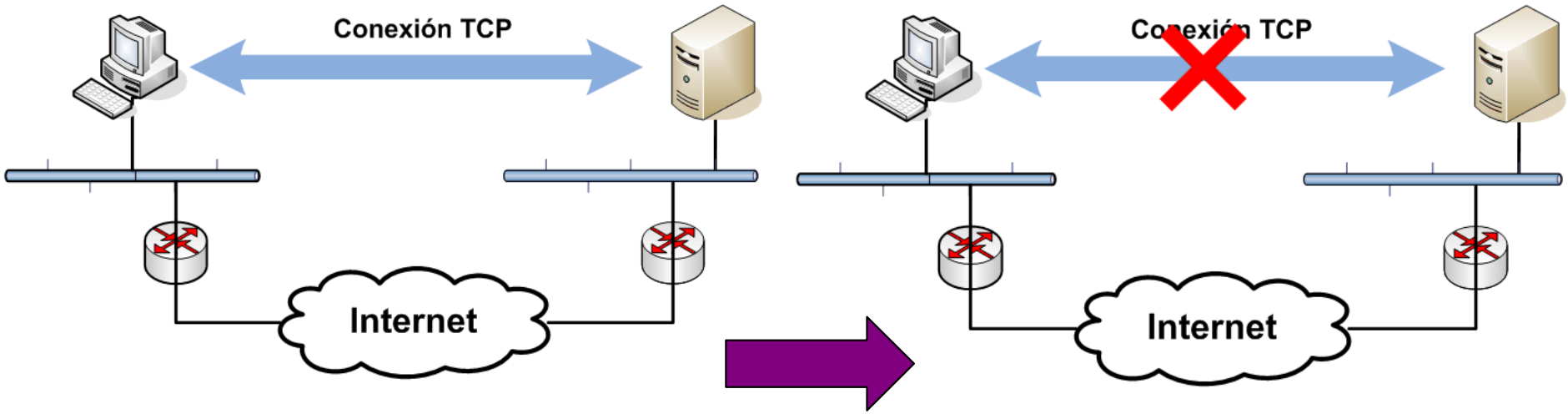
Ataques de reseteo de conexión contra TCP

Fernando Gont
Facultad Regional Haedo
Universidad Tecnológica Nacional

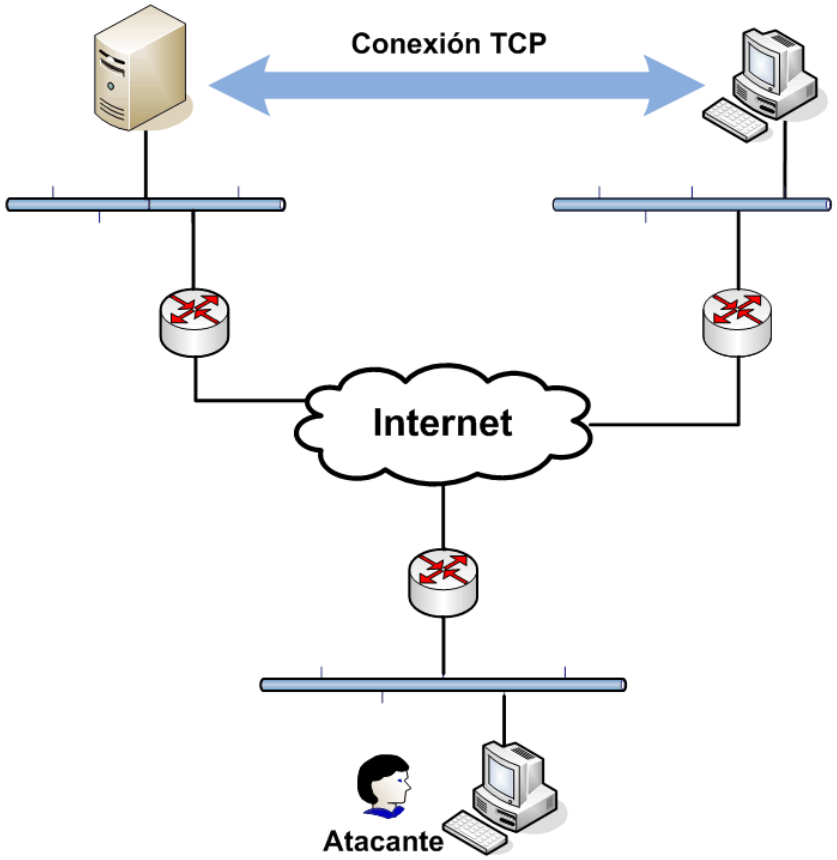
Primeras Jornadas de Divulgación Electrónica
Haedo, 26 de Octubre de 2006

Ataques de reseteo de conexión



- Un ataque de reseteo de conexión consiste en provacar que una conexión establecida entre dos sistemas sea ilegítimamente abortada.

