Introducción a TCP/IP

Algunas características de TCP e IP

por Toad, Febrero de 2005 (Licencia)

Partes

Algo sobre TCP/IP

TCP/IP es la base de Internet. Sus orígenes se remontan a 1973 y fue desarrollado para un proyecto patrocinado por el ARPA (Estados Unidos).

Aunque TCP/IP es un conjunto de protocolos, los que les dan nombre al conjunto son **Transmission Control Protocol (TCP)** e **Internet Protocol (IP)**.

TCP se encarga de controlar, ordenar y manejar los datos y de asegurarse de que llegan a su destino sin que se pierdan paquetes. Pertenece a la capa de Transporte del protocolo TCP/IP.

IP se encarga de enviar los paquetes a su destino. Pertenece a la capa Internet de TCP/IP.

Veamos algunas características de estos protocolos en detalle:

Transmission Control Protocol (TCP)

TCP actúa tanto como emisor como receptor de paquetes. El TCP emisor divide la información a transferir en paquetes y envía cada uno por separado, retransmitiéndolos si alguno no se entrega correctamente. Si es necesario el TCP receptor tendrá que reordenar los paquetes secuencialmente (cada paquete tiene un número de secuencia de 32 bits, que es un campo de la cabecera) para poder obtener la información.

Otros campos de la cabecera son *Source port (Puerto fuente)* y *Destination port (Puerto destino)* que son números sin signo (unsigned) de 16 bits. TCP usa los números de puerto para identificar la aplicación emisora y la aplicación receptora. Hay puertos TCP de 0 a 65535, sin embargo podemos distinguir tres tipos de puertos:

- **Puertos conocidos:** De 0 al 1023, están reservados por la IANA para determinado tipo de aplicaciones (servidor HTTP, FTP, etc.) y se requiere de privilegios de administrador en una máquina para activar una aplicación en uno de estos puertos. Un ejemplo es el puerto 80(HTTP).
- **Puertos registrados:** de 1024 a 49151, reservados para aplicaciones concretas. Un ejemplo es el 3306(MySQL).
- **Puertos privados:** de 49152 a 65535, estos no están reservados para ninguna aplicación concreta.

Podemos decir que un puerto puede estar en tres estados:

- **Puerto filtrado:** Un firewall (cortafuegos) bloquea el acceso al puerto.
- **Puerto cerrado:** El puerto no está bloqueado pero no hay ninguna aplicación escuchando en él.
- **Puerto abierto:** El puerto no está bloqueado y hay una aplicación escuchando en él.

A la técnica de detectar los puertos abiertos de una máquina se le denomina *Escaneo de puertos* (Port scan), que se hace con fines de seguridad pero también proporciona mucha información de utilidad a un posible atacante de nuestra máquina, ya que le permite saber qué aplicaciones tenemos activas para explotar algún fallo y conseguir el acceso.

En la cabecera de los paquetes de TCP hay 6 *flags* de 1 bit, es decir, que pueden valer ó 0 ó 1 según estén desactivadas o activadas: estas banderas son **SYN, ACK, RST, PSH, URG** y **FIN**. Después los analizaremos con detalle.

Si se usa el flag **ACK** en un paquete se incluye un campo en la cabecera del paquete con el valor TCP ACK, un número de 32 bits.

Veamos un ejemplo práctico utilizando la utilería `hping` [1] en un sistema UNIX:

```
# hping2 -c 1 -V -S -p 80 www.apple.com
using en0, addr: 192.168.1.22, MTU: 1500
HPING www.apple.com (en0 17.254.0.91): S set, 40 headers + 0 data bytes
len=46 ip=17.254.0.91 ttl=48 DF id=8071 tos=0 iplen=44
sport=80 flags=SA seq=0 win=32768 rtt=813.8 ms
seq=3342449005 ack=466630552 sum=691a urp=0


--- www.apple.com hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 813.8/813.8/813.8 ms
```

Ejemplo práctico 1

En el ejemplo práctico anterior vemos lo siguiente:

- Enviamos al puerto 80 de la máquina `www.apple.com` un paquete con el flag TCP **SYN** activado (`S set`). La información relativa a este paquete está en color rojo.
- La máquina nos devuelve otro paquete, con los flags **SYN** y **ACK** activados (`flags=SA`). El *Source Port* es el 80. Su número de secuencia TCP de 32 bits es `3342449005`. El valor TCP ACK es `466630552`. La información de este paquete está en color azul.
- Podemos ver que el "Tamaño de Ventana TCP" (TCP Window Size) especificada en la cabecera es de `32768` bytes. A continuación veremos qué es esto.

A medida que vayamos avanzando en el documento veremos qué significan más datos de los que podemos ver en la salida del comando.

El Tamaño de Ventana TCP

El tamaño de ventana TCP (TCP Window Size) es un valor que especifica el receptor TCP en la cabecera, y especifica la cantidad máxima de información en bytes que el receptor está preparado para recibir. El estándar TCP dice que el receptor tiene que ser capaz de aceptar una ventana completa de datos. Es la forma de realizar un control de flujo en TCP.

Iniciar una conexión en TCP

Para que una aplicación A inicie una conexión con otra máquina B al puerto X se sigue un protocolo de acuerdo de tres pasos:

1. La aplicación A envía un paquete con el flag **SYN** al puerto X de la máquina B. Se utilizará un número de secuencia inicial especificado por A.

   ```
   A ------ [SYN secuencia:X] ----------------> B
   ```

2. Si el puerto está abierto, B responde a A con un paquete con los flags **SYN** y **ACK** activos. B también especifica su número de secuencia inicial, y en el valor TCP ACK incluye el número de secuencia de A incrementado en una unidad:

   ```
   B ------ [SYN+ACK secuencia:Y ack:X+1] ----> A
   ```

3. La aplicación A envía al puerto X de B otro paquete con el flag **ACK**, confirmando el inicio de la conexión. Como número de secuencia utiliza su número de secuencia inicial incrementado en 1, y en el valor TCP ACK incluye el número de secuencia de B incrementado en 1:

```
A ------ [ACK secuencia:X+1 ack:Y+1] ------> B
```

La conexión se inicia entonces. A este proceso de tres pasos se le denomina *three-way handshake*.

Un ejemplo práctico de lo de los número de secuencia y TCP ACK:

```
# hping2 -V -c 1 -p 21 -S -M 1000 192.168.1.9
using en0, addr: 192.168.1.22, MTU: 1500
HPING 192.168.1.9 (en0 192.168.1.9): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=64 DF id=6095 tos=0 iplen=44
sport=21 flags=SA seq=0 win=65535 rtt=0.7 ms
seq=599635100 ack=1001 sum=3397 urp=0


--- 192.168.1.9 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.7/0.7/0.7 ms
```

En el ejemplo enviamos un **SYN** a 192.168.1.9 con el número de secuencia inicial de 1000 (en rojo).

192.168.1.9 nos responde con un paquete **SYN+ACK**, y en el campo TCP ACK especifica nuestro número de secuencia inicial incrementado (en azul).

También especifica su propio número de secuencia, marcado en verde.

Cuando se hayan terminado de transferir los datos, se envía un paquete con el flag **FIN** para finalizar la conexión.

Seguridad: SYN Cookies

El sistema mantiene una tabla donde se indica el estado de las conexiones.

En el three-way handshake, cuando la máquina A envía al puerto X de B un **SYN**, el estado de la conexión para la máquina B es SYN_RECV (o SYN_RCVD). Después envía el **SYN+ACK** a A y espera a que A conteste con un ACK.

Cuando A contesta con el **ACK** para establecer la conexión, el estado pasa a ser ESTABLISHED, hasta que se termine la conexión.

Si A no contesta con el **ACK** (por ejemplo, debido a un error de conexión) pasado un tiempo, B elimina la conexión de la lista.

Hasta aquí todo parece normal ¿no?...

