Técnicas de Detección Avanzada de interconectividad

Detección de equipos conectados a la red

Víctor Escudero Rubio

Técnicas de Detección Avanzada de interconectividad: Detección de equipos conectados a la red

por Víctor Escudero Rubio

Publicado v1.0, 23 de Octubre del 2001

Este documento pretende dar a conocer entre la comunidad las técnicas más avanzadas de detección de equipos fuertemente filtrados o protegidos por cortafuegos. Se presupone que el lector deberá tener buenos conocimientos en cuanto a los protocolos subyacentes de Internet (IP, ICMP, UDP, TCP) y las tecnicas de detección simple de equipos conectados a la red.

Tabla de contenidos

Capítulo 1. Introducción

A menudo nuestros equipos mas preciados sufren ataques del exterior de la más diversa índole y los costes asociados pueden llegar a ser verdaderamente importantes y es entonces en ese momento en el que las empresas afectadas empiezan a tomar consciencia del grave riesgo que corren por el simple hecho de formar parte de Internet.

Muchas instituciones, empresas y universidades han recurrido a la instalación masiva de cortafuegos que salvaguarden su información más preciada y su propia integridad, pero a menudo esta labor no ha sido realizada con toda la laboriosidad necesaria y ha dado lugar a una falsa idea de seguridad que es si cabe más peligrosa aún que una exposición consciente a internet.

Para evitar muchos de los ataques a nuestros cortafuegos tan sólo deberíamos ser conscientes de los peligros a los que están expuestos y tratar de minimizar los riesgos aplicando las medidas de seguridad necesarias.

Es ampliamente reconocido por los administradores de sistemas, que la mayoría de los ataques realizados desde internet vienen precedidos por largos y en general *ruidosos* escaneos de puertos realizados mediante potentes herramientas.

Generalmente los buenos administradores no tienen gran dificultad en seguir la pista de muchos de los escaneos realizados desde el otro lado por gamberros con potentes herramientas pero con escasa capacidad técnica. Recientemente los sistemas conocidos como NIDS (Network Intrusion Detection System) han venido ha paliar la necesidad de los administradores de sistemas de detectar a su vez los escaneadores de puertos mas conocidos y algunas de sus técnicas más avanzadas. Sin embargo un administrador sigue requeriendo una buena formación para entender los diversos tipos de escaneos existentes para protegerse de ellos y poder estar alerta en caso de intrusión.

Capítulo 2. Objetivo

El autor pretende dar a conocer las técnicas de detección más avanzadas de máquinas e interconectividad empleadas por algunos atacantes de la élite hacker. No se trata de describir las técnicas de escaneos de puertos empleadas por los hackers sino más bien el paso previo de determinar el objetivo a atacar. Frecuentemente, antes de realizar un escaneo de puertos se debe hacer un análisis de la red a la que se va a atacar así como la posición de determinados elementos clave, como IP’s del firewall. Con frecuencia el conocimiento de la posición del servidor de correo o del firwall es determinante para que un ataque tenga éxito.

Capítulo 3. Obtención de éste documento

Éste Cómo puede obtenerse a través del proyecto de Documentación de Linux en Español en INSFLUG (http://www.insflug.org) o a través de las páginas oficiales de ésta documentación:

- Mi propia página personal en PLucent.com (http://www.plucent.com)
- Un sitio web denominado Cortafuegos.org (http://www.cortafuegos.org/documentacion) en el que colaboramos una serie de gente con inquietudes por la seguridad informática.

Capítulo 4. Métodos de detección de máquinas

Para poder determinar con certeza la existencia de protecciones y filtrados dentro de una red a la que se va a atacar se pueden emplear cualquiera de estos dos métodos de mapeo de equipos:

- *Forzar una respuesta del servidor ante una solicitud válida*
- *Generar una respuesta del servidor ante una solicitud inválida*

4.1. Tipos de peticiones válidas para obtener respuesta de un servidor

La inmensa mayoría de las peticiones de un cliente a un servidor son perfectamente válidas, como por ejemplo un echo-request (tipo 8 de icmp= ping) es una petición habitual de un cliente frecuentemente respondida con un echo-reply (tipo 0 de icmp=pong), pero dado que nosotros estamos tratando de delatar la presencia de un sistema de filtrado, como puede ser un *router* protegido con ACL's o un firewall o grupo de ellos damos por sentado que no nos responderán a simples pings. Los tipos de consultas que realizaremos son:

- UDP Eco
- TCP Eco
- UDP hacia puertos cerrados
- TCP ACK
- TCP SYN
- TCP SYN | ACK
- TCP FIN
- TCP NULL *FLAGS*
- TCP XMAS

- ICMP Petición de eco (Tipo 8)
- ICMP Peticion de *broadcast*
- ICMP Petición de *router* (Tipo 10)
- ICMP Petición de marca de tiempo (Tipo 13)
- ICMP Petición de información (Tipo 15)
- ICMP Petición de máscara de red (Tipo 17)

4.2. Tipos de peticiones inválidas para obtener respuesta de un servidor

La mayoría de las implementaciones de los diseñadores de sistemas operativos y software de filtrado cumplen las RFC’s si no en su totalidad si más del noventa y nueve por ciento, pero hay algunas recomendaciones que no siguen y lo que es aún peor, hay comportamientos no previstos en las RFC’s que cada fabricante ha decidido seguir con cierta libertad, por lo que no se comportarán igual ante una misma solicitud. Este hecho es grandemente utilizado por las herramientas de detección de sistemas operativos y versiones enfrentando una serie predeterminada de consultas y analizando los datos devueltos con respecto a unas tablas prefabricadas de firmas de pilas conocidas (TCP/IP fingerprinting).