Ataques ciegos de reseteo de conexión



- En un ataque “ciego”, el atacante no puede ver la información transmitida por ninguno de los dos sistemas que forman parte de la conexión a ser atacada.Todo valor que se necesite pare realizar el ataque deberá ser previamente conocido o adivinado

Por qué realizar ataques de reseteo de conexión?

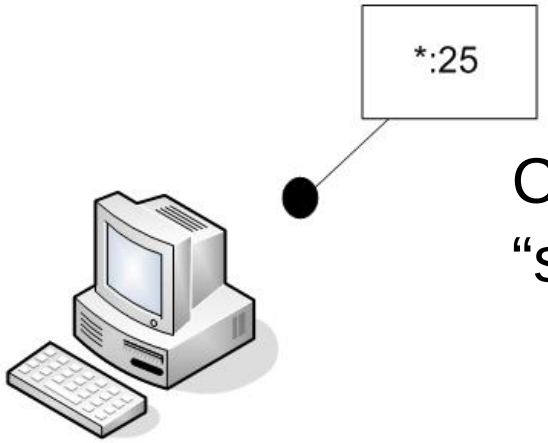
- En principio, un ataque de reseteo de conexión aborta una conexión previamente establecida, provocando Denegación de Servicio (DoS).
- Esto puede provocar que se aborte una transferencia de archivos, una sesión de ssh (consola remota), etc.
- En el caso particular de routers BGP, el reseteo de conexión provoca la eliminación (flusheo) de las rutas de la tabla de ruteo, pudiendo lograrse así el aislamiento de redes enteras. Esto lo convierte en el principal objetivo de este tipo de ataques.

Repaso de algunos conceptos básicos

Multiplexación y demultiplexación

- Para poder permitir que varias instancias de comunicación utilicen los servicios de TCP al mismo tiempo, TCP posee “puertos”, encargados de posibilitar la demultiplexación de segmentos.
- Toda aplicación que desee poder recibir conexiones, deberá quedarse escuchando en algún puerto particular.

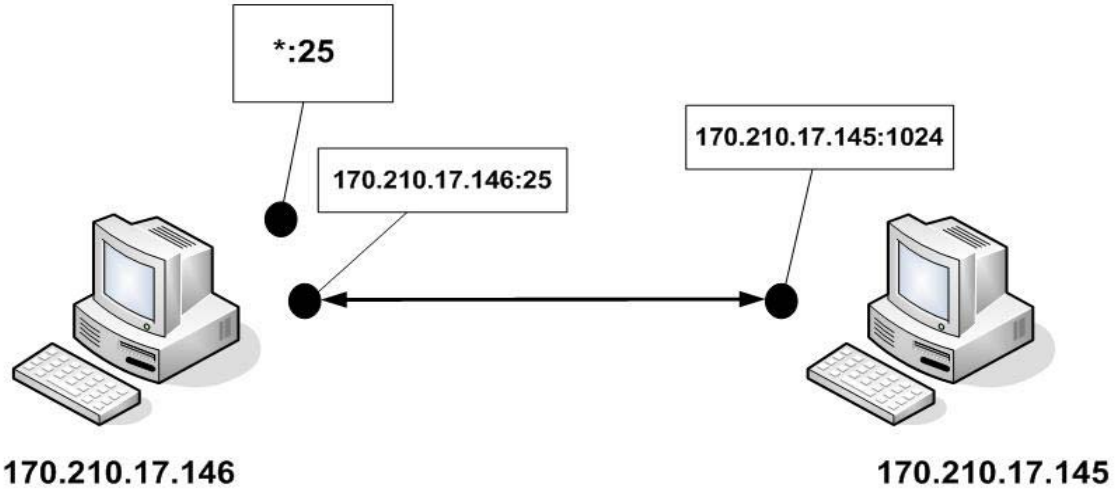
En el siguiente ejemplo, el sistema en cuestión se encuentra escuchando en el puerto 25:



Cada punto de conexion se denomina “socket”

Multiplexación y demultiplexación *(cont.)*

Si un sistema deseara conectarse al servidor de nuestro ejemplo, debería utilizar como “destiantion port” el número de puerto en el cual el servidor se encuentra escuchando, y utilizar como “source port” algún número de puerto que no estuviera utilizado actualmente.



