

Análisis Activo y Pasivo de Redes

> hAckMEetinG > BCN > 2000



'Si uno mira a la realidad lo suficientemente cerca, podrá ver los pixels'

www.sindominio.net/hackbcn00

Alejandro Castán Salinas

alex.castan@upc.es

Análisis activo y pasivo de redes - revisión 2.8

Copyright © Alejandro Castán Salinas

Se otorga el permiso para copiar, distribuir y/o modificar este documento bajo los términos de la licencia de documentación libre GNU, versión 1.1 o cualquier otra versión posterior publicada por la *Free Software Foundation*.

Puedes consultar dicha licencia en <http://www.gnu.org/copyleft/fdl.html>.

El contenido de este documento puede cambiar debido a ampliaciones y correcciones enviadas por los lectores. Encontrarás siempre la última versión del documento en <http://www.lsi.upc.es/~acastan/Hacking/>.

Jaime Agudo, la persona del esCert que originariamente debía venir a dar el taller de “Hacking pasivo y análisis de redes”, no pudo asistir al evento.

Ya que había muchos asistentes interesados en su taller, en el último momento me atreví a sustituirle para realizar una pequeña introducción al tema. En una hora intenté prepararme unos apuntes a modo de guión. Como los asistentes pudisteis comprobar, la charla no salió todo lo bien que me hubiera gustado, especialmente cuando hablaba de los aspectos más técnicos.

He preparado este breve documento a modo de contenido de lo que fue mi charla para toda la gente interesada que no pudo asistir y para la gente que pudo asistir pero se lió con mi desastre de exposición.

No sé si es correcto dedicar un resumen o unos apuntes. Si lo fuera, quiero dedicarlos a toda la gente que me animó asistiendo a la charla y aguantó mi “chapa” durante una hora. La próxima vez lo haré mejor ;-)

También quiero dedicarlos a la gente de Cathack!, un grupo dedicado al hacking que da sus primeros pasos, lleno de gente maravillosa a la que me alegro de haber conocido. <http://cathack.hn.org/>

¡Nos vemos en el Hackmeeting Leioa 2001!

Análisis activo y pasivo de redes

Introducción

En la preparación de un ataque remoto a un sistema informático, el atacante no sólo necesita conocer la dirección en la red de la víctima, sino también obtener la máxima información sobre ella.

En Internet, no basta con conocer su dirección IP o URL. Si el atacante logra conocer el sistema operativo (por ej.: Linux, FreeBSD, Solaris, Windows NT, etc.) y los servicios (por ej.: ftp, telnet, http, smtp, etc.) de la víctima, este podrá precisar más su ataque. Si además conoce la versión del sistema operativo (por ej.: Linux 2.0.35 o Linux 2.2.17) y de los servicios (por ej. wuftp 2.6.0 o wuftp 2.6.1) todavía podrá precisar mucho más su ataque, ya que muchos agujeros de seguridad dependen en gran manera de una determinada versión de sistema operativo o de los servicios que sobre él corren.

En las siguientes líneas realizaré una breve introducción al concepto de puerto y al proceso de establecimiento de una conexión TCP, necesarios para comprender puntos posteriores.

Dos protocolos separados son los encargados de manejar los mensajes TCP/IP (en el anexo D de este documento se entra un poco más en detalle sobre estos protocolos). TCP (“Transmission Control Protocol”) es el responsable de romper el mensaje en datagramas, ensamblar los datagramas en el otro extremo, reenviar toda la información extraviada y poner la información de nuevo en el orden correcto. IP (“Internet Protocol”) es el responsable de encaminar los datagramas individualmente.

Para el seguimiento de conversaciones individuales entre un cliente y un servicio, TCP utiliza un número de puerto asociado a dicho servicio (la lista con los números de puerto más utilizados y su servicio asociado se encuentran en el anexo E de este documento). Así, la conexión queda descrita por la dirección de Internet y el número de puerto de cada extremo.

Los números de puertos están divididos en tres rangos: los puertos bien conocidos (del 0 al 1023), los puertos registrados (del 1024 al 49151) y los puertos dinámicos y/o privados (del 49152 hasta el 65535). Los puertos bien conocidos son asignados y controlados por un organismo llamado IANA. En la mayoría de sistemas estos puertos tan sólo pueden ser usados por los procesos del sistema y por programas ejecutados por usuarios privilegiados. Los puertos registrados no son controlados por IANA. Estos puertos pueden ser utilizados por procesos de usuarios ordinarios y por programas ejecutados por usuarios sin privilegios de sistema.

Por ejemplo, pensemos en el caso de querer enviar un fichero a través de Internet. En dicho proceso quedan involucrados dos programas: en nuestro extremo el programa cliente de FTP, que acepta comandos desde el terminal y los envía al otro extremo, donde reside el programa servidor de FTP, que interpreta y ejecuta los comandos recibidos. El programa cliente de FTP abrirá una conexión utilizando en nuestro extremo un número aleatorio de puerto, por ej. 1234, y en el otro extremo el puerto 21, que es el número de puerto oficial para el programa servidor de FTP. El cliente de FTP no necesita utilizar para él un número de puerto bien conocido, ya que nadie trata de localizarlo. Sin embargo, el servidor de FTP si necesita un número de puerto bien conocido para que los clientes puedan iniciar una conexión y enviarle comandos. Así, cada datagrama llevará las direcciones de Internet de cada extremo en la cabecera IP, y los números de puerto de cada extremo en la cabecera TCP. Dos conexiones no pueden tener los mismos números, pero basta que un número de puerto sea diferente para que esto sea posible. En nuestro ejemplo, dos usuarios en nuestra máquina pueden enviar simultáneamente dos ficheros a otra máquina utilizando los siguientes parámetros:

	dirección origen	puerto origen	dirección destino	puerto destino
conexión 1	147.83.170.210	1234	147.83.2.11	21
conexión 2	147.83.170.210	1235	147.83.2.11	21

En el inicio de una conexión TCP entre dos ordenadores se produce un proceso previo al envío de información, que consiste en un saludo en tres pasos que garantiza que ambos lados estén preparados para transferir datos, tengan conocimiento de que el otro también lo está y acuerden un número de secuencia inicial. Escuetamente explicado, (1) el ordenador que desea iniciar la conexión envía al ordenador del otro extremo un segmento con el bit SYN activado en el campo de código y un número de secuencia inicial. El ordenador en el otro extremo recibe el segmento y, si puede iniciar la conexión (puerto de destino abierto = servicio disponible), (2) responde a su vez con un segmento con los bits SYN y ACK activados, un acuse de recibo que es el número de secuencia inicial más uno del cliente, y su propio número de secuencia inicial. A este segmento de respuesta, (3) el ordenador origen responde con un segmento con el bit ACK activado y se inicia el envío de información. Si el ordenador destino no puede iniciar la conexión (puerto de destino cerrado = servicio no disponible), al segmento SYN inicial responde con un segmento con el bit de RST activado. A este segmento de respuesta, el ordenador origen no responde nada y se cierra la conexión.

(1)	cliente	---SYN--->	servidor
(2)	cliente	<---SYN/ACK---	servidor
(3)	cliente	---ACK--->	servidor

Análisis activo de puertos

El análisis activo de puertos consiste en conocer qué servicios tiene disponibles un ordenador en la red, enviando determinados paquetes TCP y UDP a sus puertos para comprobar cuáles están abiertos (es decir, esperando una conexión) y cuáles cerrados. Un resumen de las diferentes técnicas de análisis activo de puertos se encuentra en el anexo A de este documento.

Su principal desventaja reside en que el envío de estos paquetes “sospechosos” o con características especiales es fácilmente detectable por un sistema de detección de intrusos (IDS), pudiendo el ordenador que sufre el análisis de puertos registrar dicho análisis y obtener la dirección IP del ordenador que la realiza.

Análisis activo del sistema operativo

Existen numerosas técnicas para el reconocimiento del sistema operativo de un ordenador en la red. Técnicas tradicionales, como comprobar los mensajes iniciales en las conexiones vía TELNET o FTP, se pueden prevenir fácilmente y son de una efectividad limitada. Las técnicas de mayor auge hoy en día están basadas en que cada sistema operativo implementa de una manera diferente su pila TCP, es decir, que procesan y responden de manera diferente ante un mismo mensaje TCP/IP, especialmente si se trata de un mensaje incorrecto. Así, para identificar un sistema operativo basta con enviar una serie de mensajes y comprobar los valores de respuesta en una tabla. Un resumen de las diferentes técnicas de análisis activo de sistemas operativos se encuentra en el anexo B de este documento.

De manera similar al análisis activo de puertos, en el análisis activo de sistemas operativos el envío de paquetes “especiales” TCP es fácilmente detectable por un sistema de detección de intrusos, que obtendrá la dirección IP del ordenador que realizó el análisis.

Mejoras al análisis activo

Existen algunas técnicas de mejora del análisis activo que intentan subsanar el problema de la fácil detección de éste por sistemas de detección de intrusos. Entre estas técnicas destacan el análisis al azar de puertos, el análisis lento, el análisis distribuido, la fragmentación de paquetes, el análisis a través de proxy y el análisis con señuelos.

El análisis al azar de puertos intenta burlar algunos sistemas de detección de intrusos realizando el análisis de puertos en orden aleatorio o pseudo-aleatorio, en lugar de en orden secuencial. También se puede aleatorizar el orden de las direcciones IP analizadas, el intervalo de tiempo entre las pruebas, y los valores de algunos campos no esenciales de los paquetes enviados para realizar el análisis, como el número de secuencia, el número de acuse de recibo, el identificador IP y el puerto de origen.

En el análisis lento se intenta hacer la espera entre el análisis de un puerto y el siguiente lo suficientemente larga como para no ser detectada como análisis.

En el análisis distribuido se utilizan simultáneamente varios ordenadores para realizar el análisis de manera coordinada. Este método de análisis, combinado con el análisis lento y el análisis al azar de puertos, es muy efectivo y prácticamente indetectable. Imaginemos, por ejemplo, una docena de ordenadores situados en diferentes puntos de Internet analizando los ordenadores de una gran red, a razón de dos puertos al día por ordenador víctima. En pocos días obtendrían el mapa de dicha red. Otras ventajas del análisis distribuido es la adquisición de un modelo más completo de la víctima, que incluiría información sobre múltiples rutas y la ruta más rápida.

La fragmentación de paquetes consiste en romper los paquetes IP utilizados en el análisis en fragmentos lo suficientemente pequeños como para que la información de la cabecera de los datagramas TCP o UDP que contienen también quede dividida. De esta manera los cortafuegos y sistemas de detección de intrusos que no poseen la característica de cola de ensamblaje dejan pasar los fragmentos, que se reensamblan en la pila TCP/IP del ordenador víctima. Existen dos versiones de este método. La primera versión consiste en enviar fragmentos tan pequeños (8 bytes de datos) que las señales de código del datagrama TCP no viajan en el primer paquete IP. La segunda versión consiste en enviar el primer fragmento con un puerto destino y señales de código válidas y aceptables, pero un segundo fragmento con un desplazamiento negativo que machaca dicha información cuando se reensambla el paquete.

En el análisis a través de proxy, aunque dicho análisis sea detectado, se obtendrá la dirección del proxy a través de la cual se realiza el análisis, pero no la dirección real del ordenador origen de dicho análisis. Un tipo especial de análisis a través de proxy es el *FTP bounce scan*.

