Análisis Activo y Pasivo de Redes

Alejandro Castán Salinas

alex.castan@upc.es

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 aguanto 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://www.sindominio.net/cathack/

¡Nos vemos en el Hackmeeting Madrid 2001 (hackmad01)!

Análisis activo y pasivo de redes

Introducción

En la preparación de un ataque remoto a un sistema informático, el atacante necesita no sólo conocer la dirección en la red de la víctima, sino también obtener el máximo de 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. wuftpd 2.6.0 o wuftpd 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 enrutar 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 listos para transferir datos, tengan conocimiento de que el otro también lo está y acuerden un número de secuencia inicial. Escuetamente explicado, 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. El ordenador en el otro extremo recibe el segmento y, si puede iniciar la conexión (puerto de destino abierto = servicio disponible), responde a su vez con un segmento con los bits SYN y ACK activados. A este segmento de respuesta, 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.

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.

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 enruta 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.

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.
- 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://www.enteract.com/~lspitz/finger.html.

Software

- Una herramienta excelente para el análisis activo de puertos y sistemas operativos es *nmap*. La encontrareis en http://www.insecure.org/nmap/.

- Dos herramientas para el análisis pasivo de sistemas operativos son *siphon* y *p0f*. Las encontrareis respectivamente en http://www.subterrain.net/projects/siphon/ y http://lcamtuf.na.export.pl, o buscando en el repositorio http://packetstorm.securify.com.
- 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://ethereal.zing.org. 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 si existe algún sniffer instalado en vuestra red recomiendo *antisniff*, que encontrareis en http://www.l0pht.com/antisniff/, y *check promiscuous mode*, que encontrareis en ftp://info.cert.org/pub/tools/cpm.tar.Z.
- Existen varias herramientas IDS ("Intrusion Detection System") que permiten detectar y analizar escaneados de puertos. Aunque las más famosas son *shadow* i *aide*, mis preferidas són *snort*, que encontrareis en http://www.snort.org, e *iplog*, que encontrareis en http://ojnk.sourceforge.net.
- 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.

```
                            +--(RST puerto cerrado)-> Origen
Origen --(SYN)-> Destino --+
                            +--(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.

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

  Ventajas: (1) pocos sitios registran este intento de conexión.

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

- **TCP FIN scanning.** Ante un paquete FIN, los puertos cerrados deberían replicar con el debido RST y los puertos abiertos deberían ignoran el paquete FIN. En este caso podemos determinar que puertos están cerrados y, por exclusión, cuales están abiertos.

```
                            +--(RST 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.

- **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 pequeños paquetes.

  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.

- **TCP reverse ident scanning.** El protocolo IDENT 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) funciona aunque el proceso no inicie la conexión.

  Desventajas: (1) este análisis tan solo puede realizarse con una conexión TCP completa al puerto en cuestión.

- **FTP bounce attack.** 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. Dicha característica se puede aprovechar para analizar puertos TCP a través de un servidor de FTP "proxy".

  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. 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".

- **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.

  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) es lento y (2) 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.

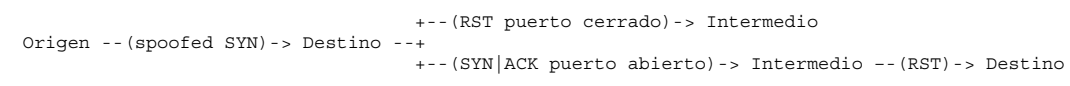
  Desventajas: (1) No funciona para todos los sistemas.

- **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 servidores de una red están activos realizando un *ping* sobre ellos.

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

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

- **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.



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

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

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 reinicializan 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) pueden ser identificados ya 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 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 1/4 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.
- **Fragmentation Handling.** Consiste en que a menudo diferentes implementaciones manejan de manera diferente fragmentos IP superpuestos. Algunos sobreescriben las porciones viejas con las nuevas, y otros sobreescriben las porciones nuevas con las viejas. Existen diferentes pruebas para determinar como se reensamblan dichos fragmentos.
- **TCP Options.** Las opciones de los mensajes TCP (Window Scale, NOP, Max Segment Size, Timestamp, End of Ops, 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 la 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*

Window	TTL	DF	Operating System
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)