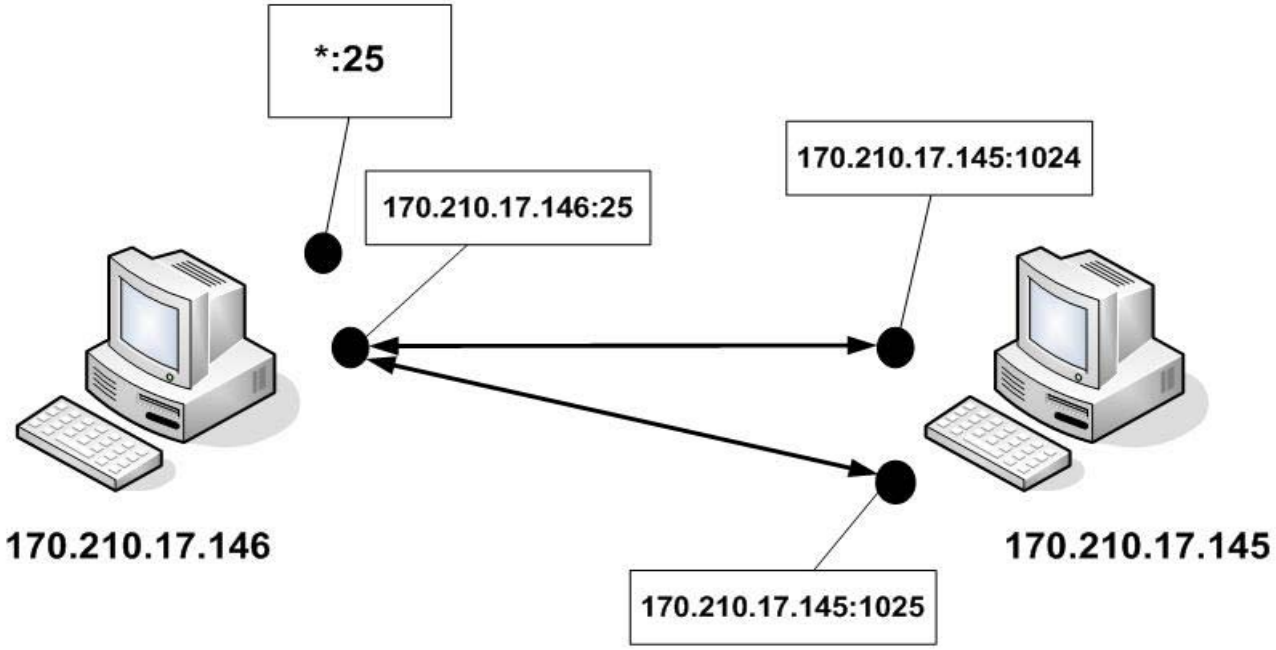
La conexión establecida estará unívocamente identificada en cada sistema como

{IP origen, puerto origen, IP destino, puerto destino}

Multiplexación y demultiplexación

(continuado)

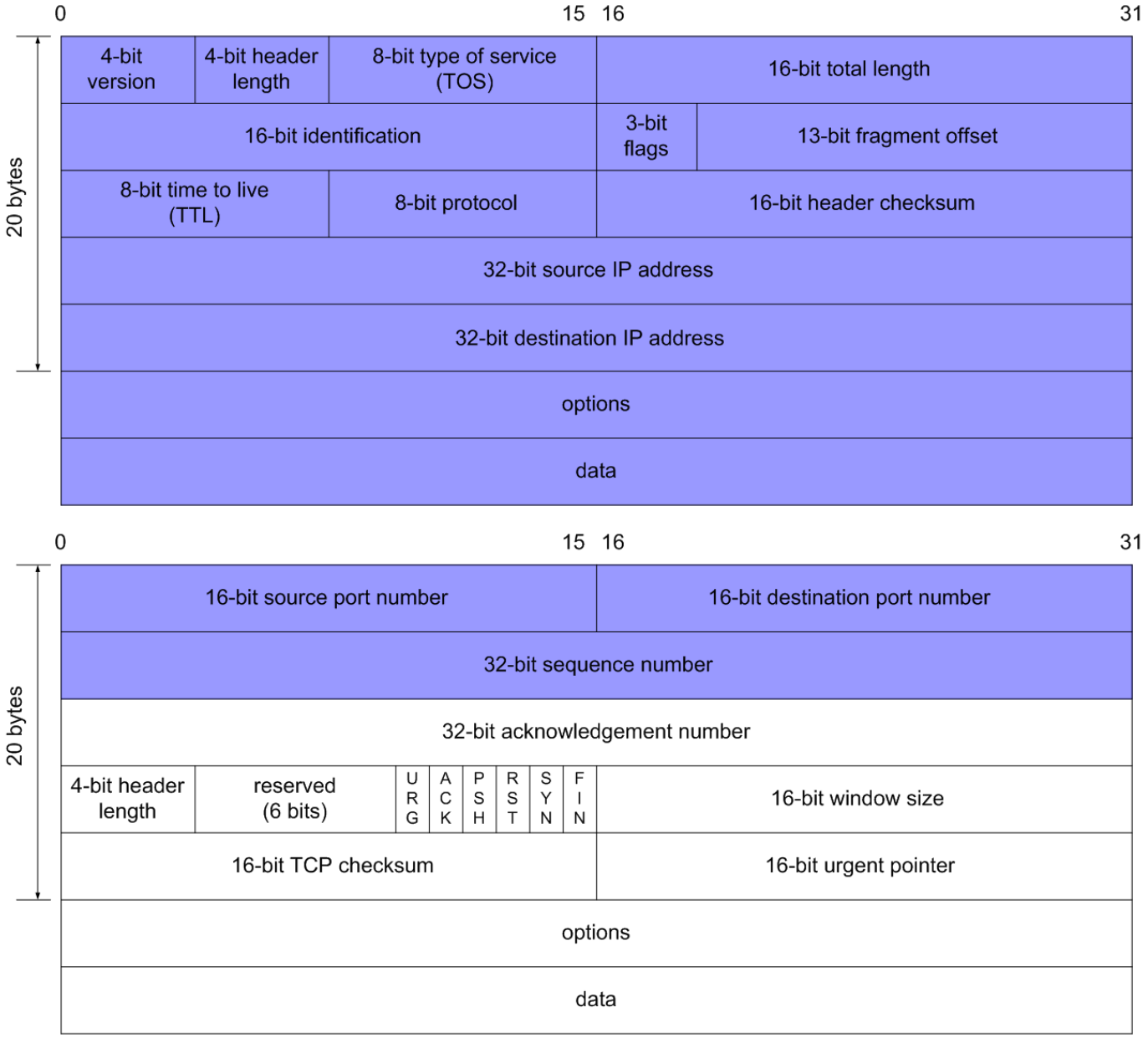
Imaginemos ahora que otra aplicación perteneciente al mismo sistema deseara establecer otra conexión al puerto 25. El cliente utilizaría algún otro número de puerto que no estuviera utilizando en ese momento. Y, de ese modo, el escenario podría ser el siguiente:



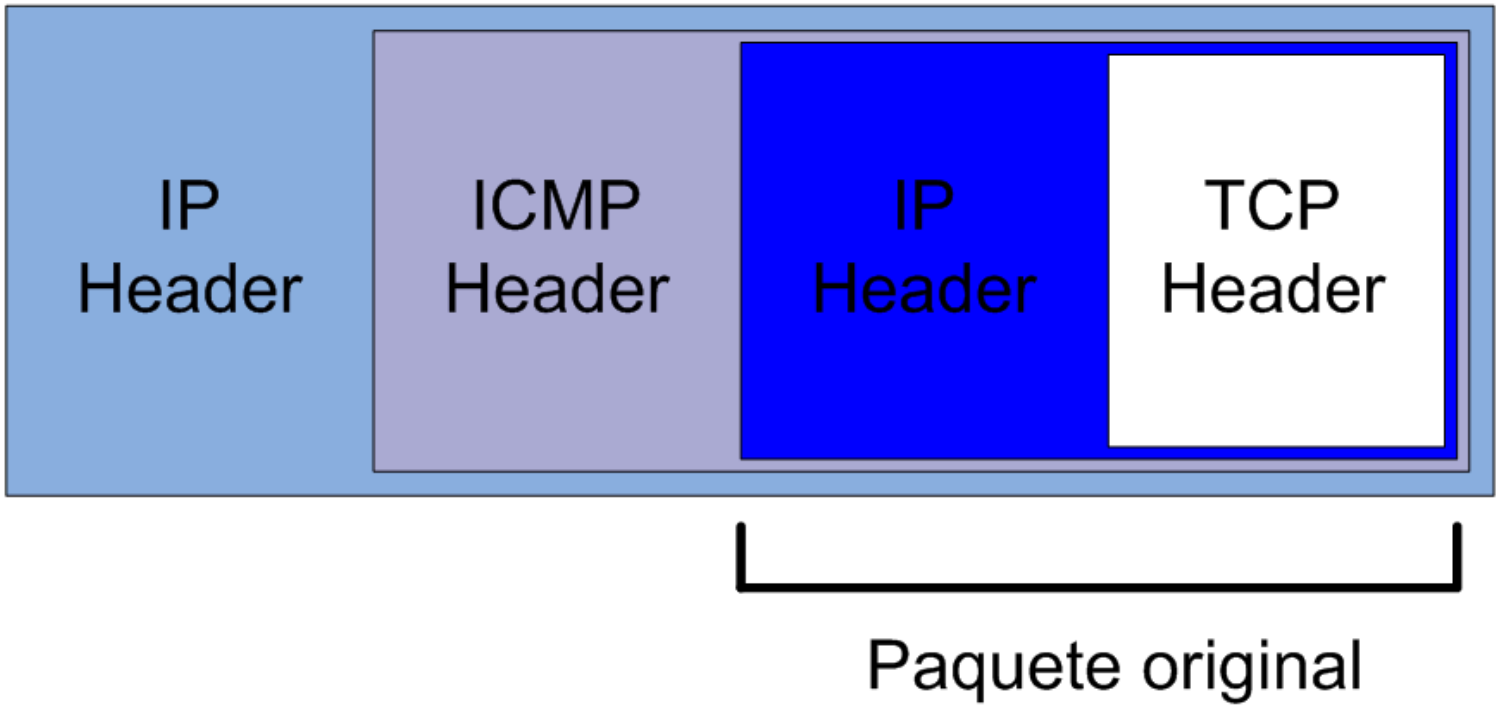
Demultiplexación de mensajes ICMP

- Cuando un sistema recibe un mensaje de error ICMP, se desea notificar la condición de error en cuestión a la instancia de comunicación que la produjo, con el fin de que el correspondiente protocolo de transporte realice su operación de “recobro de fallos”.
- Para poder demultiplexar el mensaje de error ICMP a la instancia del protocolo de transporte que la produjo, se incluirá en el mensaje de error ICMP una porción del paquete que causó el error, bajo la suposición que dicha porción del paquete original proveerá toda la información necesaria para demultiplexar el mensaje de error.