De manera similar al análisis a través de proxy, en el análisis con señuelos también es difícil de obtener la dirección real del ordenador que realiza el análisis, aunque dicho análisis sea detectado. Consiste en realizar una gran cantidad de análisis de puertos simultáneos sobre una máquina, pero todos los análisis menos uno con la dirección de origen falsa. De esta manera, a la víctima le resultará prácticamente imposible encontrar la dirección verdadera de entre los centenares de direcciones falsas. Una manera de dificultar la búsqueda de la dirección verdadera es utilizar en los paquetes IP tiempos de vida aleatorios y direcciones IP origen de ordenadores activos. Una variante de este método consiste en realizar análisis con la dirección de origen falseada hasta llenar por completo con avisos falsos el registro de eventos del sistema de detección de intrusos. Una vez queda éste registro desbordado, pueden pasar dos cosas: que no se puedan anotar los eventos de nuevos análisis o que los eventos de los análisis más antiguos sean borrados.

Análisis pasivo del sistema operativo

A diferencia del análisis activo de puertos y de sistemas operativos, el análisis pasivo no consiste en enviar información al ordenador a analizar, sino en esperar a recibirla cuando éste establece una conexión a nuestro ordenador. Los paquetes capturados contienen suficiente información para determinar el sistema operativo con que funciona. La combinación de los valores del tiempo inicial de vida (8 bits), el tamaño de ventana (16 bits), el tamaño máximo de segmento (16 bits), el bit de no fragmentación (1 bit), la opción sackOK (1 bit), la opción NOP (1 bit) y la opción de escalado de ventana (8 bits) forman una firma de 51 bits única para cada sistema.

El anexo C de este documento muestra las tablas utilizadas por dos programas de análisis pasivo de redes. Dichas tablas contienen valores utilizados por un ordenador en el primer paquete de establecimiento de la conexión (primer SYN del saludo TCP de tres tiempos).

Cabe recordar que el campo tiempo inicial de vida (TTL) es un número (normalmente una potencia de dos) que indica el tiempo de vida de un paquete IP desde que parte del ordenador origen. Cada vez que dicho paquete se encamina por una nueva red, el valor TTL se decrementa en una unidad. Si dicho valor llega a cero el paquete se destruye. Por lo tanto, en las tablas la columna TTL indica la primera potencia de dos superior al valor devuelto en el análisis.

Como curiosidad, podemos ver la ruta seguida por el paquete IP sin necesidad de alertar al ordenador que lo envió realizando un *traceroute* a la dirección IP origen con el valor de tiempo de vida $2^n - TTL - 1$. Por ejemplo, si recibimos un paquete con dirección IP de origen 147.83.170.211 con el valor 57 en el campo TTL, supondremos que partió con un valor inicial de 64 en el campo TTL y que pasó por siete enrutadores antes de llegar a nuestro ordenador. Nos mostrará el camino pues:

```
traceroute -m 6 147.83.170.211
```

En el análisis pasivo existen todavía numerosos campos y valores a explorar. Por ejemplo, los números iniciales de secuencia, los números de identificación IP, algunas opciones TCP e IP, el tipo de servicio o TOS (aunque el valor de éste último parece que depende más del protocolo que del sistema operativo), etc.

Existe otro método de análisis pasivo, alternativo al método recién explicado de análisis de paquetes IP, que consiste en recabar información proporcionada en el nivel de aplicación. Numerosos programas adjuntan a los datos que envían por Internet información suficiente para identificar el sistema operativo y hardware del sistema que los envió. Por ejemplo: los programas clientes de correo electrónico y grupos noticias acostumbran a incorporar información del usuario en el campo X-Mailer de la cabecera; las páginas web incorporan información en los campos User-Agent, Host y Server de la cabecera; cada cliente de telnet negocia la velocidad de línea, el tipo de terminal y el eco de manera diferente; etc.

Ventajas del análisis pasivo

- El análisis pasivo es imposible de detectar ya que, a diferencia del análisis activo, no enviamos ninguna información sospechosa al ordenador a analizar.
- Permanecer a la escucha de conexiones permite descubrir ordenadores que están activos durante un breve lapso de tiempo. En Internet existen servidores que tan solo funcionan durante los segundos en que envían y reciben los datos.
- Permanecer a la escucha de conexiones permite descubrir servicios ocultos. Normalmente en un análisis activo no se suelen analizar todos los puertos, ya que son 65536 y ello llevaría bastante tiempo, sino que sólo se analizan los más conocidos o los que residen en números bajos. Mediante el análisis pasivo podemos descubrir si se están utilizando puertos poco conocidos para servicios especiales o como puerta de acceso de algún troyano. Basta con buscar las conexiones SYN|ACK de los puertos por encima de 1024.
- El análisis pasivo permite identificar cortafuegos proxy remotos, ya que éstos reconstruyen las conexiones de sus clientes.

Limitaciones del análisis pasivo

- Es poco específico, en el sentido de que cuesta analizar un ordenador en concreto debido a que se debe esperar un intento de conexión por parte de dicha máquina. En el caso de que el ordenador a analizar sea un servidor de páginas web, basta con solicitar una página cualquiera (conducta normal que no generará sospechas) y analizar su respuesta. Aún así, en

la mayoría de casos no se puede escoger un ordenador ni unos servicios específicos a analizar.

- En el caso de haber sufrido un análisis activo, si se quiere conocer el sistema operativo del ordenador origen del análisis a partir del análisis pasivo de los paquetes TCP/IP enviados por éste, se debe tener en cuenta que la mayoría de los paquetes generados por herramientas de análisis activo difieren de los generados por defecto por el sistema operativo, lo que dará lugar a error.
- Es fácil cambiar los parámetros que son observados por un análisis pasivo. Por ejemplo, para cambiar el valor del campo tiempo de vida en diferentes sistemas operativos:

```
Solaris: ndd -set /dev/ip ip_def_ttl 'numero'  
Linux: echo 'numero' > /proc/sys/net/ipv4/ip_default_ttl  
NT: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

Usos del análisis pasivo

Previamente al uso de una herramienta de análisis pasivo debe instalarse un “sniffer”, que no es más que un programa o dispositivo que monitoriza todo el tráfico que circula por la red. El “sniffer” pone a trabajar a la tarjeta de red del ordenador donde está instalado en un modo denominado promiscuo, en el que la tarjeta escucha tanto los paquetes que van dirigidos explícitamente a su estación de trabajo como los que van dirigidos a otras estaciones. En un principio, una máquina no debería estar trabajando en modo promiscuo a menos que existiera una buena razón. Por ello, encontrar una tarjeta de red funcionando en dicho modo es un fuerte indicador de que un “sniffer” está espiando el tráfico de la red. Existen varios métodos para comprobar si una tarjeta de red está funcionando en modo promiscuo. El más sencillo y rápido es ejecutar el comando Unix `ifconfig -a`, aunque se debe tener en cuenta que la mayoría de “rootkits” substituyen dicha instrucción por otra falsa que impide detectar el “sniffer”.

El análisis pasivo puede tener distintos usos o motivaciones.

- Atacantes pueden determinar el sistema operativo de una víctima en potencia, como por ejemplo un servidor de páginas web, solicitando una página de dicho servidor, que es una conducta normal que no levantará sospechas, y analizando después los rastros del sniffer.
- Organizaciones pueden inventariar rápidamente los sistemas operativos de los ordenadores de sus redes sin alterar el rendimiento de dichas redes, e identificar tanto sistemas críticos (por ej. superordenadores centrales), como sistemas no autorizados (por ej. si un empleado de Microsoft o Sun ha instalado Linux o FreeBSD en su ordenador).
- Los sistemas de detección de intrusos (IDS) pueden incorporar herramientas de análisis pasivo para detectar el sistema operativo de las máquinas que han realizado un análisis activo sobre un sistema, ya que los paquetes enviados por algunas herramientas de análisis activo heredan valores del sistema operativo subyacente sobre el cual trabajan.
- Escuchando el tráfico en puntos críticos o de choque de Internet, se pueden utilizar los datos obtenidos para mapear redes. Es decir, no sólo mapear la propia red, sino mapear lentamente las redes donde se dirigen los usuarios y las redes de donde vienen solicitudes de servicios. Una red grande y distribuida de filtros puede obtener mapas de redes de calidad. Esto no es teoría, si no que ya lo están utilizando estados para mapear las redes de otros países. Por ejemplo, el proyecto SORM-2 de Rusia consta de 350 proveedores de acceso a Internet y se está utilizando para mapear redes de dentro y fuera del país. ¡Comienza la guerra electrónica!

Bibliografía

- Un excelente artículo sobre análisis activo de puertos es “The Art of Port Scanning”, que encontrareis en http://www.insecure.org/nmap/nmap_doc.html.

- Un excelente artículo sobre análisis activo de sistemas operativos es “Remote OS detection via TCP/IP Stack FingerPrinting”, que encontrareis en <http://www.insecure.org/nmap/nmap-fingerprinting-article.html>.
- Un artículo sobre análisis pasivo de sistemas operativos es “Passive FingerPrinting”, que encontrareis en <http://project.honeynet.org/papers/finger/>.
- Un artículo que permite observar el estado del arte en análisis mediante paquetes ICMP es “ICMP Usage in Scanning”, que encontrareis en http://www.sys-security.com/archive/papers/ICMP_Scanning_v3.0.pdf.
- Una lectura imprescindible para aprender algo más acerca de sniffers: “Sniffing (network wiretap, sniffer) FAQ”, en <http://www.robertgraham.com/pubs/sniffing-faq.html>.

Software

- Una herramienta excelente para el análisis activo de puertos y sistemas operativos es *nmap*. La encontrareis en <http://www.insecure.org/nmap/>. Otra herramienta también valiosa, que permite enviar paquetes a medida, es *hping3*. La encontrareis en <http://www.kyuzz.org/antirez/hping.html>.
- Dos herramientas para el análisis pasivo de sistemas operativos son *siphon* y *p0f*. Las encontrareis respectivamente en <http://gravitino.net/projects/siphon/> y <http://www.stearns.org/p0f/>, o buscando en el repositorio <http://packetstorm.decepticons.org>.
- Un conocido sniffer para el tráfico TCP es *tcpdump*, que encontrareis en <http://www.tcpdump.org>. Otro conocido sniffer es *ethereal*, que encontrareis en <http://www.ethereal.com>. Por último, *dsniff* es una imprescindible colección de herramientas para monitorizar redes, que encontrareis en <http://www.monkey.org/~dugsong/dsniff/>.

Una herramienta que aparecía referenciada en el taller de Jaime Agudo era *nstreams*, que encontrareis en <http://www.hsc.fr/ressources/outils/nstreams/>. Dicha herramienta interpreta la salida de `tcpdump -n`.

En *tcpdump* se puede conseguir aumentar la velocidad de filtrado para el análisis pasivo con el siguiente filtro: `tcpdump -q -n 'tcp[13] = 18'`.

- Para detectar tarjetas de red funcionando en modo promiscuo, indicador de que probablemente haya un sniffer instalado en vuestra red, recomiendo *antisniff*, cuya versión de evaluación encontrareis en <http://www.securitysoftwaretech.com/antisniff/>, y *check promiscuous mode*, en <ftp://coast.cs.purdue.edu/pub/tools/unix/sysutils/cpm/>.
- Existen varias herramientas que permiten detectar y analizar escaneados de puertos. Mis preferidas són: *snort*, que encontrareis en <http://www.snort.org>; *portsentry*, que encontrareis en <http://www.psionic.com/products/portsentry.html>; y *scanlogd*, que encontrareis en <http://www.openwall.com/scanlogd/>. También están: *iplog*, en <http://ojnk.sourceforge.net/tcplogd>; en <http://kalug.lug.net/tcplogd/>; y *astaro*, en <http://download.astaro.com/patches/>.
- Como curiosidad: una página web que permite analizar el sistema operativo de un ordenador conectado a Internet es <http://www.netcraft.net>.

Anexo A: resumen de técnicas para el análisis activo de puertos abiertos

- **TCP connect() scanning.** Es la forma más básica de análisis de puertos. Se intenta establecer una conexión normal al puerto mediante la llamada `connect()` del sistema.