El uso que lo daremos aquí no será determinar el tipo de máquina que habrá al otro lado pues tan sólo querremos saber que al menos ahi algo al otro lado que está filtrando determinado tipo de tráfico o no lo hay.

Los métodos que emplearemos serán:

- Provocar la respuesta de timeout ante falta de fragmentos para recomponer un paquete
- Enviar paquetes con cabeceras IP de tamaños no válidos
- Enviar paquetes con valores en ciertos campos IP.

Capítulo 5. Obtención de respuesta del servidor ante una petición válida

Un buen firewall debería estar correctamente protegido y tener unas reglas lo suficientemente auditadas como para evitar los principales escaneos, los cuales pueden resumirse en los siguientes:

- Acceso al puerto de eco
- Método UDP
- Escaneo mediante *flags* de TCP
- Peticiones ICMP

La idea principal del firewall deberá ser pasar lo más desapercibidamente posible ante los posibles atacantes registrando al mismo tiempo los accesos no permitidos de los mismos. La idea de permanecer invisibles es muy importante, de tal forma que si alguien intenta contactar con el cortafuegos los paquetes enviados parezcan desvanecerse en el éter como si no hubiese ninguna máquina con dicha IP. A menudo un ping será tirado sin dar muestras de presencia de *algo* al otro lado. En cambio ciertos escaneos avanzados con *flags* tcp podrían ser válidos y por lo tanto no se pueden tirar sin más obligandonos a responder y por lo tanto denotar nuestra presencia. Para la comprensión de nuestro objetivo siempre deberemos tener en cuenta que de cara a un posible atacante la primera medida de defensa es convertirnos en lo más invisible posible ante él para frustrar un posible ataque que nos pueda ocasionar mayores trastornos en etapas posteriores.

5.1. Escaneos al puerto de eco

El puerto de eco (echo) era antiguamente usado para comprobaciones de respuesta de red entre dos máquinas, de tal forma que todo lo que se enviaba a una máquina era devuelto íntegramente exactamente igual a como se enviaba. Este uso ha ido quedando fuera de moda y en varias distribuciones Linux/Unix viene cerrado por defecto, pero es conocida su presencia en otros sistemas como el Solaris de Sun Microsystems. Dado que la mayoría de los administradores conscientes de los peligros que acechan desde

internet cierran dichos puertos no debemos esperar que estén abiertos sino que pueden ser cebos realizados con herramientas como portsentry o tcp_wrappers.

Hay dos tipos de conexiones al puerto de eco según el protocol usado:

- Conexiones tcp
- Conexiones udp

5.1.1. Conexiones tcp al puerto de eco

La conexión tcp al puerto de eco es muy interesante para un posible atacante porque requiere una negociación completa en tres pasos (*Three-way-handshaking*). En realidad ni siquiera se necesita enviar y recibir datos sino que lo importante es que la simple negociación delata la presencia de un posible equipo que se podría convertir en objetivo de un ataque minucioso.

Veamos un ejemplo de conexión mediante tcp al puerto de eco usando un simple telnet:

```
delorian:~# telnet xxx.xxx.xxx.xxx echo
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
Todo lo que se envia a la maquina de destino es devuelto
Todo lo que se envia a la maquina de destino es devuelto
^]
telnet> close
Connection closed.
```

5.1.2. Conexiones udp al puerto de eco

Una simple respuesta del puerto de eco nos es más que suficiente para delatar la presencia de un equipo, por lo que ni siquera necesitamos de una conexión completa tcp.

Veamos un ejemplo de conexión al puerto de eco mediante protocolo udp usando netcat:

```
vicescu@delorian:~$ nc -v -u  xxx.xxx.xxx.xxx echo    # v=verbose u=udp ec
```

```
remote [xxx.xxx.xxx.xxx] 7 (echo) open
Prueba de envio y recepcion
Prueba de envio y recepcion

vicescu@delorian:~$
```

5.2. Método UDP hacia puertos cerrados

Es conocido que los puertos UDP cerrados deben responder (según RFC) ante una petición de conexión, por lo que puede ser explotable por un posible hacker.

Una petición udp de un cliente ante un puerto udp cerrado (que no esté escuchando) en el otro lado genera un error ICMP conocido como PORT_UNREACH.

La forma de determinar un puerto cerrado de un posible equipo objetivo es sencilla: elegimos uno o varios puertos altos (de 1024 a 65536).

Es posible que los paquetes udp no estén llegando a su destino o no nos esté llegando la respuesta icmp por diversos motivos:

- El puerto udp realmente está abierto, deberemos probar con algún otro.
- El puerto udp está siendo bloqueado o filtrado por un *router*, firewall o el mismo equipo de destino.
- Quizá el puerto este cerrado y generando un error icmp PORT_UNREACH, pero está siendo filtrado en su regreso.
- El protocolo udp no proporciona fiabilidad, por lo que será necesario retransmitirlo para tener ciertas garantías de que llegó a su destino.