Pero ¿qué ocurriría si alguien envía toneladas de paquetes **SYN** con direcciones IP falsas a B? La tabla de conexiones de B empezaría a llenarse de SYN_RECV, y cuando se llene dejaría de poder recibir conexiones hasta que el ataque finalice, por lo que se produciría una **negación de servicio (DoS)**.

Este ataque se conoce como **SYN flood** y se puede evitar.

Para evitarlo existe una técnica denominada SYN Cookies [2], que hace que aunque la tabla de conexiones esté llena se envíe un **SYN+ACK** a los emisores de SYNs como si no lo estuviese. Funciona así:

En este **SYN+ACK** se especifica un número de secuencia TCP especial hecho a través de un *hash* con la IP del emisor, puerto origen y más datos (la "cookie").

Cuando el emisor legítimo conteste con un **ACK**, como hemos visto antes, especificará como valor TCP ACK nuestro valor de secuencia incrementado en una unidad. El receptor comprobará entonces este valor, que en realidad es el *hash* de antes incrementado con los datos IP, puerto y demás, con lo que se abrirá la conexión; como si la tabla no estuviese llena y todo funcionase normal.

A continuación veremos el significado de cada uno de los flags o bits de los paquetes TCP

Los flags de la cabecera en los paquetes TCP

Hemos visto antes que hay los flags **SYN, ACK, RST, PSH, URG** y **FIN**. Los veremos con detalle:

- **SYN (Synchronize):** Ya hemos visto que se utiliza para iniciar una conexión TCP. Si enviamos un paquete con este flag a un puerto X de una máquina, pueden pasar varias cosas:

  Si el puerto está abierto, la máquina nos responde con un paquete con los flags **SYN** y **ACK** como hemos visto antes.
  Si el puerto está cerrado, nos responde con un paquete con los flags **RST** y **ACK**.
  Si el puerto está filtrado, no recibiremos nada.

  Veamos unos ejemplos prácticos:

```
# hping2 -c 1 -S -p 21 192.168.1.9
HPING 192.168.1.9 (en0 192.168.1.9): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=64 DF id=37670 sport=21 flags=SA seq=0 win=32768 rtt=0.4 ms

--- 192.168.1.9 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.4 ms
```

  En este ejemplo vemos cómo enviamos un paquete con el flag SYN al puerto 21 de `192.168.1.9` y nos responde con otro con los flags SYN y ACK. El puerto 21 de `192.168.1.9` está abierto.

```
# hping2 -c 1 -S -p 80 192.168.1.9
HPING 192.168.1.9 (en0 192.168.1.9): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=64 id=38005 sport=80 flags=RA seq=0 win=0 rtt=0.6 ms

--- 192.168.1.9 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

  En este otro ejemplo vemos que al enviar un paquete con flag SYN al puerto 80 de la misma máquina, nos responde con otro con flags RST y ACK. El puerto 80 de esa máquina está cerrado.

```
# hping2 -c 1 -S -p 3306 82.223.41.40
HPING 82.223.41.40 (en0 82.223.41.40): S set, 40 headers + 0 data bytes

--- 82.223.41.40 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

  En este último ejemplo vemos que al enviar un paquete con flag SYN al puerto 3306 de `82.223.41.40` no recibimos respuesta; el puerto está filtrado (o bien la máquina está caída, pero podríamos comprobarlo).