```
Origen --(SYN)-> Destino --+
                                +--(RST|ACK puerto cerrado)-> Origen
                                +--(SYN|ACK puerto abierto)-> Origen --(ACK)-> Destino
```

Ventajas: (1) no se necesita privilegios especiales para realizar el análisis y (2) se consigue una gran velocidad al analizar puertos en paralelo.

Desventajas: (1) muy fácil de detectar y filtrar, ya que en los registros del sistema para cada puerto analizado aparece que se ha intentado establecer conexión y a continuación se ha cerrado sin enviar la información.

- **TCP SYN scanning.** No establece una conexión TCP completa, sino que cuando recibe la respuesta SYN|ACK indicando que el puerto está a la escucha, inmediatamente envía un paquete RST para romper la conexión. Existe otra variante que no envía el paquete RST y, por lo tanto, deja el proceso de establecimiento de la conexión a medias.

```
Origen --(SYN)-> Destino --+
                                +--(RST|ACK puerto cerrado)-> Origen
                                +--(SYN|ACK puerto abierto)-> Origen --(RST)-> Destino
```

Ventajas: (1) pocos sitios registran este intento de conexión análisis y (2) se consigue una gran velocidad al analizar puertos en paralelo.

Desventajas: (1) hacen falta privilegios de administrador para construir el paquete SYN inicial.

- **TCP SYN|ACK scanning.** Salta el primer paso en el establecimiento de conexión TCP, enviando directamente un paquete SYN|ACK al ordenador destino. Si el puerto está abierto no se recibe respuesta, pero si está cerrado se recibe RST. En este caso podemos determinar que puertos están cerrados y, por exclusión, cuales están abiertos (mapeo inverso). En las técnicas de mapeo inverso, se pueden producir lentos falsos positivos debido a paquetes destruidos, ya sea por la acción de cortafuegos, filtros de paquetes o límites de tiempo.

```
Origen --(SYN|ACK)-> Destino --+
                                +--(RST puerto cerrado)-> Origen
                                +--(puerto abierto)
```

Ventajas: (1) los paquetes SYN|ACK son capaces de pasar a través de cortafuegos y sistemas de detección de intrusos que filtran paquetes SYN a puertos restringidos.

Desventajas: (1) Se pueden producir falsos positivos lentos y (2) la familia de sistemas BSD (BSD, OpenBSD, NetBSD y FreeBSD) ignoran los paquetes SYN|ACK sea cual sea el estado del puerto.

- **TCP ACK scanning.** Consiste en enviar un paquete ACK al ordenador destino, que siempre responderá con un paquete RST. No obstante, si el puerto está abierto, el valor del campo TTL será menor que 64 y el valor del campo win será diferente de 0.

```
Origen --(ACK)-> Destino --+
                                +--(RST puerto cerrado)-> Origen
                                +--(RST win#0 ttl<64 puerto abierto)-> Origen
```

Ventajas: (1) los paquetes ACK se pueden utilizar para mapear el conjunto de reglas de algunos cortafuegos que no devuelven respuesta para los puertos filtrados.

Desventajas: (1) Su funcionamiento depende del sistema operativo del ordenador analizado, que debe ser de tipo Unix.

- **TCP FIN scanning.** Ante un paquete FIN, los puertos cerrados deberían replicar con el debido RST y los puertos abiertos deberían ignorar el paquete FIN (mapeo inverso).

```

                                +--(RST|ACK puerto cerrado)-> Origen
Origen --(FIN)-> Destino --+
                                +--(puerto abierto)

```

Ventajas: (1) los paquetes FIN son capaces de pasar a través de cortafuegos que filtran paquetes SYN a puertos restringidos.

Desventajas: (1) Algunos sistemas (por ej. Microsoft) responden paquetes RST sea cual sea el estado del puerto y (2) se pueden producir falsos positivos lentos.

- **TCP Null scanning.** Consiste en enviar un paquete con todas las señales de código (URG, ACK, PSH, RST, SYN y FIN) de la cabecera TCP desactivadas. Si el puerto está abierto, no se recibe respuesta (mapeo inverso), pero si está cerrado se recibe RST|ACK.

```

                                +--(RST|ACK puerto cerrado)-> Origen
Origen --()-> Destino --+
                                +--(puerto abierto)

```

Ventajas: (1) los paquetes NULL son capaces de evitar algunos sistemas de detección de intrusos.

Desventajas: (1) Su funcionamiento depende del sistema operativo del ordenador analizado, que debe ser una variante de Unix, (2) es fácil de detectar y registrar, y (3) se pueden producir falsos positivos lentos.

- **TCP Xmas scanning.** Consiste en enviar un paquete con todas las señales de código (URG, ACK, PSH, RST, SYN y FIN) de la cabecera TCP activadas. Si el puerto está abierto, no se recibe respuesta (mapeo inverso), pero si está cerrado se recibe RST|ACK.

```

                                +--(RST|ACK puerto cerrado)-> Origen
Origen --(URG|ACK|PSH|RST|SYN|FIN)-> Destino --+
                                +--(puerto abierto)

```

Ventajas: (1) los paquetes XMAS son capaces de evitar algunos sistemas de detección de intrusos.

Desventajas: (1) Su funcionamiento depende del sistema operativo del ordenador analizado, que debe ser una variante de Unix, (2) es fácil de detectar y registrar, y (3) se pueden producir falsos positivos lentos.

- **UDP ICMP port unreachable scanning.** Utiliza el protocolo UDP en lugar de TCP. En dicho protocolo los puertos abiertos no envían paquetes ACK en respuesta a las pruebas y los puertos cerrados no están obligados a enviar un paquete RST. Afortunadamente, muchos puertos cerrados envían un error ICMP_PORT_UNREACH. En este caso podemos determinar que puertos están cerrados y, por exclusión, cuáles están abiertos.

```

                                +--(ICMP port_unreachable puerto cerrado)-> Origen
Origen --(UDP)-> Destino --+
                                +--(puerto abierto o cerrado)

```

Ventajas: (1) nos permite saber que puertos UDP están abiertos. (Por ejemplo, en un agujero de seguridad del *rpcbin* de Solaris, se encontró que éste escondía un puerto indocumentado por encima de 32770, así que no importaba que su puerto 111 estuviera bloqueado por un cortafuegos).

Desventajas: (1) hacen falta privilegios de administrador, (2) es lento y (3) no se garantiza la llegada ni de los paquetes UDP ni de los errores ICMP, así que en el análisis se pueden encontrar falsos positivos debidos a paquetes que no han llegado.

- **UDP recvfrom() and write() scanning.** Los usuarios no privilegiados no pueden leer directamente los errores de puertos inalcanzables. Algunos sistemas, como Linux, son capaces de informar al usuario indirectamente. Por ejemplo, una segunda llamada a *write()* sobre un puerto cerrado fallará. Otro ejemplo, una llamada a *recvfrom()* sobre sockets UDP

no bloqueados normalmente devuelve EAGAIN (error nº 13 “Try Again”) si el error ICMP no ha sido recibido, y ECONNREFUSED (error nº 111 “Connection refused”) en caso contrario.

Ventajas: (1) No son necesarios privilegios de administrador.

Desventajas: (1) No funciona para todos los sistemas.

- **TCP reverse ident scanning.** El protocolo IDENT, normalmente asociado al puerto 113, permite descubrir el nombre de usuario propietario de cualquier proceso conectado mediante TCP. Por ejemplo, si recibimos que el propietario del proceso del puerto 80 es *root*, esto quiere decir que el servidor de web se está ejecutando con privilegios de administrador.

Ventajas: (1) no se necesita privilegios especiales para realizar el análisis y (2) funciona aunque el proceso no inicie la conexión.

Desventajas: (1) este análisis es fácilmente detectable, ya que tan solo puede realizarse con una conexión TCP completa al puerto en cuestión.

- **Fragmentation scanning.** No es un nuevo método, sino una modificación de otras técnicas. Consiste en romper el paquete TCP de prueba en pequeños fragmentos IP, tales que la cabecera TCP queda dividida en paquetes tan pequeños que el primero de ellos no llega a incluir el número de puerto.

Otra variante es utilizar fragmentos con desplazamientos ilegales, tales que el primer fragmento es correcto y aceptado, pero parte de su información (por ejemplo: el puerto de destino) queda superpuesta por otro fragmento, cuando se reconstruye el paquete.

Ventajas: (1) este análisis es difícil de detectar y filtrar.

Desventajas: (1) Algunos programas tienen problemas (se cuelgan) al recibir este tipo de paquetes tan pequeños y (2) no funciona con cortafuegos y filtros que encolan y reconstruyen los fragmentos IP antes de procesarlos.

- **FTP bounce scanning.** El protocolo FTP permite conexiones “proxy”. En otras palabras, desde un ordenador se puede conectar a un servidor de FTP para solicitarle que envíe un fichero a cualquier parte de Internet o lo reciba. Dicha característica se puede aprovechar para analizar puertos TCP, ejecutando el comando PORT del servidor de FTP “proxy” para indicarle que escuche un determinado puerto del ordenador víctima. Si dicho servidor consigue establecer una conexión (puerto abierto) responderá con un mensaje FTP 150 y 226. En caso contrario (puerto cerrado) responderá con el mensaje de error FTP 425.

Ventajas: (1) es difícil realizar un rastreo del análisis de puertos y (2) puede traspasar cortafuegos, conectándose a un servidor FTP detrás del cortafuegos y después analizando puertos que estén bloqueados. Incluso, si el servidor FTP permite leer y escribir en algún directorio (normalmente el directorio */incoming*) se puede enviar datos a los puertos que se encuentren abiertos.

Desventajas: (1) este tipo de análisis es lento y (2) muchos servidores de FTP ya han deshabilitado la característica de “proxy”.

- **Dumb host scanning.** Se trata de un análisis de puertos con la dirección IP de origen falseada por la de un ordenador intermedio “sin tráfico”, es decir, que no envíe paquetes mientras se analiza al destino. En Internet hay muchos ordenadores de este tipo, especialmente por la noche. Para saber si un puerto del ordenador destino está abierto, bastará escuchar el tráfico del ordenador intermedio, ya que éste generará paquetes RST. En cambio, si el puerto del ordenador destino está cerrado no generará ningún tipo de tráfico.

```
Origin --(spoofed SYN)-> Destino --+
                                     +--(RST puerto cerrado)-> Intermedio
                                     +--(SYN|ACK puerto abierto)-> Intermedio --(RST)-> Destino
```

Ventajas: (1) El ordenador origen del análisis es difícil de detectar.

Desventajas: (1) Hace falta encontrar un ordenador intermedio sin tráfico.

- **ICMP echo scanning.** No se trata realmente de un análisis de puertos, ya que ICMP no tiene una abstracción de puertos. Consiste en determinar que ordenadores de una red están activos realizando un *ping* directamente sobre ellos o sobre la dirección de *broadcast* de la red.