Veamos una transmisión udp contra un puerto cerrado (normalmente nfs está en el 2049):

```
# Opciones utilizadas de hping:  -V=verbose, -c 1= 1 sólo paquete, -
2=Protocolo udp, -p=puerto
delorian:~# hping2 -V -V  -c 1 -2 -p 2049 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
using eth0, addr: xxx.xxx.xxx.xxx, MTU: 1500
```

```
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): udp mode set, 28 hea-
ders + 0 data bytes
ICMP Port Unreachable from xxx.xxx.xxx.xxx  (remote)

--- remote hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.8/0.8/0.8 ms
delorian:~#
```

Como hemos podido observar el puerto de 2049 estaba cerrado, por lo que hemos recibido una contestación ICMP.

Veamos una transmisión udp contra un puerto abierto (DNS en este caso esta escuchando peticiones)

```
delorian:~# hping2 -V -V  -c 1 -2 -p 53 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
using eth0, addr: xxx.xxx.xxx.xxx, MTU: 1500
HPING remote (eth0 xxx.xxx.xxx.xxx): udp mode set, 28 headers + 0 da-
ta bytes

--- remote hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

Como hemos podido observar el puerto 53 de DNS estaba abierto (o bloqueado), por lo que no hemos recibido ninguna contestación.

5.3. Escaneo mediante *flags* de TCP

Quizá el rastreo mediante la activación de determinadas banderas o *flags* especiales de TCP es el método más efectivo en términos de determinación de conectividad de un equipo. Esto se debe principalmente a que estas opciones son muy usadas en el tráfico TCP en el día a día y puede resultar difícil distinguir paquetes tratando de captar información de aquellos que son paquetes legales.

Los tipos principales de detección de interconexión mediante *flags* de tcp son:

- Paquete con el *flag* SYN activado

- Paquete con el *flag* ACK activado
- Paquete con los *flags* SYN y ACK activados
- Paquete con el *flag* FIN activado
- Paquete sin ningún *flag* activo (*Null flags scan*)
- Paquete con todos los *flags* activos (*XMAS scan*)

5.3.1. Aproximación mediante el *flag* SYN activo

Un paquete TCP con el bit SYN activo siempre debe ser respondido por una posible máquina que esté al otro lado de la conexión. La respuesta puede ser de dos tipos:

1. Respuesta SYN | ACK como petición a un puerto abierto. (*listening*)
2. Respuesta RST | ACK como petición a un puerto cerrado.

También es posible que no se responda ante una petición SYN lo que indicará que o bien la conexión a dicho puerto está filtrada o bien el equipo no está en línea.

Veamos como ejemplo un intento de conexión tcp a un puerto activo (ssh por defecto es el 22):

```
delorian:~# hping2 -c 1 -S -p 22 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): S set, 40 headers + 0 da-
ta bytes
44 bytes from xxx.xxx.xxx.xxx: flags=SA seq=0 ttl=64 id=0 win=32767 rtt=0.

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

Hemos recibido un paquete con los *flags* de Syn y Ack activos.

Veamos ahora otro intento de conexión solo que ahora a un puerto cerrado:

```
delorian:~# hping2 -c 1 -S -p 36345 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
```

```
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): S set, 40 headers + 0 da-
ta bytes
44 bytes from xxx.xxx.xxx.xxx: flags=RA seq=0 ttl=64 id=0 win=32767 rtt=0.

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

La respuesta que hemos obtenido (RST + ACK) nos indica que el puerto estaba cerrado.

5.3.2. Aproximación mediante el *flag* ACK activo

Un paquete TCP con el *flag* ACK activo no nos indicará el estado del puerto contra el que se dirige, pero en cambio si nos delatará la presencia de un equipo al otro lado de la conexión.

Un paquete con el bit ACK activo a un puerto cualquiera, esté abierto o cerrado deberá ser respondido con un RST. El problema que tiene la aproximación mediante este *flag* activo es que a diferencia de la aproximación mediante un SYN su uso denota claramente un escaneo de puertos. Su "ruidosidad" es bastante elevada y de hecho un cortafuegos cuyas reglas estén bien construidas deberá desechar todos los paquetes que pretendan abrir una conexión con cualquier combinación de *flags* distintas de un solo bit activando el *flag* de SYN.

En cambio muchos son los cortafuegos que impiden abrir una nueva conexión contra determinados puertos prohibiendo los SYN, pero en cambio *"olvidan"* que esos puertos sí que responderán ante un intento con ACK. Este método es ideal para realizar lo que comunmente se denomina *Firewalking* que consiste en determinar los paquetes que serán filtrados y los que no por un cortafuegos basándose en los paquetes que logran atravesarlo y los que no para deducir el conjunto de reglas que están siendo empleadas en la proteción de terceros equipos.

Veamos un ejemplo contra un puerto cerrado de un FreeBSD:

```
delorian:~# hping2 -c 1 -A -p 2244 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): A set, 40 headers + 0 da-
ta bytes
40 bytes from xxx.xxx.xxx.xxx: flags=R seq=0 ttl=255 id=0 win=0 rtt=0.2 ms

--- xxx.xxx.xxx.xxx hping statistic ---
```

```
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
```

La misma consulta no hubiera sido respondida por un Win2K o un NT por ejemplo.

5.3.3. Aproximación mediante los *flags* SYN y ACK activos

Este método tiene la desventaja de no ser efectivo en todos los casos, pues las distintas implementaciones lo tratan de forma diferente.

Por ejemplo la respuesta a un paquete TCP con los flags SYN | ACK activos por parte de equipos Linux/Windows es un simple RST, en cambio la familia de sistemas libres derivados de BSD (OpenBSD, NetBSD y FreeBSD) y el propio BSD ignoran totalmente estos paquetes y los descartan en cuanto son recibidos.

Veamos un ejemplo contra un servidor web Windows 2000 Advanced Server:

```
delorian:~# hping2 -c 1 -S -A -p 80  xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): SA set, 40 headers + 0 da-
ta bytes

--- xxx.xxx.xxx.xxx hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
```

En cambio un OpenBSD responde sin ningún problema a nuestra petición:

```
delorian:~# hping2 -c 1 -S -A -p 22 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): SA set, 40 headers + 0 da-
ta bytes
40 bytes from xxx.xxx.xxx.xxx: flags=R seq=0 ttl=255 id=0 win=0 rtt=0.2 ms

--- xxx.xxx.xxx.xxx hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
```

5.3.4. Aproximación mediante el *flag* FIN activo

Posiblemente éste sea uno de los métodos de aproximación más clandestinos.

Un paquete con el flag FIN activo deberá ser respondido de la siguiente forma:

1. RST | ACK ante un intento a un puerto cerrado.
2. Nada ante un intento en un puerto abierto.

Una falta de respuesta ante un paquete FIN podrá deberse a uno cualquiera de estos casos:

- Paquetes FIN bloqueados por cortafuegos, *routers* o ACL's.
- Tráfico de entrada a ese puerto en concreto está siendo filtrado.
- El puerto consultado resultó estar abierto (se deberá intentar en otro).
- El equipo de destino está realmente desconectado de la red o caído.

Veamos la respuesta de un Debian GNU/Linux ante una tentativa de finalizar una conexión contra un puerto que de hecho estaba cerrado:

```
delorian:~# hping2 -c 1 -F -p 1234  203.13.123.45
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): F set, 40 headers + 0 da-
ta bytes
40 bytes from xxx.xxx.xxx.xxx: flags=RA seq=0 ttl=255 id=0 win=0 rtt=0.2 m

--- xxx.xxx.xxx.xxx hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
```

5.3.5. Aproximación mediante paquetes TCP sin ningún *flag*

Este tipo de aproximación es comunmente conocido como *TCP Null flag*. Se basa en enviar un paquete TCP al que se le han eliminado todas los flags, una vez en el destino

el equipo deberá responder con un paquete con RST | ACK activados si el puerto estaba cerrado y nada si está abierto.

Generalmente la mayoría de ACL's de los *routers* intermedios y firewalls están prevenidos contra este tipo de intentos por lo que no es probable que obtengamos ninguna respuesta.

Forzaremos una respuesta de un Solaris 2.8 enviando un paquete sin ningún flag a un puerto cerrado:

```
delorian:~# hping2 -c 1 -p 1234  xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): NO FLAGS are set, 40 hea-
ders + 0 data bytes
40 bytes from xxx.xxx.xxx.xxx: flags=RA seq=0 ttl=255 id=0 win=0 rtt=0.2 m

--- 203.13.123.45 hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
```

5.3.6. Aproximación mediante la activación de todos los *flags* TCP

Inversamente a la aproximación Null *flag* en la que no se activaba ninguna de las opciones de TCP, existe el caso contrario los paquetes conocidos como XMAS *(XMAS ⟶ Christmas)* en los que todos los *flags* son activados, es decir: SYN | ACK | FIN | RST | URG | PSH.

Si bien es bastante eficaz contra las pilas TCP/IP derivadas del primigenio BSD, es decir Linux, Solaris, *BSD's, etc, los equipos Windows no responden a este tipo de paquetes.

La respuesta de los equipos unix ante un paquete XMAS es devolver un RST | ACK en los puertos cerrados y nada en los abiertos al igual que pasaba con los paquetes sin ningún *flag* activo.

Veamos un ejemplo contra un vetusto BSD sobre un puerto cerrado:

```
delorian:~# hping2 -c 1 -F -S -R -P -A -U -X -Y -p 9987 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): RSAFPUXY set, 40 hea-
ders + 0 data
```

```
s
40 bytes from xxx.xxx.xxx.xxx: flags=RA seq=0 ttl=255 id=0 win=0 rtt=0.3 m
--- xxx.xxx.xxx.xxx hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.3/0.3/0.3 ms
```

Sin embargo en un Debian con woody y kernel 2.4.0 estándar el kernel no reconoce estos paquetes y se deshace de ellos:

```
delorian:~# hping2 -c 1 -F -S -R -P -A -U -X -Y -p 6666 xxx.xxx.xxx.xxx
default routing interface selected (according to /proc)
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): RSAFPUXY set, 40 hea-
ders + 0 data
s

--- xxx.xxx.xxx.xxx hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.9/0.9/0.9 ms
```

5.3.7. Comentario a los rastreos con *flags* TCP

En general al ser las diversas variantes de Unix bastante heterogéneos entre sí pero con un objetivo común de mantener la compatibilidad con antiguas implementaciones, es común que Unix en general responda por defecto ante un gran número de paquetes malformados. Por contra la opción de Microsoft fue desde un comienzo tomar como alternativa dejar de lado algunas recomendaciones y tirar paquetes malformados de tal forma que la conectividad entre todo tipo de Windows sea plena, pero no se asegure conectividad total con antiguas implementaciones. Este hecho puede hacer que Windows aparezca como más trasparente ante diversos tipos de paquetes, pero en realidad veremos que es totalmente visible al igual que el resto con la particularidad de que responde a paquetes que los demás descartan y viceversa.

5.4. Métodos ICMP

El protocolo ICMP fue originalmente diseñado para reportar errores en los procesos de transmisión de datos como una ayuda a la capa IP.

Debido a la creciente necesidad de seguridad muchos son los que están bloqueando todos los tipos de paquetes ICMP, pero si bien algunos son realmente dañinos (*smurf broadcast*, pings de tamaño excesivo, demasiadas respuestas simuladando destinos inalcanzables, etc) otros son de gran ayuda en la comunicación como por ejemplo las marcas de tiempo *(timestamps)*.

5.4.1. Peticiones ICMP de eco (tipo 8)

Es el primer método que deberíamos de probar ante la gran mayoría de máquinas no protegidas, se trata de enviar una petición de eco llamada ping (tipo 8) y esperar una respuesta de eco llamada pong (tipo 0)

Veamos un simple ping lanzado desde un linux:

```
delorian:~# ping -c 1 xxx.xxx.xxx.xxx
PING xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx): 56 data bytes
64 bytes from xxx.xxx.xxx.xxx: icmp_seq=0 ttl=255 time=0.4 ms

--- xxx.xxx.xxx.xxx ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.4 ms
```

Desgraciadamente suele ser el primer tipo de filtro que se suele poner a la mayoría de las máquinas que deben quedar expuestas al exterior por lo que es muy poco probable que recibamos respuesta.

5.4.2. Peticiones ICMP *broadcast*

Este tipo de peticiones deben ser siempre filtradas en un entorno protegido, pues puede dar lugar a ataques de tipo *smurf* y colapsos.

Puede ser muy útil en redes Unix pues la mayoría de los Unix responden a peticiones contra la dirección de red o la de broadcast devolviendo al origen un mensaje de vuelta de eco (tipo 0) delatando la presencia de muchas máquinas junto con la estructuracion de la red en subredes.

Por contra los Windows NT 4.0 con Service Pack superior o igual a 4, los Windows 2000 y toda la familia de Windows 9x y Millenium no responden por defecto a este tipo de mensajes icmp.

Veamos un ping lanzado contra una red de tipo C en la que hay 5 máquinas actualmente levantadas, de las cuales dos son linux (.1 y .2), otro es un Solaris (.3) y los otros dos son dos Windows 98 (.4 y .5) que no responden ni a una petición a la red ni a la dirección de broadcast:

```
delorian:~# hping2 -1 xxx.xxx.xxx.0 -c 1
default routing not present
HPING xxx.xxx.xxx.0 (eth0 xxx.xxx.xxx.0): icmp mode set, 28 headers + 0 da
ta bytes
28 bytes from xxx.xxx.xxx.1: icmp_seq=0 ttl=255 id=43 rtt=0.2 ms
28 bytes from xxx.xxx.xxx.2: icmp_seq=0 ttl=255 id=310 rtt=0.3 ms
50 bytes from xxx.xxx.xxx.3: icmp_seq=0 ttl=255 id=323 rtt=0.8 ms

--- xxx.xxx.xxx.xxx hping statistic ---
1 packets tramitted, 3 packets received, -100% packet loss
round-trip min/avg/max = 0.2/0.4/0.8 ms
```

Similares resultados hubieramos obtenido si hiciésemos una petición a la dirección xxx.xxx.xxx.255 de broadcast los dos windows no responderían mientras los Unix responderían.

5.4.3. Peticiones ICMP de solicitud de *router* (tipo 10)

Cada router periódicamente envía mediante multicast una trama ICMP por cada una de sus interfaces advirtiendo de su presencia a posibles equipos que puedieran estar buscando el router más cercano. Esta norma estándar por otra parte es implementada a nivel de aplicación mediante RIP ya sea con el antiguo de monio routed o el nuevo gated. Cualquier máquina puede estar corriendo uno de estos demonios y pudiera dar a conocer sus rutas, pero es obligatorio para los *routers*, de tal forma que en general las máquinas a las que no se las cofigura un *gateway* por defecto suelen escuchar dicho tráfico y hacen peticiones, pero no aceptan peticiones de otras máquinas (modo silencioso), mientras que los *routers* deben ser configurados para que si respondan ante este tipo de peticiones. En redes grandes el uso de encaminamiento dinámico puede ser una ventaja, pero en redes pequeñas y fáciles de administrar su uso suele estar desaconsejado por problemas de seguridad.

Hping y hping2 son herramientas excepcionales para la construcción de paquetes, pero por desgracia aún no soporta varios tipos de paquetes icmp (9,10,12,13,14,15,16,17) por lo que deberemos recurrir a otro tipo de herramientas como *sing* e *icmpush* entre otras.

Veamos una solicitud de router utilizando icmpush:

```
delorian:/cdrom# icmpush -vv -rts xxx.xxx.xxx.xxx ->
Outgoing interface = xxx.xxx.xxx.xxx -> ICMP total size = 8 bytes -
>
Outgoing interface = xxx.xxx.xxx.xxx -> MTU = 1500 bytes -> Total packet
size (ICMP + IP) = 28 bytes ICMP Router Solicitation packet sent to
xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx)