Información incluída en el mensaje ICMP



Estructura del paquete resultante



El paquete ICMP contiene en su “payload” parte del paquete original que causó el error. Dicho paquete ICMP se encapsula en un paquete IP, para ser enviado hacia el sistema que debe recibir el mensaje de error.

Validación de mensajes ICMP

Las especifiaciones de la IETF no recomiendan ningún tipo de chequeo en los mensajes de error ICMP recibidos.

Esto significa que siempre que el mensaje de error ICMP contenga los valores {source IP, source TCP port, destination IP, destination TCP port} correctos, será pasado a la correspondiente instancia de TCP, y será procesado, causando las acciones recomendadas por las especificaciones

Información necesaria para atacar

Para lograr que un mensaje de error ICMP sea pasado a una instancia de TCP dada, el atacante tendría que “adivinar”:

- Dirección IP origen (Source IP)
- Dirección IP destino (Destination IP)
- Puerto TCP origen (Source TCP port)
- Puerto TCP destino (Destination TCP port)

En principio, esto haría suponer que la información necesaria para poder atacar es demasiada como para que los ataques sean realizables en la práctica.

No hay tanto que adivinar....

- La dirección IP del servidor usualmente será conocida
- La dirección IP del cliente podría ser conocida
- El puerto TCP del servidor usualmente será conocido
- El puerto TCP del cliente usualmente no será conocido, pero podrá ser adivinado

Asumiendo que el atacante conoce las partes involucradas, y el servicio utilizado por las mismas,

Cuanto mucho, el atacante deberá enviar 65536 paquetes para realizar cualquier tipo de ataque ICMP contra TCP

Rango de puertos efímeros

La mayoría de los sistemas eligen sus “puertos efímeros” de un subespacio de todo el espacio de puertos disponible

Sistema operativo	Puertos efímeros
Microsoft Windows	1024 - 4999
Linux kernel 2.6	1024 - 4999
Solaris	32768 - 65535
AIX	32768 - 65535
FreeBSD	1024 - 5000
NetBSD	49152 - 65535
OpenBSD	1024 - 65535

Número práctico de paquetes requeridos para realizar un ataque ICMP contra TCP

En la mayoría de los casos, el atacante precisará enviar **a lo sumo** 4K paquetes para realizar cualquiera de los ataques ICMP contra TCP

Con un enlace de 128 kbps, esto llevaría nada más que **unos pocos segundos**!

Blind connection-reset attack

- Reseteando ciegamente una conexión TCP arbitraria

Generación de mensajes ICMP

- Cuando un sistema detecta una condición de error en la red, usualmente emitirá un mensaje de error ICMP, para notificar sobre la condición de error al sistema remitente del paquete que la generó.
- ICMP define una serie de mensajes de error, para notificar diversas condiciones de error en la red.
- Algunos de estos mensajes pueden ser generados por sistemas intermediarios (routers), mientras que otros pueden ser generados por sistemas finales (hosts).

Clasificación de mensajes de error ICMP

Las especificaciones de la IETF hacen una clasificación grosera de los mensajes de error ICMP en aquellos que se suponen indicar “**errores leves**” (soft errors), y aquellos que suponen indicar “**errores groseros**” (hard errors)

- Los errores leves son aquellos que se supone que se solucionarán en un corto plazo.
- Los errores groseros son aquellos que se supone que no desaparecerán en el corto plazo.

Clasificación de mensajes de error ICMP

Code	Descripción	Clasificación
0	Net unreachable	leve
1	Host unreachable	leve
2	Protocol Unreachable	grosero
3	Port unreachable	grosero
4	Frag. needed and DF set	grosero
5	Source route failed	leve

Reacción de TCP a los mensajes ICMP

Cuando TCP es notificado de una condición de error, el mismo aplicará su política de recobro de fallos:

- Si el error notificado es un “error leve” (soft error), TCP recordará el error, y continuará retransmitiendo la información hasta recibir un acuse de recibo, o hasta que decida abortar la conexión (por haber reintentado muchas veces).
- Si el error notificado es un “error grave” (hard error), TCP abortará la correspondiente conexión inmediatamente.