```
Origen -- (ICMP echo_request) -> Destino ---+
                                         +-- (ICMP echo_reply activo) -> Origen
                                         +-- (muerto)
```

Ventajas: (1) es rápido, ya que dicho análisis se puede realizar en paralelo.

Desventajas: (1) los *ping* quedan registrados.

- **IP protocol scanning.** No se trata realmente de un análisis de puertos, sino que consiste en determinar qué protocolos IP son soportados por la pila IP del ordenador destino (existen numerosos protocolos sobre IP aparte de los archiconocidos TCP, UDP e ICMP: IGMP, IGP, EGP, GRE, SWIPE, NARP, MOBILE, SUN-ND, EIGRP, OSPFIGP, IPIP, PIM, etc.). La técnica consiste en enviar paquetes IP con los diferentes números de protocolo a probar. Si el protocolo no se utiliza recibiremos el mensaje ICMP de “protocolo inalcanzable”. En caso contrario no obtendremos respuesta, asumiendo entonces que el protocolo sí se utiliza o que ha sido filtrado por un encaminador.

```
Origen -- (IP protocol) -> Destino ---+
                                         +-- (ICMP protocol_unreachable cerrado) -> Origen
                                         +-- (abierto o filtrado)
```

Ventajas: (1) permite conocer otros protocolos utilizados aparte de TCP, UDP e ICMP, y (2) permite conocer el sistema operativo, ya que muchos protocolos soportados son característicos de un fabricante, mientras que otros no son implementados.

Desventajas: (1) algunos sistemas operativos (AIX, HP-UX, Digital UNIX) y cortafuegos pueden no enviar los mensajes de protocolo inalcanzable, causando que todos los protocolos aparezcan como “abiertos”.

Podéis probar la mayoría de estos análisis con la herramienta *nmap*, y estudiar el tráfico generado con las herramientas *tcpdump* y *snort*. La línea de comandos de *nmap* es:

```
nmap [-s<tipo paquete>] [-p <puertos>] [-<otras opciones>] <dirección destino>
```

TCP connect scan: `nmap -sT <destino>`

TCP SYN scan: `nmap -sS <destino>`

TCP ACK scan: `nmap -sA <destino>` y `nmap -sW <destino>`

TCP FIN scan: `nmap -sF <destino>`

TCP Null scan: `nmap -sN <destino>`

TCP Xmas scan: `nmap -sX <destino>`

UDP scan: `nmap -sU <destino>`

RPC scan: `nmap -sR <destino>`

TCP reverse ident scan: `nmap -I <destino>`

ICMP echo scan: `nmap -sP <destino>`

IP protocol scan: `nmap -sO <destino>`

Operative System scan: `nmap -O <destino>`

Slow scan: `nmap --scan_delay <milisegundos> <destino>`

Fragmentation scan: `nmap -f <destino>`

FTP bounce scan: `nmap -b <[usuario:contraseña@]direcciónFTP[:puerto]> <destino>`

Spoofed scan: `nmap -S <origen> <destino>`

Decoy scan: `nmap -D <señuelo1 [,señuelo2][,ME],...> <destino>`

Dumb host scan: `nmap -sI <intermedio[:puerto]> <destino>`

Random scan: `nmap --randomize_hosts <destino>` (por defecto, los puertos destino ya están en orden aleatorio)

Ejemplos de uso:

```
nmap -h
nmap -v destino.ejemplo.com
nmap -sS -O -I destino.ejemplo.com/24
nmap -sX -f -p 22,53,110,143,4564 198.116.*.1-127
nmap --randomize_hosts -p 80 '*.*.2.3-5'
```

Las anotaciones del preprocesador de análisis de puertos de *snort* son fáciles de leer. Su estructura básica es:

```
fecha hora origen:puerto -> destino:puerto tipo_paquete
```

A continuación vemos las huellas que deja en *snort* un análisis horizontal, en el que el atacante conoce una vulnerabilidad en un servicio y está intentando encontrar todas las máquinas que exponen dicho servicio. El atacante analiza el mismo puerto para un rango de direcciones IP.

```
Apr 1 19:02:12 666.66.666.66:1078 -> 111.11.11.197:53 SYNFIN
Apr 1 19:02:12 666.66.666.66:1079 -> 111.11.11.198:53 SYNFIN
Apr 1 19:02:12 666.66.666.66:1080 -> 111.11.11.199:53 SYNFIN
Apr 1 19:02:12 666.66.666.66:1081 -> 111.11.11.200:53 SYNFIN
Apr 1 19:02:12 666.66.666.66:1082 -> 111.11.11.201:53 SYNFIN
Apr 1 19:02:12 666.66.666.66:1083 -> 111.11.11.202:53 SYNFIN
```

A continuación vemos las huellas que deja en *snort* un análisis vertical, en el que el atacante está interesado en una máquina en particular e intenta averiguar todos los servicios que esta dispone, seguramente con la intención de a continuación buscar en Internet programas que exploten vulnerabilidades en dichos servicios.

```
Apr 1 19:36:01 666.66.666.66:1093 -> 111.11.11.49:21 SYN
Apr 1 19:36:01 666.66.666.66:1094 -> 111.11.11.49:23 SYN
Apr 1 19:36:01 666.66.666.66:1095 -> 111.11.11.49:25 SYN
Apr 1 19:36:02 666.66.666.66:1096 -> 111.11.11.49:79 SYN
Apr 1 19:36:02 666.66.666.66:1097 -> 111.11.11.49:80 SYN
```

Cabe recordar que, hasta la fecha, el plugin de *snort* que le proporciona la característica de detección de análisis de puertos, no detecta análisis distribuidos, análisis lentos, ni análisis con paquetes fragmentados.

En comparación con *snort*, resulta un tanto lioso interpretar la salida de *tcpdump*. El significado de los campos para un paquete TCP/IP es el siguiente:

```
timestamp origen.puerto > destino.puerto : señales TCP números de
secuencia (bytes de datos) tamaño de ventana <opciones TCP> (señal de
no fragmentación) (tiempo de vida, identificador IP)
```

Ejemplo de salida de *tcpdump* para un TCP connect() scan al puerto 21 (abierto) y 37 (cerrado):

```
11:56:20.442740 connect.scanner.net.1141 > victim.cablemodem.com.21: S 929641:929641(0) win 8192 (DF)
11:56:21.191786 victim.cablemodem.com.21 > connect.scanner.net.1141: S 779881634:779881634(0) ack 929642 win 8576 (DF)
11:56:21.201490 connect.scanner.net.1141 > victim.cablemodem.com.21: . ack 1 win 8576 (DF)

11:56:23.954930 connect.scanner.net.1144 > victim.cablemodem.com.37: S 932103:932103(0) win 8192 (DF)
11:56:24.647238 victim.cablemodem.com.37 > connect.scanner.net.1144: R 0:0(0) ack 1 win 0
```

Ejemplo de salida de *tcpdump* para un TCP SYN scan al puerto 21 (abierto) y 37 (cerrado):

```
10:22:45.030552 half.scanner.net.49724 > victim.cablemodem.com.21: S 2421827136:2421827136(0)
10:22:45.030552 victim.cablemodem.com.21 > half.scanner.net.49724: S 4046313668:4046313668(0) ack 2421827137
10:22:45.030552 half.scanner.net.49724 > victim.cablemodem.com.21: R 2421827137:2421827137(0)

10:22:45.050552 half.scanner.net.49724 > victim.cablemodem.com.37: S 2418821749:2418821749(0)
10:22:45.050552 victim.cablemodem.com.37 > half.scanner.net.49724: R 0:0(0) ack 2418821750
```

Anexo B: resumen de técnicas para el análisis activo de sistemas operativos

- **FIN probe.** Consiste en enviar a un puerto abierto un paquete FIN o cualquier paquete sin el bit de ACK o SYN activado, y esperar la respuesta. El comportamiento correcto es no responder, pero muchas implementaciones incorrectas (MS Windows, BSDI, CISCO, HP/UX, MVS e IRIX) envían como respuesta un RST.