Receiving ICMP replies ...
-------------------------------------------------
xxx.xxx.xxx.xxx ...
         Type = Router Solicitation (0xA)
   Code = 0x0     Checksum = 0xFFF5
     Id = 0x0         Seq# = 0x0
-------------------------------------------------
xxx.xxx.xxx.xxx -> Router Advertisment (xxx.xxx.xxx.xxx)
icmpush: Program finished OK
```

A menudo un cortafuegos puede estar haciendo el papel de router interno para una DMZ (o zona desmilitarizada) pero siempre debería tener capado todo el tráfico de multicast (al igual que el broadcast) de cara afuera. De no ser esto así un posible atacante que encuentre entre varias máquinas una que conteste al tipo de solicitud de *router* (tipo 9) puede dirigir sus esfuerzos hacia este equipo con grandes posibilidades de abrir una brecha en nuestra red.

5.4.4. Peticiones ICMP de marcas de tiempo (tipo 13)

Es muy las marcas de tiempo *(timestamps)* se usan a menudo como protección para evitar la falsificación de paquetes TCP alterando su secuencialidad, pero una forma alternativa de aprovecharse de dicha información es solicitando una marca de tiempo a un servidor mediante una petición ICMP de tipo 13 a la que el servidor responderá con su hora actual. No sólo nos revelará que el equipo está activo sino que además en función de su zona horaria podremos deducir su situación geográfica.

Por desgracia es común entre los Windows no responder a este tipo de peticiones, mientras que la mayor parte de las variantes Unix responden correctamente. Esto puede utilizarse para reconocer los sistemas operativos implicados, a parte de ser muy útil para la detección de servidores *NIX.

Veamos una petición a un linux con kernel 2.2.16:

```
delorian:~# icmpush -vv -tstamp xxx.xxx.xxx.xxx
 -> Outgoing interface = xxx.xxx.xxx.xxx
 -> ICMP total size = 20 bytes
 -> Outgoing interface = xxx.xxx.xxx.xxx
 -> MTU = 1500 bytes
 -> Total packet size (ICMP + IP) = 40 bytes
ICMP Timestamp Request packet sent to xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx)

Receiving ICMP replies ...
-------------------------------------------------
xxx.xxx.xxx.xxx ...
         Type = Timestamp Request (0xD)
   Code = 0x0     Checksum = 0x83D0
     Id = 0x7B6       Seq# = 0xC2BB
-------------------------------------------------
xxx.xxx.xxx.xxx     -> Timestamp Reply transmited at 16:52:10
icmpush: Program finished OK
```

5.4.5. Peticiones ICMP de información (Descubrimiento de IP’s) (tipo 15)

Se trata de un tipo de peticiones obsoletas en la actualidad que proporcionan a un cliente la dirección de red del equipo que responde.

A pesar de ser bastante obsoleto muchos de los Unix siguen respondiendo a este tipo de peticiones mientras que los Windows no lo hacen.

Veamos una petición un Solaris 2.6:

```
delorian:~# icmpush -v -info xxx.xxx.xxx.xxx
ICMP Info Request packet sent to xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx)

Receiving ICMP replies ...
xxx.xxx.xxx.xxx     -> Info Reply (xxx.xxx.xxx.xxx)
```

```
icmpush: Program finished OK
```

5.4.6. Peticiones ICMP de máscara de red (tipo 17)

Este tipo de solicitudes se hacen para obtener la dirección de la máscara de subred en una red local. Si la petición tiene éxito una respuesta ICMP de tipo 18 deberá ser devuelta con dicha información.

A diferencia de lo que ocurría con la petición de descubrimiento de IP's de tipo 15 los Linux no responden por defecto a este tipo de peticiones, mientras que los Windows por defecto si lo hacen. Esto puede ser muy útil a la hora de detectar la presencia de Windows a parte de para obtener valiosísima información de como está estructurada la red internamente. No solo Windows responde a este tipo de peticiones sino que algunos tipos de Unix poco seguros por defecto también responden como es el caso de las diversas versiones de SunOS y Solaris.

Aquí vemos una petición icmp de máscara a un Windows 98 contestada con un icmp tipo 18 indicando una máscara de tipo C:

```
delorian:~# icmpush -v -mask xxx.xxx.xxx.xxx
ICMP Address Mask Request packet sent to xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx)