- **ACK (Acknowledgement):** Este *flag* se utiliza para confirmaciones; y como hemos visto antes al utilizarlo se incluye un valor TCP ACK en la cabecera.
  Si enviamos un paquete ACK "no solicitado" a un puerto X de una máquina, debería ser respondido con otro paquete con flag RST, sea cual fuere el estado del puerto. De hecho hay firewalls que sólo filtran los paquetes de intento de conexión con SYN, no filtrando los que llevan ACK.

```
# hping2 -c 1 -A -p 21 192.168.1.9
HPING 192.168.1.9 (en0 192.168.1.9): A set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=64 id=38407 sport=21 flags=R seq=0 win=0 rtt=0.7 ms

--- 192.168.1.9 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.7/0.7/0.7 ms
#
# hping2 -c 1 -A -p 80 192.168.1.9
HPING 192.168.1.9 (en0 192.168.1.9): A set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=64 id=38408 sport=80 flags=R seq=0 win=0 rtt=0.6 ms

--- 192.168.1.9 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

  En el ejemplo anterior enviamos paquetes con flag ACK "no solicitados" a la misma máquina de antes, y vemos que sea cual sea el estado de los puertos nos responde con un paquete con flag RST.
  Lo mismo ocurre al enviar un SYN+ACK no solicitado.

- **RST (Reset):** Este bit se utiliza para reiniciar una conexión debido a paquetes corrompidos o a SYN duplicados, retardados, etc.
  Un paquete con flag RST no solicitado es simplemente ignorado:

```
# hping2 -c 1 -R -p 21 192.168.1.9
HPING 192.168.1.9 (en0 192.168.1.9): R set, 40 headers + 0 data bytes

--- 192.168.1.9 hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

- **PSH (Push):** Se utiliza para forzar el enviado inmediato de los datos tan pronto como sea posible. Si el TCP emisor envía un paquete con este flag activado, el TCP receptor sabe que tiene que entregar los datos inmediatamente a la aplicación receptora sin ponerlo en un buffer y esperar a más datos.

- **URG (Urgent):** Sirve para definir un bloque de datos como "urgente". El campo *Urgent Pointer* de la cabecera es un puntero que "apunta" a donde terminan estos datos urgentes; de esta manera se marca el bloque de datos.

- **FIN (Finalize):** Sirve para finalizar una conexión.

Internet Protocol (IP)

Ya hemos dicho que este protocolo se encarga de la transmisión de los paquetes de su origen a su destino, puede que recorriendo varias redes.

TCP toma los datos y los fragmenta en paquetes de no más de 64K. IP transporta esos datagramas, fragmentándolos en trozos más pequeños, a su destino, donde TCP los reordena y ensambla para volver a formar los datos originales.

Los datagramas IP tienen una **cabecera**, con una parte fija de 20 bytes y una parte variable. Esta cabecera tiene varios campos:

El campo *VERS*, Versión

Especifica la versión del protocolo, al que pertenece el datagrama. Por ejemplo, 4 (0100) para IPv4 o 6 (0110) para IPv6. Tiene un tamaño de 4 bits.

El campo *HLEN*, Longitud de la Cabecera

Especifica la longitud de la cabecera, en palabras de 32bits, de forma que si este campo tiene el valor de 7 (0111) significa que la longitud de la cabecera es de 7 palabras de 32bits, es decir, de 224 bits. El valor mínimo es 5 (0101) y el máximo es 15 (1111). El tamaño de este campo es de 4 bits.

El campo *TOS*, Tipo de Servicio

Especifica el tipo de servicio con respecto a la seguridad, velocidad, fiabilidad... Su tamaño es de 8 bits.

- Los primeros 3 (del 0 al 2) bits de este campo especifican la "preferencia", el tipo de tráfico que es prioritario para el tipo de servicio que queremos.
- El bit 3 (Delay), si está activo el paquete solicita un bajo "retardo" o "retraso" en la transferencia.
- El bit 4 (Throughput), si está activo el paquete solicita alto "rendimiento" en la transferencia.
- El bit 5 (Reliability), si está activo el paquete solicita alta fiabilidad en la transferencia.
- El bit 6 y el 7 están reservados.

Por ejemplo, en un tipo de servicio como una transferencia de archivos puede interesarnos sobre todo la fiabilidad, que los datos lleguen enteros, por lo que podemos activar el bit 5. En cambio en una videoconferencia puede interesarnos más la velocidad, por lo que podemos activar el bit 3.

El campo Longitud Total

La longitud total de todo el datagrama, incluidos los datos. La máxima longitud es 65535 bytes. El tamaño de este campo es 16 bits.