```
                                +-- (correcto)
Origen -- (FIN)-> Puerto abierto destino --+
                                +-- (RST incorrecto)-> Origen
```

- **BOGUS flag probe.** Consiste en activar un bit TCP no definido (64 o 128) en la cabecera TCP de un paquete SYN. Las versiones de Linux anteriores a la 2.0.35 devuelven el bit activado en su respuesta. Algunos otros sistemas operativos reinician la conexión cuando reciben un paquete SYN+BOGUS.
- **TCP ISN Sampling.** Consiste en encontrar patrones en los números de secuencia iniciales elegidos por la implementación de TCP cuando se responde a una solicitud de conexión. Existen varios grupos: 64K (varias versiones antiguas de Unix), incrementos aleatorios (versiones nuevas de Solaris, IRIX, FreeBSD, Digital UNIX, Cray y otros), aleatorio verdadero (Linux 2.0.*, OpenVMS, versiones nuevas de AIX, etc.), dependiente del tiempo (Windows y unos pocos otros), constante (algunos hubs 3Com y impresoras Apple LaserWriter). A su vez se pueden clasificar grupos como el de incrementos aleatorios calculando varianzas, máximos comunes divisores y otras funciones sobre el conjunto de números de secuencia y las diferencias entre dichos números.
- **Don't Fragment bit.** Algunos sistemas operativos activan el bit IP de no fragmentación en algunos de los paquetes que envían, para conseguir varios beneficios de rendimiento. Ya que no todos los sistemas operativos lo hacen y algunos lo hacen en determinados casos, prestar atención a este bit proporciona información sobre el sistema operativo.
- **TCP Initial Window.** Consiste en comprobar el tamaño de ventana en los paquetes devueltos. Esta prueba proporciona información valiosa, ya que algunos sistemas operativos (por ej. AIX) ya pueden ser identificados únicamente por este campo.
- **ACK Value.** Aunque parece que debiera ser completamente estándar, en algunos casos las implementaciones difieren del valor que utilizan para el campo ACK. Por ejemplo, si se envía FIN|PSH|URG a un puerto cerrado la mayoría de implementaciones activarán ACK para que sea el mismo número de secuencia inicial, aunque Windows y algunas impresoras de red responderán con el mismo número de secuencia inicial más uno. Otro ejemplo, si se envía SYN|FIN|URG|PSH a un puerto abierto el sistema operativo Windows se comporta de manera inconsistente, respondiendo a veces el mismo número de secuencia, respondiendo otras veces con el mismo número de secuencia incrementado en uno y respondiendo otras veces con un número de secuencia aparentemente aleatorio.
- **ICMP Error Message Quenching.** Algunos sistemas operativos siguen la sugerencia de limitar la tasa a la que son enviados varios mensajes de error. Por ejemplo, el núcleo de Linux (en net/ipv4/icmp.h) limita la generación de mensajes de destino inalcanzable a 80 cada 4 segundos, con una penalización de ¼ de segundo si dicha tasa se excede. Una manera de probar esto es enviar un grupo de paquetes a algún puerto UDP alto y aleatorio, y contar el número de mensajes de puerto inalcanzable recibidos.
- **ICMP Message Quoting.** La especificación de los mensajes de error ICMP indica que éstos hacen referencia a una pequeña cantidad de un mensaje ICMP que causa diversos errores. En un mensaje de destino inalcanzable la mayoría de implementaciones retornan la cabecera IP requerida y 8 bytes. Sin embargo Solaris retorna un bit más, y Linux todavía más. Ello nos

permite reconocer a ordenadores con Linux y Solaris aunque no tengan ningún puerto abierto.

- **ICMP Error message echoing integrity.** En caso de error de puerto inalcanzable, el ordenador destino debe devolver parte del mensaje original enviado junto con el error. Algunos ordenadores utilizan la cabecera del mensaje enviado como “espacio de trabajo” durante el procesamiento del error y, por lo tanto, la cabecera está ligeramente modificada cuando se devuelve el mensaje. Devolver el campo IP de longitud aumentado, cambiar la dirección IP de origen, devolver sumas de verificación TCP o UDP inconsistentes, etc., son cambios que realizan algunos sistemas operativos y que permiten identificarlos.
- **Type of Service.** En el mensaje ICMP de puerto inalcanzable, la mayoría de sistemas operativos devuelven el campo TOS con valor 0, aunque Linux devuelve el valor hexadecimal 0xC0. En el mensaje ICMP de solicitud de eco con el último bit del campo TOS activado a 1, la mayoría de sistemas operativos devuelven el último bit del campo TOS con valor 0, aunque Windows 2000 y Ultrix devuelven dicho bit con el mismo valor 1.
- **Fragmentation Handling.** Consiste en que a menudo diferentes implementaciones manejan de manera diferente fragmentos IP superpuestos. Algunos sobrescriben las porciones viejas con las nuevas, y otros sobrescriben las porciones nuevas con las viejas. Existen diferentes pruebas para determinar como se reensamblan dichos fragmentos.
- **TCP Options.** Las opciones de los mensajes TCP (escala de ventana, no operación, tamaño máximo de segmento, marca de tiempo, fin de opciones, etc.) son una gran fuente de información a la hora de identificar sistemas operativos. Estas opciones son generalmente opcionales, así que no todos los sistemas las implementan. Se puede saber si una máquina las implementa enviándole un mensaje con una opción activada. Si la máquina destino soporta la opción generalmente lo hará saber activando dicha opción en la respuesta. Dentro de un mismo mensaje, se pueden activar varias opciones para probarlas todas simultáneamente.

En el caso de que varios sistemas operativos soporten el mismo conjunto de opciones, a veces se les puede distinguir por los valores de dichas opciones. Incluso en el caso de que soporten el mismo conjunto de opciones y devuelvan los mismos valores, a veces se les puede distinguir por el orden en que dichas opciones se devuelven y donde se aplica el relleno.

- **Exploit Chronology.** Incluso con todos los tests comentados anteriormente, a veces no se puede limitar suficiente como para distinguir el sistema operativo. Por ejemplo, Windows 95, Windows 98 y Windows NT 4.0 utilizan el mismo sistema de pila TCP. Una solución a este problema puede ser probar diferentes vulnerabilidades en el orden cronológico en que fueron apareciendo, para comprobar que versión del sistema operativo y parches tiene instalados la máquina. En el ejemplo anterior, podemos probar consecutivamente con “Ping of Death”, “Winnuke”, “Teardrop”, “Land”, y tras cada ataque comprobar con un *ping* si la máquina se ha colgado o no.
- **SYN Flood Resistance.** Algunos sistemas operativos dejan de aceptar nuevas conexiones si les envían numerosos paquetes SYN falsos. Unos sistemas operativos pueden manipular tan solo 8 paquetes, mientras que versiones recientes de Linux y otros sistemas operativos permiten varios métodos para impedir que la recepción de estos paquetes sea un serio problema. Así, se puede obtener información sobre el sistema operativo enviando 8 paquetes con origen falso a un puerto abierto y comprobando si después se puede establecer conexión a dicho puerto.

Anexo C: tablas para el análisis pasivo

Siphon*

<u>Window</u>	<u>TTL</u>	<u>DF</u>	<u>Operating System</u>
7D78	64	1	Linux 2.1.122 - 2.2.14
77C4	64	1	Linux 2.1.122 - 2.2.14
7BF0	64	1	Linux 2.1.122 - 2.2.14
7BC0	64	1	Linux 2.1.122 - 2.2.14
832C	64	1	Linux 2.0.34 - 2.0.38
7FE0	64	0	Linux 2.0.34 - 2.0.38
0B68	64	1	Linux 2.0.32 - 2.0.34
4470	64	0	FreeBSD 2.2.1 - 4.0
4470	64	1	FreeBSD 2.2.1 - 4.0
43E0	64	1	FreeBSD 2.2.1 - 4.0
4074	64	0	OpenBSD 2.x
43E0	64	0	OpenBSD 2.x
4000	64	0	FreeBSD 2.2.1 - 4.0
4000	64	1	FreeBSD 2.2.1 - 4.0
2238	64	1	BSDI BSD/OS
2180	64	1	BSDI BSD/OS
2220	64	1	BSDI BSD/OS
2000	64	1	BSDI BSD/OS
81D0	64	1	Compaq Tru64 UNIX 5.0
ED90	64	1	IRIX 6.2 - 6.5
EE48	64	1	IRIX 5.1 - 5.3
EF88	64	1	IRIX 6.2 - 6.5
C000	64	1	IRIX 6.2 - 6.5
1800	64	1	VMS MultiNet V4.2(16) / OpenVMS V7.1-2
3EBC	64	1	AIX 4.02.0001.0000 / AIX 4.2
1000	64	0	SCO UnixWare 2.1.2
60F4	64	1	SCO OpenServer 5.0.5
2238	32	1	Windows NT / Win9x
2190	32	1	Windows NT / Win9x
2180	32	1	Windows NT / Win9x
2238	128	0	Windows NT / Win9x
2010	128	1	Windows NT / Win9x
2058	128	1	Windows NT / Win9x
2000	128	1	Windows NT / Win9x
2180	128	1	Windows NT / Win9x
2190	128	1	Windows NT / Win9x
2220	128	1	Windows NT / Win9x
2238	128	0	Windows NT / Win9x
2238	128	1	Windows NT / Win9x
21D2	128	1	Windows NT / Win9x
4470	128	1	Windows 2000 RC1
2328	255	1	Solaris 2.6 - 2.7
2238	255	1	Solaris 2.6 - 2.7
2400	255	1	Solaris 2.6 - 2.7
2798	255	1	Solaris 2.6 - 2.7
FE88	255	1	Solaris 2.6 - 2.7
87C0	255	1	Solaris 2.6 - 2.7
FAF0	255	0	Solaris 2.6 - 2.7
FAF0	255	1	Solaris 2.6 - 2.7
FFFF	255	1	Solaris 2.6 - 2.7
1020	255	1	Cisco IOS
4150	64	1	Cisco Localdirector 430 / running OS 2.1
200	0	1	Ascend Pipeline 50 ISDN Router

Leyenda:

Window = *window size* - tamaño de ventana

TTL = *time to live* - tiempo de vida

DF = *don't fragment flag* - bit de no fragmentación (0 = no activado, 1 = activado)

Pof*

Window	TTL	Segment	DF	WS	SF	NOP	PS	Operating System
32694	255	536	0	0	0	0	-1	3Com HiPer ARC, System V4.2.32
16384	64	512	0	0	0	0	44	AIX 3.2, 4.2 - 4.3
16384	64	1460	0	-1	0	0	44	AIX 4.3 - 4.3.3