Receiving ICMP replies ...
xxx.xxx.xxx.xxx     -> Address Mask Reply (255.255.255.0)
icmpush: Program finished OK
```

5.4.7. Comentario a los análisis ICMP

A pesar de que los diversos métodos de obtener información mediante ICMP en general pueden ser bastante satisfactorios, debemos tener en cuenta que no siempre nos valen para el propósito de la detección de máquinas remotas en tanto en cuanto las propias RFC's nos aseguran que no siempre debe ser generado un paquete ICMP.

La RFC 1122 asegura que al menos hay tres posibles casos tipificados en los que un mensaje de error ICMP no debe ser enviado:

1. No se debe enviar un error ICMP de respuesta ante la recepción de otro mensaje ICMP de error.

   - Esto evita bucles de mensajes ICMP's.

2. No se debe enviar respuesta ICMP ante la recepción de un datagrama destinado a una dirección de broadcast o multicast ni siquiera un broadcast a nivel de enlace (arp's de bootp por ejemplo)

   - Esto determina que por ejemplo los paquetes UDP de videoconferencia no deberán generar tráfico ICMP de error, pues es preferible perder cierta información a saturar la línea.

3. No deberá de enviarse ninguna respuesta ICMP ante la llegada de un fragmento no inicial o un datagrama cuya dirección de origen no defina a una única máquina.

   - Por ejemplo un datagrama con dirección origen la IP de una red, de broadcast, multicast, una dirección de loopback o una dirección de clase E.

Estos tres tipos de mensajes podrían resumirse con las siguiente frase: "Nunca deberá enviarse una respuesta ICMP ante la llegada de otro paquete ICMP ni en ningún otro caso en el que o bien el destino o bien el origen no correspondan con una IP que identifique únivocamente a una máquina"

Capítulo 6. Obtención de respuesta del servidor ante una petición inválida

La mayoría de los diseñadores de sistemas han tenido más o menos presentes las RFC's pertinentes a la hora implementar cada una de las capas de la pila TCP/IP, por lo que sus respuestas son básicamente iguales a la hora de tratar con peticiones válidas tipificadas en las RFC's. Sin embargo las RFC's especialmente la RFC 791 referente al protocolo IP a menudo omite voluntariamente referencia alguna a como tratar los datagramas (IP) , segmentos (TCP) o paquetes completos inválidos. El objetivo de las RFC's es proponer un estándar que luego los diseñadores y desarrolladores deberán seguir, pero precisamente por su carácter unificador éstas a menudo dicen cómo han de hacerse la interconexión mediante implementaciones que siguen las normas, pero intentan evitar en la medida de lo posible indicar que se debe hacer con las implementaciones que se salgan de la norma.

Un ejemplo de esto es qué hacer cuando dos fragmentos distintos se solapan. La solución drástica es eliminar los dos fragmentos y por ende tirar el paquete entero y forzar la retransmisión. Dado que esto es una medida muy drástica y que disminuye el rendimiento, hay dos opciones que se pueden llevar a cabo:

- Algunos sistemas como Solaris y Windows NT prefieren quedarse con el primer fragmento que llegó tirando el último.
- Otros sistemas en cambio como Linux y *BSD prefieren quedarse con el último fragmento machacando al primero que llegó.

Esta forma diferente de actuar puede utilizarse para diversos propósito, como por ejemplo:

- Determinar el Sistema operativo de la máquina que se analiza en función de la respuesta a ciertos paquetes (firma TCP/IP).
- Evadir los sistemas de detección de intrusión basados en red (NIDS) mediante por ejemplo la inserción de fragmentos solapados.
- Detectar la presencia de máquinas dedicadas al filtrado de paquetes ya sean *routers*, cortafuegos, etc.

El uso que nosotros daremos al envío de paquetes inválidos será precisamente la detección de máquinas al responder esta a ciertos tipos de peticiones con algunos mensajes de error. Para generar la respuesta ante peticiones inválidas del servidor básicamente nos centraremos en el protocolo IP siendo éste independiente de los protocolos superiores que lo encapsulan (TCP,UDP,etc).

Principalmente existen estos tipos de peticiones no válidas ante un servidor:

- Envío de paquetes fragmentados que hacen saltar temporizadores.
- Envío de datagramas IP con longitudes de cabeceras inválidas.
- Envío de datagramas IP con contenido no válido en los campos de las cabeceras.

6.1. Generación de respuestas por timeout en la recepción de fragmentos IP

Todas las implementaciones TCP/IP definen un temporizador que inicializan ante la llegada del primer fragmento IP recibido de un paquete. Para que el paquete completo se de por válido todos los fragmentos de dicho paquete deberán llegar antes de que caduque el temporizador encargado de controlar la defragmentación. Si transcurrido dicho tiempo no se han reensamblado todos los datagramas IP por la falta de alguno de ellos se genera una respuesta ICMP Tipo 11 y código 1 indicando que el tiempo de reensamblaje de los fragmentos ha concluído *(Time Exceed Fragment Reassembly)*.

Para forzar este tipo de respuesta podemos enviar un paquete tcp con los *flags* "More fragments" y "Don’t fragment" activos simultánemente.

Veamos un ejemplo contra un Debian GNU/Linux 2.2 Potato al que enviamos un paquete tcp indicando que no se puede fragmentar y al mismo tiempo diciendo que se esperan más fragmentos:

```
delorian:/# hping2 -c 1 -x -y  xxx.xxx.xxx.xxx
default routing not present
HPING xxx.xxx.xxx.xxx (eth0 xxx.xxx.xxx.xxx): NO FLAGS are se

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

No hemos recibido ninguna respuesta, pero esto tan sólo es aparentemente, puesto que lo que está sucediendo es que *hping* no registra los datagramas icmp que genera la máquina remota.

Si ejecutarámos un tcpdump, snoop o cualquier sniffer de red veremos una respuesta similar a esta:

```
18:34:55.964952 xxx.xxx.xxx.xxx > xxx.xxx.xxx.xxx: icmp: ip reas-
sembly time exceeded (DF) [tos 0xc0]
```

En ocasiones este tipo de respuestas son filtradas, pero entonces el cliente no tiene forma de darse cuenta que debe retransmitir el paquete, da por hecho que llegó bien y sigue retransmitiendo sucesivos paquetes, hasta que recibe la indicación por parte del servidor de que reduzca la ventana de emisión y tendrá que reenviar varios paquetes de nuevo ralentizando las comunicaciones vía WAN.

6.2. Obtención de respuesta provocada por una longitud no válida de cabecera IP

El envío de un datagrama de longitud de cabecera inválida provoca que la máquina destinataria responda generando un mensaje ICMP tipo 12 indicando error en los parámetros *(Parameter Problem)*.

Los paquetes de tipo 12 a su vez pueden venir indicando dos posibles causas:

•

código 0 ⟶

El puntero indica la posición exacta del byte que causó el error.

código 2 ⟶

Longitud incorrecta indicando que el paquete entero contiene errores.

Esta es un método que a menudo logra sobrepasar muchísimos tipos de ACL's de routers no correctamente configurados.