El campo ID. Implementaciones inseguras.

Este campo de 16 bits permite asignarle un número de identificación (ID) único a cada datagrama, con el objetivo de permitir reensamblar el datagrama al dividirlo en fragmentos más pequeños.

Como el campo es de 16 bits, el valor máximo que puede tomar es 65535 (1111111111111111)

Muchos sistemas operativos implementan esto con un contador global que se suma uno por cada paquete enviado, y al llegar a 65535 vuelve a 0. Esto a primera vista puede parecer buena idea (simple y rápida) pero en realidad es nefasta, ya que le permite saber a cualquiera el número de paquetes que está enviando la máquina.

Un ejemplo de máquina sometida a este "bug":

```
# hping2 -c 3 -i 1 -S -p 80 xxxx.xxx.xxxx.xx.jp
HPING xxxx.xxx.xxxx.xx.jp (en0 xxx.xxx.xxx.22): S set, 40 headers + 0 data bytes
len=46 ip=xxx.xxx.xxx.22 ttl=232 DF id=42457 sport=80 flags=SA seq=0 win=9112 rtt=428.4 ms
len=46 ip=xxx.xxx.xxx.22 ttl=232 DF id=42458 sport=80 flags=SA seq=1 win=9112 rtt=432.3 ms
len=46 ip=xxx.xxx.xxx.22 ttl=232 DF id=42459 sport=80 flags=SA seq=2 win=9112 rtt=438.1 ms

--- xxxx.xxx.xxxx.xx.jp hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 428.4/432.9/438.1 ms
```

Otro ejemplo de cómo podemos saber el tráfico que está enviando una máquina con este "bug":

```
# hping2 -c 4 -i 1 -S -p 80 xyz.xyz.be
```

```
HPING xyz.xyz.be (en0 217.xxx.xxx.224): S set, 40 headers + 0 data bytes
len=46 ip=217.xxx.xxx.224 ttl=117 DF id=61881 sport=80 flags=SA seq=0 win=16616 rtt=129.2 ms
len=46 ip=217.xxx.xxx.224 ttl=117 DF id=61963 sport=80 flags=SA seq=1 win=16616 rtt=135.0 ms
len=46 ip=217.xxx.xxx.224 ttl=117 DF id=62014 sport=80 flags=SA seq=2 win=16616 rtt=130.9 ms
len=46 ip=217.xxx.xxx.224 ttl=117 DF id=62110 sport=80 flags=SA seq=3 win=16616 rtt=128.7 ms

--- xyz.xyz.be hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 128.7/131.0/135.0 ms
```

Vemos, por ejemplo, que entre el primer paquete de respuesta y el segundo ha enviado otros 82 paquetes.

Como media la máquina `xyz.xyz.be` está enviando unos 76.3 paquetes por segundo (la opción `-i 1` envía un paquete por segundo).

De todas formas muchos sistemas implementan soluciones a este problema. Solaris, por ejemplo, mantiene un contador diferente para cada IP. OpenBSD, por su parte, genera números ID aleatorios (*ID randomization*).

La solución de OpenBSD se puede aplicar al kernel de Linux. [3]

Y por si fuera poco, las máquinas susceptibles a este bug pueden ser usadas como "zombies" para un tipo de escaneo de puertos avanzado llamado *Idle Scan*, que veremos más adelante a modo de curiosidad.

El campo de flags

Este es un campo de 3 bits, el primero de los cuales no se usa y vale siempre 0.

El segundo bit es el **DF**, y significa *no fragmentar*. Si un datagrama tiene este bit activo, los *routers* no lo fragmentarán en pedazos más pequeños.