65535	64	1460	0	1	0	1	48	AOL proxy, Compaq Tru64 UNIX V5.1 (Rev. 732)
8192	64	1460	1	-1	0	0	44	AXCENT Raptor Firewall Windows NT 4.0/SP3
4096	32	1024	0	245	0	0	-1	Alcatel (Xylan) OmniStack 5024
4096	32	1024	0	245	0	0	-1	Alcatel (Xylan) OmniStack 5024 v3.4.5
32696	64	536	1	0	1	1	60	Anonymizer.com proxy (Unixware?)
8192	64	1460	1	0	1	1	60	BSDI BSD/OS 3.0 - 4.0 (or MacOS, NetBSD)
8192	64	1460	1	0	0	1	60	BSDI BSD/OS 3.1 - 4.0
12288	255	1460	0	-1	0	0	44	BeOS 5.0 (1)
12288	255	1460	0	-1	0	1	44	BeOS 5.0 (2)
65535	128	1368	1	-1	0	0	44	BorderManager 3.0 - 3.5
8192	64	1460	0	-1	0	0	44	CacheFlow 500x CacheOS 2.1.08 - 2.2.1
65535	64	1460	0	0	0	1	60	CacheOS 3.1 on a CacheFlow 6000
4096	32	1024	0	-1	0	0	-1	Chorus MiX V.3.2 r4.1.5 COMP-386
2144	255	536	0	-1	0	0	-1	Cisco IGS 3000 IOS 11.x(16), 2500 IOS 11.2(3)P
4288	255	536	0	-1	0	0	-1	Cisco IOS 1600/11.2(15)P, 2500/11.2(5)P, 4500/11.1(7)
4128	255	556	0	0	0	0	-1	Cisco IOS 1750/12.0(5), 2500/11.3(1), 3600/12.0(7)
4128	255	1460	0	-1	0	0	-1	Cisco IOS 2611/11.3(2)XA4, C2600/12.0(5)T1, 4500/12.0(9), 3640/12.1(2), 3620/12.0(8) or 11.3(11a)
4288	255	1460	0	-1	0	0	-1	Cisco IOS 3620/11.2(17)P
65535	64	1432	0	-1	0	0	44	Cisco webcache
32768	128	1460	1	0	0	1	48	Dec V4.0 OSF1
32768	64	1460	1	0	0	1	48	Digital UNIX V4.0E
5535	64	1460	1	0	0	1	60	FreeBSD 2.2.1 - 4.1
16384	255	1460	1	0	0	1	48	FreeBSD 2.2.6-RELEASE
16384	64	1460	1	0	0	1	44	FreeBSD 2.2.8-RELEASE
16384	64	1460	1	0	0	0	44	FreeBSD 4.0-STABLE, 3.2-RELEASE
16384	64	1460	1	94	0	1	44	FreeBSD 4.0-STABLE, 3.2-RELEASE (2)
16384	64	1460	1	98	0	0	44	FreeBSD 4.0-STABLE, 3.2-RELEASE (3)
16384	64	1460	1	112	0	0	44	FreeBSD 4.0-STABLE, 3.2-RELEASE (4)
16384	64	1460	1	0	0	1	68	FreeBSD 4.3 - 4.4PRERELEASE
32768	64	1460	1	0	0	0	44	HP-UX B.10.01 A 9000/712
61440	64	512	0	-1	0	0	-1	IRIX 5.3 / 4.0.5F
61440	64	1460	0	-1	0	0	44	IRIX 6.3
49152	64	1460	0	0	0	0	44	IRIX 6.5 / 6.4
61440	64	1460	0	-1	1	1	48	IRIX 6.5.10
32320	64	1616	1	0	1	1	60	Linux (unknown?) (1)
5840	64	1460	0	0	1	1	60	Linux (unknown?) (2)
32120	64	1460	0	-1	0	0	-1	Linux 2.0.33 (1)
512	64	1460	0	52	0	0	44	Linux 2.0.33 (2)
512	64	1460	0	-1	0	0	44	Linux 2.0.34-38
512	64	0	0	-1	0	0	-1	Linux 2.0.35 - 2.0.37
512	64	1460	0	0	0	0	44	Linux 2.0.35 - 2.0.38
32120	64	1460	0	-1	0	0	44	Linux 2.0.38 (2)
64240	255	1460	1	-1	0	0	44	Linux 2.1.xx (?)
31944	64	1412	1	0	1	1	60	Linux 2.2
31072	64	3884	1	0	1	1	-1	Linux 2.2.12-20 (RH 6.1)
32120	32	1460	1	0	1	1	60	Linux 2.2.13 (1)
8192	128	1456	1	0	1	1	64	Linux 2.2.13 (2)
32120	64	1460	1	100	1	1	60	Linux 2.2.14
32120	64	1460	1	101	1	1	60	Linux 2.2.15
32120	64	1460	1	190	1	1	60	Linux 2.2.16
32120	64	1460	0	0	1	1	60	Linux 2.2.19
32120	64	1460	1	0	1	1	60	Linux 2.2.9 - 2.2.18
32120	64	1460	1	9	1	1	60	Linux 2.2.x
16060	64	1460	1	0	1	1	60	Linux 2.2.x Debian/Caldera (check)

Leyenda:

Window = *window size* - tamaño de ventana
TTL = *time to live* - tiempo de vida
Segment = *maximum segment size* - tamaño máximo de segmento
DF = *don't fragment flag* - bit de no fragmentación (0 = no activado, 1 = activado)
WS = *window scaling* - escala de ventana (-1 = no presente, otro = valor)
SF = *sackOK flag* - bit sackOK (0 = no activado, 1 = activado)
NOP = *nop flag* - bit NOP (0 = no activado, 1 = activado)
PS = *packet size* - tamaño de paquete (-1 = irrelevante)

63903	128	0	0	-1	0	0	40	Linux 2.2.x or 2.4.x
31856	64	1460	1	0	1	1	60	Linux 2.3.99-ac - 2.4.0-test1
24820	64	1460	1	0	0	1	60	Linux 2.4.0
5840	64	1460	1	0	1	1	52	Linux 2.4.1-14 (1)
5840	64	1460	1	0	1	1	48	Linux 2.4.1-14 (2)
5808	64	1452	1	0	1	1	60	Linux 2.4.10 (1)
5808	64	1452	1	111	1	1	60	Linux 2.4.10 (2)
5840	64	1460	1	0	0	1	60	Linux 2.4.13-ac7
5840	64	1460	1	0	1	1	60	Linux 2.4.2 - 2.4.14 (1)
16384	64	1460	1	0	0	1	60	Linux 2.4.2 - 2.4.14 (2)
5840	64	1460	1	223	1	1	60	Linux-2.4.13-ac7
16616	255	1460	1	0	0	0	48	Mac OS 7.x-9.x
32768	255	1460	1	0	0	1	48	Mac OS 9 (1)
65535	255	1460	1	1	0	1	48	Mac OS 9 (2)
32768	64	1460	1	0	0	1	48	Mac OS X
32768	64	1460	1	0	0	1	60	Mac OS X 10.1
32768	64	1460	1	122	0	1	60	Mac OS X 10.1 (2)
1024	64	0	0	-1	0	0	40	NMAP scan (distance inaccurate) (1)
4096	64	0	0	-1	0	0	40	NMAP scan (distance inaccurate) (10)
4096	64	265	0	10	0	1	60	NMAP scan (distance inaccurate) (11)
4096	64	536	0	-1	0	0	40	NMAP scan (distance inaccurate) (12)
1024	64	265	0	10	0	1	60	NMAP scan (distance inaccurate) (2)
1024	64	536	0	-1	0	0	40	NMAP scan (distance inaccurate) (3)
3072	64	0	0	-1	0	0	40	NMAP scan (distance inaccurate) (4)
3072	64	265	0	10	0	1	60	NMAP scan (distance inaccurate) (5)
3072	64	536	0	-1	0	0	40	NMAP scan (distance inaccurate) (6)
2048	64	0	0	-1	0	0	40	NMAP scan (distance inaccurate) (7)
2048	64	265	0	10	0	1	60	NMAP scan (distance inaccurate) (8)
2048	64	536	0	-1	0	0	40	NMAP scan (distance inaccurate) (9)
16384	64	1460	0	0	0	1	60	NetBSD 1.3/i386
32768	128	1460	1	-1	0	0	-1	Novell NetWare 4.11
16384	64	512	0	0	0	1	60	OpenBSD 2.6-2.8
16384	64	572	1	0	1	1	64	OpenBSD 3.0
32768	64	1432	0	0	0	0	44	PlusGSM, InterNetia proxy ???
4660	255	0	0	-1	0	0	40	Queso 1.2 (OS unknown, Linux, Solaris, *BSD)
24820	64	1460	1	0	0	0	44	SCO UnixWare 7.0.1
32696	64	536	0	0	1	1	60	SCO UnixWare 7.1.0 x86 (1)
24820	64	1460	1	0	0	1	60	SCO UnixWare 7.1.0 x86 (2)
24820	64	1460	1	0	0	1	48	SCO UnixWare 7.1.0 x86 ? (3)
8760	64	1460	1	0	0	0	-1	Solaris 2.6 (2)
9140	255	9140	1	0	0	0	-1	Solaris 2.6 (sunsite)
8760	255	1460	1	-1	1	0	44	Solaris 2.6 - 2.7
8760	255	1460	1	0	0	0	44	Solaris 2.6 or 2.7 (1)
8760	255	1460	1	-1	0	1	44	Solaris 2.6 or 2.7 (2)
8760	255	1380	1	0	0	0	44	Solaris 2.7
33580	255	1460	1	-1	0	0	44	Solaris 7
24820	64	1460	1	-1	1	1	48	SunOS 5.8
24820	64	1460	1	-1	1	1	-1	SunOS 5.8 Sparc
16384	64	0	0	-1	0	0	-1	ULTRIX V4.5 (Rev. 47)
8192	64	1460	1	0	1	1	64	WebTV netcache engine (BSDI)
16384	128	1460	1	0	1	1	48	Windows 2000 (1)
16384	128	1460	1	52	1	1	48	Windows 2000 (1)
16384	128	25275	1	-1	1	1	-1	Windows 2000 (2)
8760	128	1460	1	-1	1	1	48	Windows 2000 (2)
60352	64	1360	1	2	1	1	52	Windows 2000 (3)
16384	128	1452	1	-1	1	1	48	Windows 2000 (4)
16384	128	1360	1	-1	1	1	48	Windows 2000 (5)
16384	128	1414	1	-1	1	1	48	Windows 2000 (8)
16384	128	1460	1	-1	1	1	48	Windows 2000 (9)
8760	128	536	1	-1	1	1	48	Windows 2000 Pro (2128)
44032	128	64059	1	-1	1	1	-1	Windows 2000 SP2 (1)
44032	128	1452	1	-1	1	1	48	Windows 2000 SP2 (2)
16384	128	55370	1	-1	1	1	48	Windows 2000 early
5840	128	536	1	0	1	1	48	Windows 95 (3)
8192	32	1456	1	-1	0	0	44	Windows 95 (4)
8192	128	1460	0	-1	1	1	48	Windows 95 (?) (6)
5840	128	1460	1	-1	1	1	48	Windows 95 or early NT4
16384	64	1460	0	-1	0	0	44	Windows 98
65535	128	1372	1	-1	1	1	48	Windows 98 (2)
8192	128	1460	1	52	1	1	48	Windows 98 (3)
65535	128	1460	1	-1	1	1	48	Windows 98 (4)
15972	64	1452	1	0	1	1	60	Windows 98 (?)
65535	128	1368	1	-1	0	0	44	Windows 98, Windows NT 5.0
24820	64	1460	1	0	0	0	44	Windows 9x
8192	128	1460	1	0	1	1	48	Windows 9x (1)
2144	64	536	1	0	1	1	48	Windows 9x (10)
8192	128	536	1	0	1	1	48	Windows 9x (2)

2144	64	536	1	0	1	1	60	Windows 9x (4)
8192	128	1460	1	0	1	1	64	Windows 9x (5)
8192	128	1460	1	0	1	1	44	Windows 9x (6)
8192	128	1360	1	-1	1	1	48	Windows 9x (9)
8192	128	536	1	-1	1	1	48	Windows 9x or 2000
8192	128	1414	1	-1	1	1	48	Windows 9x or NT4
32768	32	1460	1	-1	0	0	44	Windows CE 3.0 (Ipaq 3670) (1)
32768	32	1460	1	-1	0	1	44	Windows CE 3.0 (Ipaq 3670) (2)
16384	128	1460	1	75	1	1	48	Windows ME
8192	128	1460	1	-1	1	0	44	Windows NT
8192	128	1460	1	0	0	0	44	Windows NT 4.0 (1)
8192	32	1460	1	0	0	0	44	Windows NT 4.0 (2)
4288	128	1460	1	-1	1	1	48	Windows NT 4.0 SP3 (1)
8192	128	1456	1	-1	1	1	48	Windows NT 4.0 SP3 (2)
16384	128	1272	1	-1	1	1	48	Windows NT 4.0 SP3 (3)
16384	128	572	1	-1	1	1	48	Windows NT 4.0 SP4+
8192	128	25443	1	-1	1	1	-1	Windows NT 4.0 Server SP5
16384	128	1440	1	-1	1	1	48	Windows XP Pro
64240	128	1460	1	-1	1	1	48	Windows XP Pro, Windows 2000 Pro

Anexo D: TCP, UDP, ICMP e IP

Transmission Control Protocol

El propósito de TCP es proporcionar un servicio de reparto seguro orientado a conexión. TCP interpreta los datos como flujo de bytes, no como tramas, y su unidad de transferencia se denomina segmento. Los segmentos se utilizan para establecer conexiones, así como para transportar datos y acuses de recibo. TCP cuida de asegurar la fiabilidad, control del flujo y mantenimiento de la conexión, recuperando los datos que están dañados, perdidos, duplicados o intencionadamente fuera de secuencia. Para lograr su objetivo, TCP añade una cabecera de 20 bytes a inicio de cada datagrama:

0	4	10	15 16	24	31
PUERTO TCP DE ORIGEN			PUERTO TCP DE DESTINO		
NÚMERO DE SECUENCIA					
NÚMERO DE ACUSE DE RECIBO					
DESPLAZ.	RESERVADO	SEÑALES CÓDIGO		TAMAÑO DE VENTANA	
SUMA DE VERIFICACIÓN TCP			PUNTERO DE URGENCIA		
OPCIONES TCP (SI LAS HAY)				RELLENO	
DATOS					
...					

Formato de los campos en un segmento TCP con un encabezado TCP seguido de datos.

Señales de código	Significado si el bit está puesto a 1
10 URG	El campo de puntero urgente es válido
11 ACK	El campo de acuse de recibo es válido
12 PSH	El receptor no pondrá en cola los datos, sino que los pasará a la aplicación
13 RST	Destruir la conexión
14 SYN	Iniciar la conexión - Sincronizar los números de secuencia
15 FIN	Finalizar la conexión - El emisor ha llegado al final de su flujo de octetos

Bits del campo código en el encabezado TCP.