Para poder generar un paquete así necesitaremos la ayuda de algún software especial.

Usaremos ISIC (IP Stack Integrity Checker) puede utilizarse para ensamblar un paquete con un tamaño de cabecera erróneo de 66 bytes.

```
delorian:~# isic -s ZZZ.ZZZ.ZZZ.ZZZ -d XXX.XXX.XXX.XXX -p 1 -V 0 -
F 0 -I 66
Compiled against Libnet 1.0.1b
Installing Signal Handlers.
Sending with 5099
No Maximum traffic limiter
Bad IP Version= 0%   Odd IP Header Length  = 100%  Frag'd Pcnt  = 0%
ZZZ.ZZZ.ZZZ.ZZZ-> XXX.XXX.XXX.XXX tos[137] id[0] ver[4] frag[0]

 Wrote 1 packets in 0.00s @ 5839.14 pkts/s
```

Al observar la recepción de paquetes con un tcpdump veremos un error icmp similar a éste:

```
19:34:02.717470 XXX.XXX.XXX.XXX > YYY.YYY.YYY.YYY: icmp: parameter pro-
blem - octet 20 [tos 0xd0]  (ttl 255, id 21508)
```

6.3. Obtención de respuesta provocada por contenido no válido en los campos de cabecera IP

Hay muchos campos que pueden modificarse en la cabecera de un datagrama IP. De todos ellos uno de los más interesante es el que campo llamado IP Proto que indica el protocolo de la capa superior al que hace referencia. Este campo tiene una longitud de 8 bits por lo que hay hasta 256 posibles valores.

Según las RFC's cualquier datagrama que referencie a un protocolo de nivel superior no soportado en el destino provocará que éste genere una respuesta ICMP de tipo 3 y código 2 indicando que el protocolo de destino no es válido (Destination

Unreachable/Protocol Unreachable). Cualquier protocolo que se especifique y no genere respuesta ICMP deberá considerarse válido. De esta forma no sólo daremos a conocer la existencia de un equipo en ese destino al que estamos analizando sino que además nos indicará cuales son los protocolos que soporta indicando de alguna forma el sistema operativo que utiliza, pues muchos protocolos no son implementados mientras que otros son característicos de ciertos fabricantes como Microsoft.

Podemos probarlo de muchísimas formas, por ejemplo usando la opción --ipproto o -H de hping, con programas en perl como apsend (http://www.elxsi.de) o con una versión reciente de nmap:

Una de las opciones avanzadas recientemente incorporadas al nmap es el escaneo de protocolos soportados, la cual usaremos aquí para mostrar un ejemplo:

```
delorian:~# nmap -sO xxx.xxx.xxx.xxx -p 1,6,17

Starting nmap V. 2.54BETA29 ( www.insecure.org/nmap/ )
Interesting protocols on  (xxx.xxx.xxx.xxx):
Protocol   State       Name
1          open        icmp
6          open        tcp
17         open        udp


Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

El equipo remoto no ha enviado ningún mensaje icmp, por lo que nmap ha dado por hecho que estos protocolos están disponibles. Si bien esto es cierto en este caso pues son típicos protocolos usados en internet, hay muchas implementaciones que no envían nunca el error de tipo 3 y código 2 indicando un protocolo no alcanzable, por lo que hay que tener cuidado.

Por ejemplo si escaneamos los protocolos disponibles contra un AIX de IBM obtendremos una respuesta similar a ésta:

```
delorian:~# nmap -o protocols -sO XXX.XXX.XXX.XXX -p 1,6,17,254,255

Starting nmap V. 2.54BETA29 ( www.insecure.org/nmap/ )
Interesting protocols on  (XXX.XXX.XXX.XXX):
Protocol   State       Name
1          open        icmp
6          open        tcp
17         open        udp
```

```
254         open         unknown
255         open         unknown


Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

En el caso que acabamos de ver el AIX que acabamos de atacar nos ha comunicado que incluso los tipos 254 y 255 están implementados en su pila de protocolos lo cual es totalmente falso.

Es bien conocido que diversas implementaciones de Unix no envián este tipo de mensajes icmp, por ejemplo AIX, HP-UX y Digital UNIX rehusan enviar el mensaje icmp que esperábamos por lo que éste método no nos es válido con ellos.

Capítulo 7. Agradecimientos

Quisiera agradecer en esta sección a toda la gente que con su trabajo, dedicación y esfuerzo han ayudado a elaborar este documento para el bien de toda la comunidad.

En especial quisiera destacar la magnífica ayuda prestada por Dethy (mailto:dethy@synnergy.net) al que personalmente le debo el haberme abierto los ojos por la temática de este Cómo y haber contribuído con su trabajo a cimentar las bases de este documento.

En lugar destacado quisiera también agradecer a mis compañeros:

- Francisco Javier Sucunza (mailto:sucunza@cortafuegos.org)
- Antonio Javier García (mailto:ajgarcia@cortafuegos.org)

por el ánimo que me han dado en todo momento para elaborar ésta documentación para la comunidad de Documentación y Software libre.

Agradezco enormemente que me envíen todo tipo de sugerencias, correcciones, etc. para poder mejorar la calidad del documento, al igual que pueden ponerse en contacto conmigo para realizar alguna colaboración al respecto en mi dirección de correo:
Víctor Escudero Rubio (mailto:vescudero@cortafuegos.org)

Capítulo 8. Licencia GNU de documentación libre y Copyrights