Blind connection-reset attack

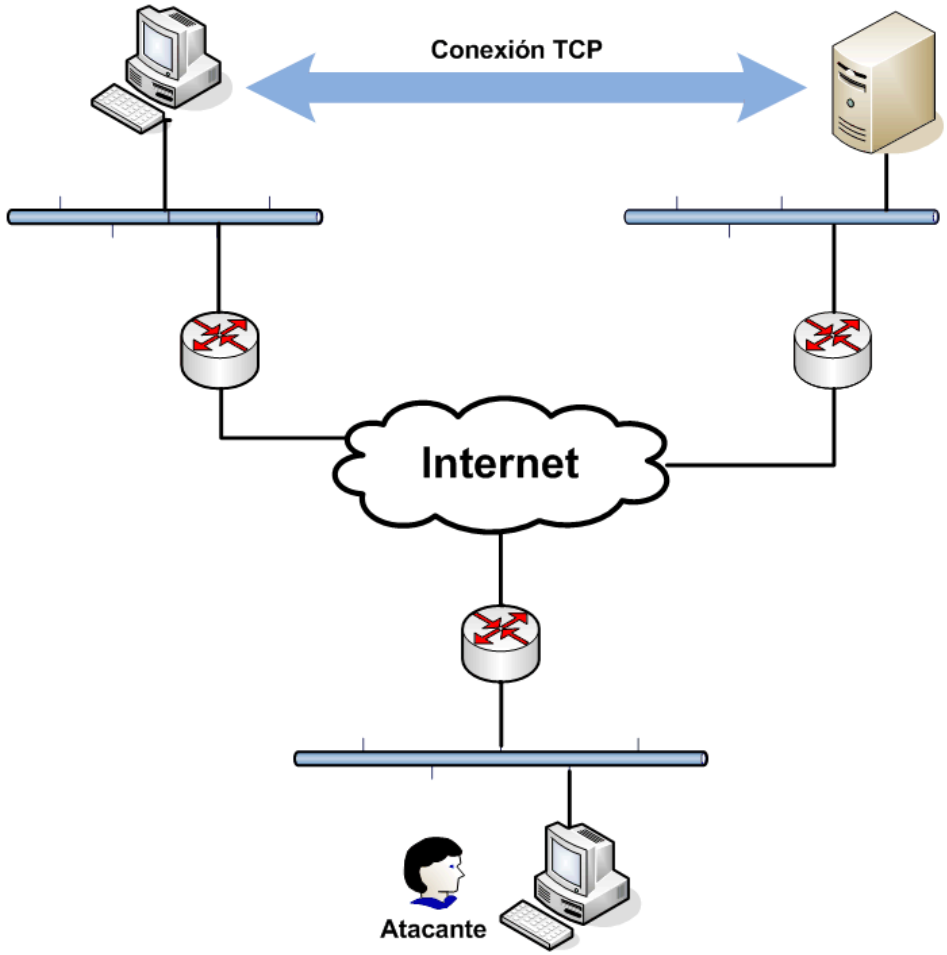
Un atacante podría falsificar un mensaje de error ICMP que indique un “error grosero” (hard error).

Como resultado, tal como indica el RFC 1122,

“TCP SHOULD abort the connection”

(*“TCP DEBERÍA abortar la conexión”*)

Impacto del ataque



Un atacante podría resetear virtualmente cualquier conexión TCP, incluso estando fuera del camino que siguen los paquetes correspondientes a la conexión

Aplicaciones afectadas

Las aplicaciones mas afectadas serán aquellas que requieren conexiones de larga vida para un correcto funcionamiento. Entre ellas podríamos encontrar:

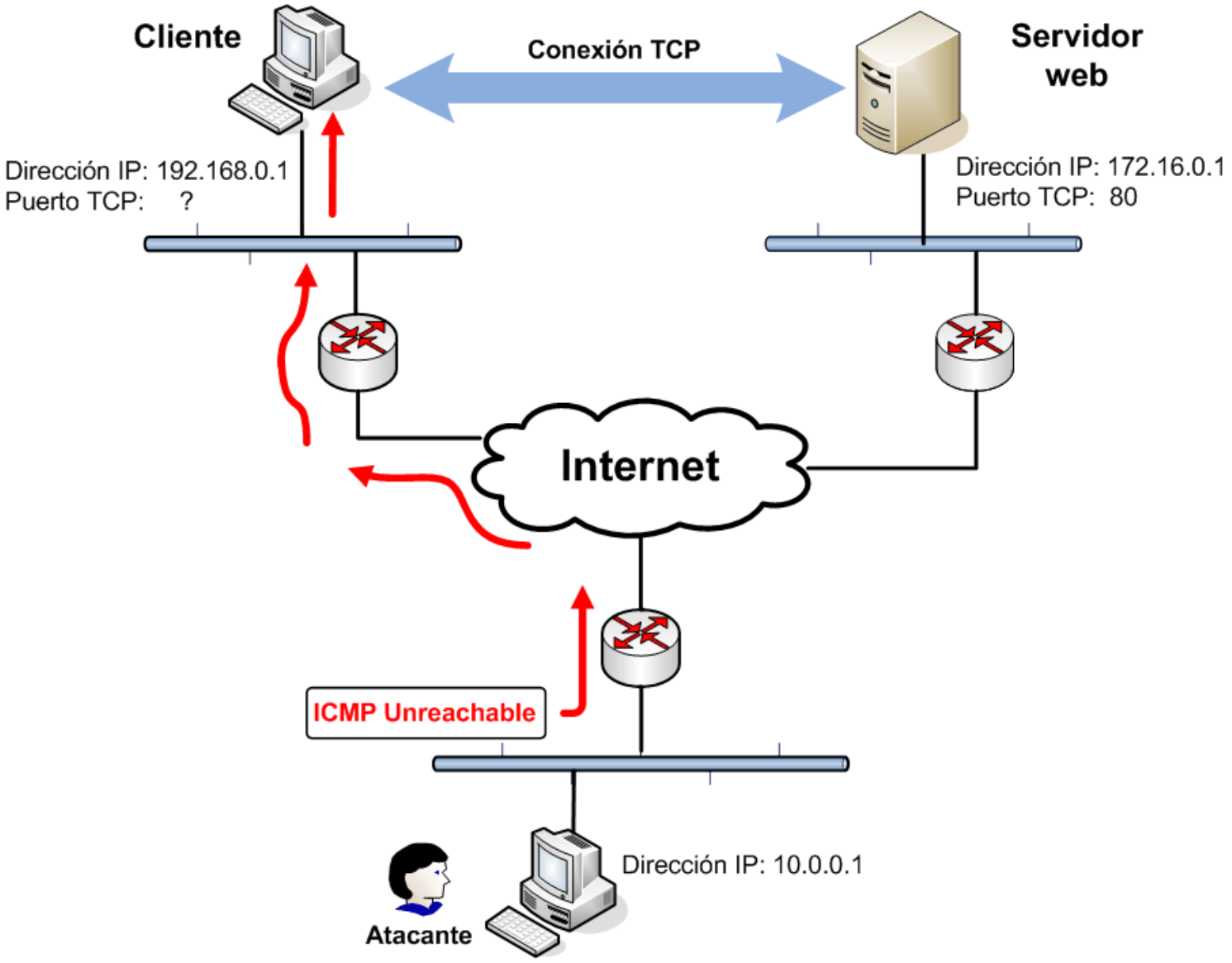
- BGP
- Sistemas de backup vía-red
- VoIP (si se utiliza TCP para la señalización)

Si el objetivo del ataque es BGP, este ataque podría provocar un estado de negación de servicio a redes enteras

Realizando el ataque

Bang bang, he shot me down
Bang bang, I hit the ground
Bang bang, that awful sound
Bang bang, my baby shot me down
- "Bang Bang", Nancy Sinatra

Escenario del ataque



Tcpdump output

22:20:56 172.16.0.1.80 > 192.168.0.1.3270: . 58849:60269(1420) ack 203 win 17040 (DF) (ttl 63, id 36261)
22:20:57 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 51749 win 9940 (DF) (ttl 118, id 23142)
22:20:57 172.16.0.1.80 > 192.168.0.1.3270: . 60269:61689(1420) ack 203 win 17040 (DF) (ttl 63, id 63275)
22:20:57 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 53169 win 9940 (DF) (ttl 118, id 23143)
22:20:57 172.16.0.1.80 > 192.168.0.1.3270: . 61689:63109(1420) ack 203 win 17040 (DF) (ttl 63, id 36878)
22:20:58 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 54589 win 9940 (DF) (ttl 118, id 23144)
22:20:58 172.16.0.1.80 > 192.168.0.1.3270: . 63109:64529(1420) ack 203 win 17040 (DF) (ttl 63, id 55051)
22:20:58 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 56009 win 9940 (DF) (ttl 118, id 23146)
22:20:58 172.16.0.1.80 > 192.168.0.1.3270: . 64529:65949(1420) ack 203 win 17040 (DF) (ttl 63, id 51041)
22:20:58 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 57429 win 9940 (DF) (ttl 118, id 23147)
22:20:58 172.16.0.1.80 > 192.168.0.1.3270: . 65949:67369(1420) ack 203 win 17040 (DF) (ttl 63, id 59428)
22:20:58 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 58849 win 9940 (DF) (ttl 118, id 23148)
22:20:58 172.16.0.1.80 > 192.168.0.1.3270: . 67369:68789(1420) ack 203 win 17040 (DF) (ttl 63, id 56440)
22:20:59 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 60269 win 9940 (DF) (ttl 118, id 23152)
22:20:59 172.16.0.1.80 > 192.168.0.1.3270: . 68789:70209(1420) ack 203 win 17040 (DF) (ttl 63, id 53543)
22:20:59 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 61689 win 9940 (DF) (ttl 118, id 23153)
22:20:59 172.16.0.1.80 > 192.168.0.1.3270: . 70209:71629(1420) ack 203 win 17040 (DF) (ttl 63, id 56257)
22:20:59 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 63109 win 9940 (DF) (ttl 118, id 23154)
22:20:59 172.16.0.1.80 > 192.168.0.1.3270: . 71629:73049(1420) ack 203 win 17040 (DF) (ttl 63, id 43027)
22:20:59 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 64529 win 9940 (DF) (ttl 118, id 23156)
22:20:59 172.16.0.1.80 > 192.168.0.1.3270: . 73049:74469(1420) ack 203 win 17040 (DF) (ttl 63, id 34869)
22:21:00 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 65949 win 9940 (DF) (ttl 118, id 23158)
22:21:00 172.16.0.1.80 > 192.168.0.1.3270: . 74469:75889(1420) ack 203 win 17040 (DF) (ttl 63, id 42831)
22:21:00 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 67369 win 9940 (DF) (ttl 118, id 23159)
22:21:00 172.16.0.1.80 > 192.168.0.1.3270: . 75889:77309(1420) ack 203 win 17040 (DF) (ttl 63, id 38361)
22:21:00 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 68789 win 9940 (DF) (ttl 118, id 23162)
22:21:00 172.16.0.1.80 > 192.168.0.1.3270: . 77309:78729(1420) ack 203 win 17040 (DF) (ttl 63, id 47968)