Los números de puerto se utilizan para el seguimiento de diferentes conversaciones.

El número de secuencia se utiliza para que el extremo que recibe los datagramas se asegure de colocarlos en el orden correcto y de no haber extraviado ninguno. TCP asigna un número de secuencia a cada byte transmitido, no a cada datagrama. Así, si hay 500 bytes de datos en cada datagrama, el primer datagrama será numerado 0, el segundo 500, el siguiente 1000, etc. El ordenador que recibe los datos debe devolver un segmento con el bit ACK activado en el campo de código para confirmar que recibió la información. Si no lo hace antes de un cierto periodo de tiempo, se retransmiten los datos.

El número de acuse de recibo guarda el valor del siguiente número de secuencia esperado y confirma que se han recibido todos los datos a través del número de acuse de recibo menos uno.

Los bits del desplazamiento de datos, multiplicados por cuatro, indican la longitud en bytes de la cabecera TCP, que puede variar según se añadan más o menos opciones TCP. Algunas de estas opciones son: tamaño máximo de segmento, escala de ventana, marca de tiempo, no operación (NOP), acuse de recibo selectivo, acuse de recibo selectivo permitido (SackOK) y datos del acuse de recibo selectivo.

El tamaño de ventana se utiliza para controlar cuanta información puede estar en tránsito en un momento dado. No es práctico esperar a que cada datagrama enviado sea confirmado antes de transmitir el siguiente, cosa que ralentizaría bastante el proceso. Por otro lado, si se envía la información sin esperar la confirmación, el ordenador que envía la información puede sobrepasar la capacidad de absorber la información del ordenador que la recibe si éste último es más lento. Así, cada extremo indica en el campo tamaño de ventana cuantos bytes de datos nuevos está actualmente preparado para aceptar. A medida que un ordenador recibe datos, la cantidad de espacio que queda en su ventana decrementa hasta aproximarse a cero, momento en el cual el ordenador que envía los datos debe parar. Mientras el receptor procesa los datos, incrementa su tamaño de ventana indicando que está preparado para aceptar más datos. A menudo el mismo datagrama puede utilizarse para confirmar la recepción de datos y para dar permiso para transmitir nuevos datos incrementando el tamaño de ventana.

La suma de verificación es un número que se calcula, más o menos, sumando todos los bytes del datagrama. Al recibir los datos en el otro extremo, se calcula la suma de verificación de nuevo. Si la suma es errónea, no se envía el segmento ACK de confirmación y los datos son reenviados.

User Datagram Protocol

UDP es un protocolo no orientado a conexión que, al igual que TCP, utiliza IP para enviar datagramas, pero que a diferencia de TCP, no vigila que los paquetes lleguen a su destino. UDP se utiliza en aplicaciones donde no es esencial que lleguen el 100% de los paquetes (como el flujo de sonido o vídeo) o donde los mensajes caben en un solo datagrama y no es necesaria la complejidad de TCP (si no se obtiene respuesta pasados unos segundos, se vuelve a enviar). Recientemente muchas aplicaciones de Internet comienzan a utilizar a la vez UDP y TCP. TCP para enviar los datos más esenciales y de control, mientras que UDP para los datos cuyas pérdidas son aceptables.

La cabecera de un datagrama UDP es mucho más sencilla que la de un segmento TCP:

0	15 16	31
PUERTO UDP DE ORIGEN		PUERTO UDP DE DESTINO
LONGITUD DEL MENSAJE UDP		SUMA DE VERIFICACIÓN UDP
DATOS		
...		

Formato de los campos en un datagrama UDP.

Internet Control Message Protocol

ICMP es un protocolo alternativo utilizado para la transmisión de mensajes de error y otros mensajes relacionados con el software TCP/IP, más que con programas particulares del usuario. ICMP es un mecanismo de reporte de errores que proporciona una forma para que los enrutadores que encuentren un error lo envíen a la fuente original. Por ejemplo, si un ordenador intenta conectar con otro al que no encuentra, recibirá un mensaje ICMP diciendo "host unreachable". ICMP también puede utilizarse para obtener cierta información sobre la red.

0	7 8	15 16	31
TIPO	CÓDIGO	SUMA DE VERIFICACIÓN	
MAS CAMPOS Y DATOS DEPENDIENTES DEL TIPO DE MENSAJE			
...			

Formato de mensaje ICMP.

Campo de tipo	Tipo de mensaje ICMP
0	Respuesta de eco
3	Destino inalcanzable
4	Origen acallado
5	Redireccionar (cambiar una ruta)
8	Solicitud de eco
11	Tiempo excedido para un datagrama
12	Problema de parámetros en un datagrama
13	Solicitud de marca de tiempo
14	Respuesta de marca de tiempo
15	Solicitud de información (obsoleto)
16	Respuesta de información (obsoleto)
17	Solicitud de máscara de dirección
18	Respuesta de máscara de dirección

Campo de tipo en un mensaje ICMP.

Aunque cada mensaje ICMP tiene su propio formato, todos comienzan con un campo de tipo de mensaje que identifica el mensaje, un campo código de mensaje que proporciona más información sobre el tipo del mensaje y un campo de suma de verificación.

Internet Protocol

El trabajo de IP es encontrar una ruta para los datagramas TCP, UDP o ICMP y llevarlos a su destino. IP añade una cabecera propia a dichos datagramas para permitir a las puertas de enlace y sistemas intermedios reenviar el datagrama.

0	4	8	15	16	19	24	31
VERSION		LONG. CAB.	TIPO DE SERVICIO		LONGITUD TOTAL		
IDENTIFICACIÓN				BANDERAS	DESPLAZAMIENTO DE FRAGMENTO		
TIEMPO DE VIDA		PROTOCOLO		SUMA DE VERIFICACIÓN DE LA CABECERA			
DIRECCIÓN IP DE ORIGEN							
DIRECCIÓN IP DE DESTINO							
OPCIONES IP (SI LAS HAY)						RELLENO	
DATOS							
...							

Formato de un datagrama IP, la unidad básica de transferencia en Internet.

Los campos principales de la cabecera son: la dirección Internet del origen, necesaria para saber de donde viene el datagrama; la dirección Internet del destino, necesaria para que las puertas de enlace intermedias sepan hacia donde deben dirigir el datagrama; el número de protocolo, que indica cual es el protocolo del datagrama contenido dentro del datagrama IP (no sólo TCP utiliza IP, hay más protocolos que también lo utilizan); y una suma de verificación de la cabecera, que permite comprobar si ésta se dañó durante el transporte.

Las direcciones de Internet son campos de 32 bits divididos en cuatro subcampos de 8 bits, que contienen valores como por ejemplo 147.83.170.211, aunque actualmente se está trabajando en unas nuevas direcciones IP de 128 bits (IPv6).

El número de identificación, las banderas y el desplazamiento de fragmento se utilizan para el seguimiento de las partes cuando un datagrama se deba partir, por ejemplo, porque los datagramas se reenvían por una red para la cual son demasiado grandes.

El tiempo de vida es un número que se decrementa cada vez que el datagrama pasa a través de un sistema. Cuando llega a cero, el datagrama se destruye. Esto es útil si de alguna manera se originara un bucle en el sistema (caso teóricamente imposible).

Los primeros cuatro bits de la cabecera (versión) indican el formato de la cabecera IP. Los siguientes cuatro bits (longitud de cabecera), multiplicados por cuatro, indican la longitud en bytes de la cabecera, que es variable debido a que no es obligatorio que aparezcan todas las opciones IP. Dichas opciones son: no operación, seguridad, ruta de origen desconectada, ruta de origen estricta, registro de ruta, identificador de flujo y marcas de tiempo.

El tipo de servicio se utiliza para la priorización de datagramas IP. Los tres primeros bits (campo de precedencia) indican el nivel de prioridad del datagrama. Existen ocho niveles de prioridad y los datagramas con mayor prioridad se envían antes que los menos prioritarios. Los siguientes cuatro bits (campo de tipo de servicio) indican como debe la red equilibrar entre espera, rendimiento, fiabilidad y coste en el momento de encaminar el datagrama IP. El último bit (campo MBZ) no se utiliza y debe ser cero.

Para aprender más sobre los protocolos TCP, UDP, ICMP e IP de Internet, recomiendo leer, respectivamente, los RFC 793, 768, 792 y 791, que se pueden encontrar en <http://www.faqs.org/rfcs/> y <http://www.rfc-editor.org/rfcxx00.html>.

Anexo E: algunos puertos “famosos”

Lista de los puertos más utilizados*

Puerto	Protocolo	Palabra clave	Descripción
0	TCP / UDP		Reservado
1	TCP	TCPMUX	Multiplexador de servicios TCP
5	TCP	RJE	Entrada de trabajo remoto
7	TCP / UDP	ECHO	Eco
9	TCP / UDP	DISCARD	Descartar (el /dev/null de Internet)
11	TCP	SYSTAT	Usuarios activos
13	TCP / UDP	DAYTIME	Hora del día (en formato humano)
15	TCP	- (antes NETSTAT)	No asignado (antes, programa de estado de red)
17	TCP / UDP	QUOTE	Cita del día
19	TCP / UDP	CHARGEN	Generador de caracteres
20	TCP	FTP-DATA	Protocolo de transferencia de archivos (datos)
21	TCP	FTP	Protocolo de transferencia de archivos (control)
22	TCP	SSH	Conexión de terminal segura
23	TCP	TELNET	Conexión de terminal
25	TCP	SMTP	Protocolo de transporte de correo sencillo
37	TCP / UDP	TIME	Hora del día (en formato máquina)
42	TCP / UDP	NAMESERVER	Servidor de nombres de anfitriones
43	TCP	NICNAME	¿Quién está ahí? (“whois”)
53	TCP / UDP	DOMAIN	Servidor de nombres de dominios (“dns”)
67	UDP	BOOTPS	Servidor de protocolo bootstrap