Un ejemplo de paquete recibido con este bit:

```
# hping2 -c 1 -S -p 21 192.168.1.9
HPING 192.168.1.9 (en0 192.168.1.9): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.9 ttl=64 DF id=18938 sport=21 flags=SA seq=0 win=65535 rtt=0.6 ms

--- 192.168.1.9 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

El tercer bit, **MF**, debe estar activo en todos los fragmentos de un datagrama excepto en el último. Sirve para verificar que el datagrama llega completo a su destino.

El campo *Fragmentation Offset*

Este campo de 13 bits especifica la posición a la que pertenece el fragmento en el datagrama. Se especifica en unidades de 8 bytes, ya que ésta es la unidad básica de fragmentación. El valor máximo es 8192.

El campo *TTL*, Tiempo de Vida

Este campo sirve para especificar el número máximo de *routers* que un paquete puede atravesar. Usualmente cada router decrementa en una unidad el Tiempo de Vida, y si llega a 0 el paquete se destruye.

La mayoría de routers, cuando tienen que destruir un paquete por este motivo, informan al emisor mediante un mensaje ICMP TIME_EXCEEDED.

Como curiosidad, en esto se basa la utilidad `traceroute(8)`, que nos indica los routers por donde tiene que pasar nuestro paquete para que llegue a una máquina X.