Tcpdump output (continuado)

22:21:00 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 70209 win 9940 (DF) (ttl 118, id 23164)
22:21:00 172.16.0.1.80 > 192.168.0.1.3270: . 78729:80149(1420) ack 203 win 17040 (DF) (ttl 63, id 56881)
22:21:01 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 71629 win 9940 (DF) (ttl 118, id 23165)
22:21:01 172.16.0.1.80 > 192.168.0.1.3270: . 80149:81569(1420) ack 203 win 17040 (DF) (ttl 63, id 50563)
22:21:01 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 74469 win 9940 (DF) (ttl 118, id 23167)
22:21:01 172.16.0.1.80 > 192.168.0.1.3270: . 81569:82989(1420) ack 203 win 17040 (DF) (ttl 63, id 39445)
22:21:01 172.16.0.1.80 > 192.168.0.1.3270: . 82989:84409(1420) ack 203 win 17040 (DF) (ttl 63, id 61324)
22:21:01 172.16.0.1 > 192.168.0.1: icmp: 172.16.0.1 protocol 6 unreachable for 192.168.0.1.3270 > 172.16.0.1.80: [|tcp] (ttl 158, id 61654) (ttl 214, id 31456)
22:21:02 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 75889 win 9940 (DF) (ttl 118, id 23169)
22:21:02 172.16.0.1.80 > 192.168.0.1.3270: . 84409:85829(1420) ack 203 win 17040 (DF) (ttl 63, id 46016)
22:21:02 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 78729 win 9940 (DF) (ttl 118, id 23171)
22:21:02 172.16.0.1.80 > 192.168.0.1.3270: . 85829:87249(1420) ack 203 win 17040 (DF) (ttl 63, id 64644)
22:21:02 172.16.0.1.80 > 192.168.0.1.3270: . 87249:88669(1420) ack 203 win 17040 (DF) (ttl 63, id 39376)
22:21:02 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 80149 win 9940 (DF) (ttl 118, id 23173)
22:21:02 172.16.0.1.80 > 192.168.0.1.3270: . 88669:90089(1420) ack 203 win 17040 (DF) (ttl 63, id 58117)
22:21:02 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 81569 win 9940 (DF) (ttl 118, id 23175)
22:21:02 172.16.0.1.80 > 192.168.0.1.3270: . 90089:91509(1420) ack 203 win 17040 (DF) (ttl 63, id 62887)
22:21:03 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 82989 win 9940 (DF) (ttl 118, id 23176)
22:21:03 172.16.0.1.80 > 192.168.0.1.3270: . 91509:92929(1420) ack 203 win 17040 (DF) (ttl 63, id 54586)
22:21:03 192.168.0.1.3270 > 172.16.0.1.80: . [tcp sum ok] 203:203(0) ack 84409 win 9940 (DF) (ttl 118, id 23177)
22:21:03 172.16.0.1.80 > 192.168.0.1.3270: . 92929:94349(1420) ack 203 win 17040 (DF) (ttl 63, id 56692)
22:21:03 192.168.0.1.3270 > 172.16.0.1.80: R [tcp sum ok] 4174694923:4174694923(0) win 0 (ttl 118, id 23180)

Solución: Son los “errores graves” realmente graves?

Si se reporta un supuesto “error grave” a una conexión que **ya ha sido establecida**, entonces el error no puede ser “grave”. Si lo fuera, nunca podría haber establecido la conexión!

Para conexiones TCP en cualquiera de los estados sincronizados (es decir, desde que se ha establecido la conexión, en adelante), todos los errores ICMP deberían ser considerados “leves”

Preguntas y respuestas

Fernando Gont

fernando@gont.com.ar

Más información en:

http://www.gont.com.ar