68	UDP	BOOTPC	Cliente de protocolo bootstrap
69	UDP	TFTP	Transferencia trivial de archivos
70	TCP	GOPHER	Gopher
77	TCP	-	Cualquier servicio RJE privado
79	TCP	FINGER	Finger
80 i 80xx	TCP	WWW-HTTP	World Wide Web HTTP
88	UDP	KERBEROS	Protocolo de autenticación Kerberos
101	TCP	HOSTNAME	Servidor de nombre de anfitrión NIC
110	TCP	POP3	Protocolo de oficina postal v. 3
111	TCP / UDP	RPC / PORTMAP	Llamada a procedimiento remoto
113	TCP	AUTH	Servicio de autenticación
117	TCP	UUCP-PATH	Servicio de trayecto UUCP
119	TCP	Nntp	Protocolo de transferencia de noticias de red
129	TCP	PWDGEN	Protocolo generador de clave de acceso
137	TCP / UDP	NETBIOS-NS	Servicio de nombre NETBIOS
138	TCP / UDP	NETBIOS-DGM	Servicio de datagrama NETBIOS
139	TCP / UDP	NETBIOS-SSN	Servicio de sesión NETBIOS
143	TCP / UDP	IMAP	Protocolo de acceso a mensajes de Internet
161	UDP	SNMP	Monitor de red SNMP
162	UDP	SNMPTRAP	Interrupciones SNMP
389	TCP / UDP	LDAP	Protocolo de acceso ligero a directorios
443	TCP	HTTPS	HTTP seguro sobre TLS/SSL
512	UDP / TCP	BIFF / EXEC	Notific. correo / Ejecución remota de procesos
513	UDP / TCP	WHO / LOGIN	Quien está conectado / Conexión remota a telnet
514	UDP / TCP	SYSLOG / SHELL	Conexión de sistema / Línea de comandos remota
515	TCP	PRINTER	Cola de la impresora (“spooler”)
517 / 518	UDP	TALK / NTALK	Protocolo de conversación
520	UDP	ROUTE	Tablas de encaminamiento
1080	TCP	SOCKS	Socks
2049	UDP	NFS	Sistema de ficheros de red
6000 a 6xxx	TCP	X11	X-Windows
6667	TCP	IRC	Transmisor de charlas (“chat”)

* Encontrareis la lista de puertos completa en <http://www.iana.org/assignments/port-numbers>

Lista de puertos (por defecto) utilizados por troyanos*

2	Death	1011	Doly Trojan
21	Back Construction, Blade Runner, Doly Trojan, Fore, FTP Trojan, Invisible FTP, Larva, MBT, Motiv, Net Administrator, Senna Spy FTP Server, WebEx, WinCrash	1012	Doly Trojan
23	Tint Telnet Server, Truva Atl	1015	Doly Trojan 1.5
25	Ajan, Antigen, Email Password Sender, Gip, Haebu Coceda (Naebi), Happy 99, I Love You, Kaung2, Pro Mail Trojan, Shtrilitz, Stealth, Tapiras, Terminator, WinPC, WinSpy	1016	Doly Trojan 1.6
31	Agent 31, Hackers Paradise, Masters Paradise	1020	Vampire
41	Deep Throat	1024	NetSpy, Psyber Streaming Server
48	DRAT	1029	InCommand Access
50	DRAT	1033	NetSpy
59	DMSSetup	1042	Blah 1.1
79	Firehotcker	1045	Rasmin
80	Back End, Executer, Hooker, RingZero	1050	Mini Command 1.2 Access
99	Hidden Port 2.0	1080	WinHole
110	ProMail Trojan	1081	WinHole
113	Invisible Identd daemon, Kazimas	1082	WinHole
119	Happy99	1083	WinHole
121	Jammer Killah V	1090	Xtreme
123	Net Controllor	1095	RAT
133	Faranz, port 146 - Infector	1097	RAT
146 UDP	Infector	1098	RAT
170	A-Trojan	1099	BFevolution, RAT
421	TCP Wrappers	1170	Psyber Stream Server, Streaming Audio Trojan, Voice
456	Hackers Paradise	1200 UDP	NoBackO
531	Rasmin	1201 UDP	NoBackO
555	ini-Killer, NetAdmin, Phase Zero, Stealth Spy	1207	Softwar
666	Attack FTP, Back Construction, Cain & Able, NokNok, Satanz Backdoor, ServeU, Shadow Phyre	1212	Kaos
667	SniperNet	1225	Scarab
669	DP Trojan	1234	Ultors Trojan
692	GayOL	1243	BackDoor-G, SunSeven, SubSeven
777	Aimspy		Apocalypse
133	Faranz, port 146 - Infector	1245	VooDoo Doll
808	WinHole	1255	Scarab
911	Dark Shadow	1256	Project nEXT
999	Deep Throat , WinSatan	1269	Mavericks Matrix
1000	Der Spaehher 3	1313	NETrojan
1001	Der Spaehher 3, Doly Trojan, Silencer, WebEx	1338	Millenium Worm
1010	Doly Trojan 1.35	1349 UDP	BackOrifice DLL
		1492	FTP99CMP
		1509	Psyber Streaming Server
		1524	Trinoo
		1600	Shivka-Burka
		1777	Scarab
		1807	Spy Sender
		1981	Shockrave
		1966	Fake FTP
		1969	OpC BO
		1981	Shockrave
		1999	TransScout, Backdoor
		2000	Der Spaehher 3, TransScout, Insane
		2001	Der Spaehher 3, TransScout, Trojan

* Encontrareis varias listas actualizadas en http://www.sys-security.com/html/papers/trojan_list.html, <http://www.simovits.com/nyheter9902.html>, <http://www.robertgraham.com/pubs/firewall-seen.html>, <http://www.wittys.com/files/all-ip-numbers.txt>, o buscando "trojan port list" en <http://www.google.com>.

	Cow	5742	WinCrash
2002	TransScout	5882 UDP	Y3K RAT
2003	TransScout	5888	Y3K RAT
2004	TranScout	6000	The Thing
2005	TransScout	6006	The Thing
2023	Ripper Pro, PassRipper	6272	Secret Service
2080	WinHole	6400	The Thing
2115	Bugs	6666	TCPShell (*NIX Backdoor)
2140	Deep Throat , The invasor	6669	Vampyre
2155	illusion Mailer	6670	Deep Throat
2283	HVL Rat 5	6711	SubSeven
2300	Xplorer	6712	SubSeven
2565	Striker	6713	SebSeven
2583	WinCrash 2	6723	Mstream
2600	Digital Root Beer	6771	Deep Throat
2716	The Prayer 2	6776	BackDoor-G, SubSeven
2773	SubSeven	6838 UDP	Mstream
2801	Phineas Phucker	6883	DeltaSource
2989 UDP	Rat	6912	ShitHeep
3000	Remote Shutdown	6913	ShitHeep Danny
3024	WinCrash	6939	Indoctrination
3128	RingZero	6969	GeteCrasher, Priority, IRC 3
3129	Masters Paradise	6970	GeteCrasher
3150	Deep Throat , The invasor	7000	Remote Grab, Kazimas
3459	Eclipse 2000, Sanctuary	7001	Freak88
3700	Portal Of Doom	7215	SubSeven
3791	Eclypse, Totaleclipse 1.0	7300	NetMonitor
3801 UDP	Eclypse	7301	NetMonitor
4000	Psyber Streaming Server, Skydance	7302	NetMonitor
4092	WinCrash	7303	NetMonitor
4242	Virtual Hacking Machine	7304	NetMonitor
4321	BoBo, SchoolBus 1.0	7305	NetMonitor
4444	Prosiak, Swift remote	7306	NetMonitor
4567	File Nail	7307	NetMonitor
4590	ICQTrojan	7308	NetMonitor
5000	Bubbel, Back Door Setup, S ockets de Troie, Socket 23	7309	NetMonitor
5001	Back Door Setup, Socket de Troie	7424	Host Control
5010	Solo	7424 UDP	Host Control
5011	One Of The Last Trojans (OOTLT), OOTLT Cart	7789	Back Door Setup, ICKiller
5031	NetMetropolitan 1.0/1.04	7983	Mstream
5032	NetMetropolitan	8080	RingZero
5321	Filehotcker	8787	BO2K
5343	wCrat	8879	Hack Office Armageddon
5400	Blade Runner, Back Construction 1.2/1.5	8988	BacHack
5401	Blade Runner, Back Construction	8989	Rcon
5402	Blade Runner, Back Construciton	9000	Netministrator
5512	illusion Mailer	9325 UDP	Mstream
5521	illusion Mailer	9400	InCommand
5550	Xtcp, Xtcp2	9872	Portal Of Doom
5555	ServeMe	9873	Portal Of Doom
5556	Bo Facil	9874	Portal Of Doom
5557	Bo Facil	9875	Portal Of Doom
5569	RoboHack	9876	Cyber Attack, RUX
5637	Crasher	9876	TransScout
5638	Crasher	9878	TransScout
5714	WinCrash	9989	ini-Killer
5741	WinCrash	9999	The Prayer 1
		10067 UDP	Portal Of Doom
		10085	Syphillis
		10086	Syphillis
		10101	BrainSpy

10167	UDP	Portal Of Doom	29104	Host Control
10520		Acid Shivers	29891	UDP The Unexplained
10528		Host Control	30001	TerrOr32
10607		Coma	30029	AOL Trojan 1.1
10666		Ambush	30100	NetSphere
10752		LINUX mounts Backdoor	30101	NetSphere
11000		Senna Spy	30102	NetSphere
11050		Host Control	30103	NetSphere
11051		Host Control	30103	UDP NetSphere
11223		Progenic Trojan	30133	NetSphere Final 1.31.337, Trojan Spirit 2001a
12076		Gjamer	30303	Sockets de troie, Socket 23, Socket 25
12223		Hack 99 KeyLogger	30974	Intruse
12345		NetBus , GabanBus, X-Bill, Pie Bill Gates	30999	Kaung 2
12346		NetBus 1.0, GabanBus, X-Bill	31335	UDP Trinoo
12361		Whack-a-Mule	31336	BO Whack, ButtFunnel
12362		Whack-a-Mule	31337	Baron Night, BO Client , BO2, BO Facil
12623	UDP	DUN Control	31337	UDP Back fire, Back Orifice , Deep BO
12624		Buttman	31338	NetSpy DK, ButtFunnel
12631		Whack Job	31338	UDP Back Orifice , Deep BO
12701		Eclipse 2000	31399	NetSpy DK
12754		Mstream	31554	Schwindler
13000		Senna Spy	31666	BoWhack
13010		Hacker Brazil	31785	Hack a Tack
13700		Kuang 2 The Virus	31787	Hack a Tack
15092		Host Control	31788	Hack a Tack
15104		Mstream	31789	UDP Hack a Tack
16484		Mosucker	31791	UDP Hack a Tack
16660		Stracheldracht	31792	Hack a Tack
16772		ICQ Revenge	32100	Peanut Brittle, Project nEXT
16959		Subseven DEFCON8 2.1	32418	Acid Battery 1.0
16969		Priority, Portal Of Doom	33333	Blakharaz, Prosiak
17166		Mosaic	33577	PsychWard
17300		Kaung 2 The Virus	33777	PsychWard
17777		Nephron	33911	Spirit 2001a
18753	UDP	Shaft	34324	BigGluck, TN, Tiny Telnet Server
19864		ICQ Revenge	34555	UDP Trinoo - Windows
20000		Milennium	35555	UDP Trinoo - Windows
20001		Milennium	37651	Yet Another Trojan
20002		AcidkoR	40412	The Spy
20034		NetBus 2 Pro	40421	Masters Paradise, Agent 40421
20203		Logged!, Chupacabra	40422	Masters Paradise
20331		Bla	40423	Masters Paradise
20432		Shaft	40426	Masters Paradise
20432	UDP	Shaft	41666	Remote Boot
21544		Girl Friend, Kidterror, Schwindler 1.8, Schwindler 1.82	41666	UDP Remote Boot
22222		Prosiak	43210	SchoolBus 1.6/2.0
23023		Logged	44444	Prosiak
23432		Asylum	47262	UDP Delta Source
23456		Evil FTP, Ugly FTP, Whack Job	49301	Online KeyLogger
23476		Donald Duck	50505	Sockets de Troie
23476	UDP	Donald Duck	50766	Fore, Schwindler
23477		Donald Duck	50776	Fore, Remote Windows Shutdown
26274	UDP	Delta Source	51996	Cafeini
26681		Spy Voice	52317	Acid Battery 2000
27374		SubSeven 2.1	53001	Remote Windows Shutdown
27444	UDP	Trinoo	54283	SubSeven
27573		SubSeven	54320	Back Orifice 2000
27665		Trinoo		

54321 [Back Orifice 2000](#), SchoolBus
1.6/2.0
54321 UDP [Back Orifice](#)
57341 Netraider
58339 ButtFunnel
60000 [Deep Throat](#)
60068 Xzip 6000068
60411 Connection

61348 BunkerHill
61466 TeleCommando
61603 BunkerHill
63485 BunkerHill
65000 Devil 1.03, Stacheldracht
65432 The Traitor
65432 UDP The Traitor
65535 RC