to check if a key (we bruteforced) is valid or not.

We are gonna create a text file that contains a large number of passwords. Then use them with the handshake in order to check whether this password is valid or not.

**CREATING A WORDLIST**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **crunch** | **[Min]** | **[Max]** | **[Characters]** | **-t [Pattern]** | **-o [FileName.txt]** |
| (Necessary)  Tool name | (Necessary)  Minimum number of characters | (Necessary)  Maximum number of characters | (Necessary)  Characters that you want to generate words from | (Optional)  To give a pattern. If you know password starts with a, you can tell the password start with a and then give me all possible combinations | (Optional)  To specify the file name where the words are gonna be stored |

For example:

**crunch 6 8 123abc$ -t a@@@@b -o wordlist**

Generated passes:

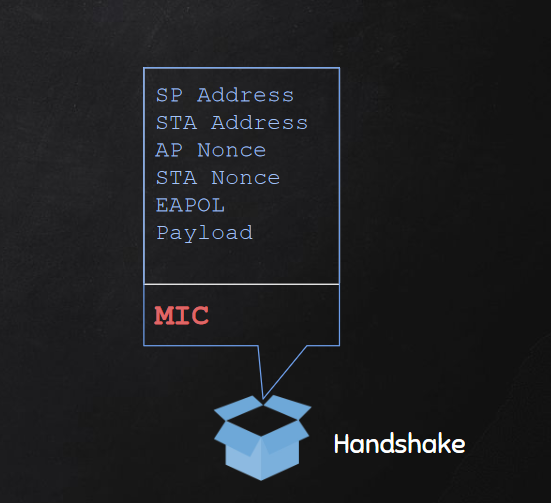
* aaaaab
* aabbbb
* aan$$b

1. **crunch 10 12 ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890 -o test.txt**

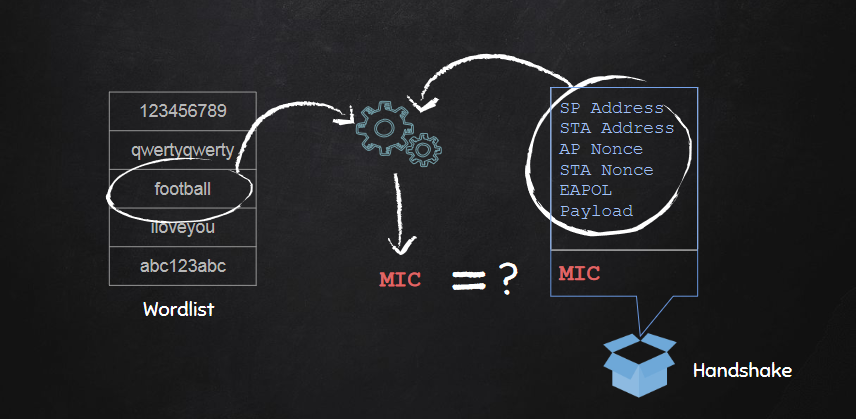
Two things needed to crack WPA/WPA2:

* 4-way handshake
* Wordlist

aircrack-ng is gonna unpack the handshake and extract the useful information.

MIC (Message Integrity Code) is what’s used by the AP to verify whether a password is correct or not.

So, aircrack-ng is gonna separate this and put it to the side, and then it’s gonna use all of the other information (blue ones) combined with the first password from the wordlist to generate an MIC. Then, it’s gonna compare this MIC to the one that’s already in the handshake. If the MIC generated using the blue information + the first password is same with the MIC in the handshake, then the password used to generate our MIC is the password of the network. Otherwise this password is wrong, and it will move to the next password.



1. **aircrack-ng wpa\_handshake-01.cap -w test.txt**

aircrack-ng will try each word in the test file one by one

**POST-CONNECTION ATTACKS**