```
# traceroute www.theos.com
traceroute to zeus.theos.com (199.185.137.1), 30 hops max, 40 byte packets
 1  speedtouch (192.168.1.1)  0.932 ms  11.932 ms  0.366 ms
 2  217.76.136.31 (217.76.136.31)  90.975 ms  89.274 ms  91.068 ms
 3  gi5-0-1.core2.ban.mad.es.arsysinternet.com (217.76.146.141)  84.782 ms  84.514 ms  85.207 ms
 4  if-4-0.core1.mx2-madrid.teleglobe.net (195.219.197.1)  98.621 ms  87.812 ms  90.94 ms
 5  if-1-2.core1.mdi-madrid.teleglobe.net (195.219.133.41)  437.031 ms  463.92 ms  429.405 ms
 6  if-1-0.core1.lhx-london.teleglobe.net (195.219.133.62)  278.759 ms  280.427 ms  277.516 ms
 7  195.219.196.9 (195.219.196.9)  278.31 ms  277.827 ms  276.797 ms
 8  195.219.196.18 (195.219.196.18)  278.885 ms  277.509 ms  278.534 ms
 9  if-3-0.mcore3.njy-newark.teleglobe.net (216.6.57.5)  277.519 ms  286.197 ms  281.412 ms
10  if-4-0.mcore3.laa-losangeles.teleglobe.net (216.6.84.1)  278.307 ms  276.868 ms  282.637 ms
11  if-4-0.core1.sqo-sacramento.teleglobe.net (216.6.84.14)  278.728 ms  288.35 ms  277.553 ms
12  if-1-0.core2.sqo-sacramento.teleglobe.net (64.86.83.222)  278.48 ms  278.375 ms  283.209 ms
13  if-3-0.core2.sep-seattle.teleglobe.net (66.110.64.9)  278.829 ms  300.6 ms  299.145 ms
14  if-6-0.core1.sep-seattle.teleglobe.net (207.45.196.38)  358.712 ms  346.258 ms  479.536 ms
15  if-7-2.core1.vby-burnaby.teleglobe.net (207.45.196.1)  300.184 ms  276.636 ms  275.003 ms
16  ix-1-2.core1.vby-burnaby.teleglobe.net (207.45.196.10)  282.432 ms  280.309 ms  280.933 ms
17  p8-2-s1.bb2.cal1.sprint-canada.net (204.50.128.190)  282.708 ms  277.99 ms  279.338 ms
18  g5-0-0-s1.gw1.cal1.sprint-canada.net (204.50.251.195)  278.492 ms  292.072 ms  278.932 ms
19  z-s0-0-0-24-s1.gw1.cal1.sprint-canada.net (207.107.204.178)  286.639 ms  281.639 ms  281.399 ms
20  pf.openbsd.org (199.185.230.2)  278.402 ms  291.874 ms  275.036 ms
21  zeus.theos.com (199.185.137.1)  324.508 ms  279.76 ms  281.164 ms
```

La idea es sencilla: inicialmente traceroute ajusta el TTL a 1, de forma que el primer router ya devuelva el ICMP TIME_EXCEEDED para anotar su IP. Después lo ajusta a 2 y entonces es el segundo router el que "mata" nuestro paquete.

Si algún router no nos envía el TIME_EXCEEDED, traceroute marcará asteriscos en lugar de la IP del router.

Este campo tiene un tamaño de 8 bits, por lo que su valor máximo es 255.

El campo *Protocolo*

Especifica qué protocolo de transmisión estamos usando. Por ejemplo, el protocolo TCP tiene el número 6 (00000110) y el ICMP el 1 (00000001). Su tamaño es de 8 bits.

El campo *Header Checksum*

Un *checksum* de 16 bits de la cabecera para controlar errores. Hay que notar que cada "salto" que hace el paquete a través de un router hay que volver a calcular este número, ya que el Tiempo de Vida (TTL) cambia, como hemos visto antes.

Los campos *Source Address* y *Destination Address*

De 32 bits cada uno, estos campos especifican las **direcciones IP** del emisor y del receptor.

A la técnica de falsear la IP del emisor (es decir, enviar un paquete desde A con una Source Address diferente) se le llama *IP Spoofing*.

Después analizaremos con detalle las direcciones IP.

El campo *IP Options*

Este campo es de longitud variable. Si se especifican bytes de "IP Options" y si está activo el el primer bit, significa que la opción no será copiada en cada fragmento si el datagrama está dividido en fragmentos.

Las direcciones IP

La **dirección IP** es la identificación única que el protocolo IP hace de una máquina y de la red a la que pertenece. Es un número de 32 bits, representado por 4 bloques de 8 bits separados por puntos (por ejemplo, 01000011.11101011.00010001.10011111) .

Según el tamaño de la red hay tres clases de IPs diferentes:

Clase	Bits iniciales	ID de red	ID de máquina	Redes	Máquinas	Rango
A	0000-0111	1 byte	3 bytes	126	16.387.064	0-126
B	1000-1011	2 bytes	2 bytes	16.256	64.516	128-191
C	1100-1101	3 bytes	1 byte	2.064.512	254	122-223

Observando la anterior tabla vemos que hay números que no aparecen. Por ejemplo, los números 0 y 255 tienen un significado especial. El 0 lo usan las máquinas que no conocen su dirección. El 255 se emplea para difundir mensajes por toda la red (*broadcast*).

El 127 se emplea generalmente como bucle local. Por ejemplo, la dirección de bucle local de cada ordenador es 127.0.0.1, y la de la red es 127.0.0.0.

Las direcciones con primer bit mayor de 223 pertenecen a las clases D y E, reservadas para otros usos.

Subredes

Es posible hacer divisiones internas en una organización de forma que, por ejemplo, lo que externamente es una red de clase B (es decir, los dos primeros bytes identifican la red) internamente se divida en subredes de clase C utilizando el tercer byte para identificar cada una de ellas.

Toda la red de la organización se trataría como una red de clase B para cualquiera que accediese externamente. Por lo que es necesario un software específico en la organización para efectuar esta división interna.

Máscara de red

Una **máscara de red** o **máscara de subred** es usada por las máquinas de una red para determinar cuáles máquinas forman parte de su misma red y cuáles no. Es una dirección IP cuyos bits activos (1) son los empleados para identificar la red y cuyos bits no activos (0) forman parte de la identificación de cada ordenador.

Tomemos por ejemplo una red de clase C: `11001011.00110111.00011010.HHHHHHHH`, donde H identifica a cada ordenador. La máscara de red a usar sería `11111111.11111111.11111111.00000000`, es decir, 255.255.255.0.

---

[1]: `hping` es una completa utilería de seguridad y a mayores una buena herramienta práctica para aprender TCP/IP. www.hping.org.

[2]: http://cr.yp.to/syncookies.html

[3]: http://www.tuxedo-es.org/blog/archives/9-OpenBSD-Networking-randomization-on-the-Linux-kernel.html

---