P0f*

Window	TTL	Segment	DF	WS	SF	NOP	Operating System
32694	255	536	0	0	0	0	3Com HiPer ARC, System V4.2.32
32768	64	1432	0	0	0	0	??? (PlusGSM, InterNetia proxy)
16384	64	512	0	0	0	0	AIX 3.2, 4.2 - 4.3
8192	64	1460	1	-1	0	0	AXCENT Raptor Firewall Windows NT 4.0/SP3
4096	32	1024	0	245	0	0	Alcatel (Xylan) OmniStack 5024
4096	30	1024	0	245	0	0	Alcatel (Xylan) OmniStack 5024 v3.4.5
8192	64	1460	1	0	0	1	BSDI BSD/OS 3.1
65535	128	1368	1	-1	0	0	BorderManager 3.5
4096	30	1024	0	-1	0	0	Chorus MiX V.3.2 r4.1.5 COMP-386
4288	255	536	0	-1	0	0	Cisco 1600 IOS 11.2(15)P
4128	255	556	0	0	0	0	Cisco 1750 IOS 12.0(5), Cisco 2500 IOS 11.3(1)
4288	255	536	0	-1	0	0	Cisco 2500 IOS 11.2(5)P, Cisco 4500 IOS 11.1(7)
4128	255	1460	0	-1	0	0	Cisco 2611 IOS 11.3(2)XA4
4128	255	556	0	0	0	0	Cisco 3600 IOS Versión 12.0(7)
4288	255	1460	0	-1	0	0	Cisco 3620 IOS 11.2(17)P
4128	255	1460	0	-1	0	0	Cisco 3620 IOS 12.0(8), 11.3(11a), 3640 12.1(2)
4128	255	1460	0	-1	0	0	Cisco 4500 IOS 12.0(9)
4128	255	1460	0	-1	0	0	Cisco C2600 IOS 12.0(5)T1
2144	255	536	0	-1	0	0	Cisco IGS 3000 IOS 11.x(16), 2500 IOS 11.2(3)P
32768	64	1460	1	0	0	1	Digital UNIX V4.0E
16384	255	1460	1	0	0	1	FreeBSD 2.2.6-RELEASE
16384	64	1460	1	0	0	1	FreeBSD 2.2.8-RELEASE
16384	64	1460	1	0	0	0	FreeBSD 4.0-STABLE, 3.2-RELEASE
32768	64	1460	1	0	0	0	HP-UX B.10.01 A 9000/712
61440	64	512	0	-1	0	0	IRIX 5.3 / 4.0.5F
61440	64	1460	0	-1	0	0	IRIX 6.3
49152	64	1460	0	0	0	0	IRIX 6.5 / 6.4
32120	64	1460	0	-1	0	0	Linux 2.0.33
512	64	1460	0	52	0	0	Linux 2.0.33
512	64	0	0	-1	0	0	Linux 2.0.35 - 2.0.37
512	64	1460	0	0	0	0	Linux 2.0.38
32120	58	1460	0	-1	0	0	Linux 2.0.38
31072	64	3884	1	0	1	1	Linux 2.2.12-20 (RH 6.1)
32120	32	1460	1	0	1	1	Linux 2.2.13
8192	128	1456	1	0	1	1	Linux 2.2.13
32120	64	1460	1	0	1	1	Linux 2.2.14 o Cobalt Linux 2.2.12C3
32120	64	1460	1	101	1	1	Linux 2.2.15
32120	64	1460	1	190	1	1	Linux 2.2.16
16060	64	1460	1	0	1	1	Linux 2.2.x Debian/Caldera (check)
31856	64	1460	1	0	1	1	Linux 2.4.0-test1
16384	64	1460	0	0	0	1	NetBSD 1.3/i386
32768	128	1460	1	-1	0	0	Novell NetWare 4.11
16384	64	512	0	0	0	1	OpenBSD 2.6
24820	64	1460	1	0	0	0	SCO UnixWare 7.0.1
32696	64	536	0	0	1	1	SCO UnixWare 7.1.0 x86
24820	64	1460	1	0	0	1	SCO UnixWare 7.1.0 x86
8760	64	1460	1	0	0	0	Solaris 2.6 (2)
9140	255	9140	1	0	0	0	Solaris 2.6 (sunsite)
8760	255	1460	1	0	0	0	Solaris 2.6 or 2.7
8760	255	1380	1	0	0	0	Solaris 2.7
33580	255	1460	1	-1	0	0	Solaris 7
16384	64	0	0	-1	0	0	ULTRIX V4.5 (Rev. 47)
16384	128	1460	1	0	1	1	Windows 2000
5840	128	536	1	0	1	1	Windows 95 (3)
8192	32	1456	1	-1	0	0	Windows 95 (?)
8192	128	1460	1	0	1	1	Windows 9x (1)
8192	128	536	1	0	1	1	Windows 9x (2)
2144	64	536	1	0	1	1	Windows 9x (4)
8192	128	1460	1	-1	1	0	Windows NT
8192	128	1460	1	0	0	0	Windows NT 4.0
8192	32	1460	1	0	0	0	Windows NT 4.0
8192	128	25443	1	-1	1	1	Windows NT 4.0 Server SP5

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* – escalado 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)

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:

<table>
<tr><td colspan="32">0 4 10 15 16 24 31</td></tr>
<tr><td colspan="16">PUERTO TCP DE ORIGEN</td><td colspan="16">PUERTO TCP DE DESTINO</td></tr>
<tr><td colspan="32">NÚMERO DE SECUENCIA</td></tr>
<tr><td colspan="32">NÚMERO DE ACUSE DE RECIBO</td></tr>
<tr><td colspan="4">HLEN</td><td colspan="6">RESERVADO</td><td colspan="6">CODE BITS</td><td colspan="16"></td></tr>
<tr><td colspan="16">SUMA DE VERIFICACIÓN TCP</td><td colspan="16">PUNTERO DE URGENCIA</td></tr>
<tr><td colspan="24">OPCIONES TCP (SI LAS HAY)</td><td colspan="8">RELLENO</td></tr>
<tr><td colspan="32">DATOS</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

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

Code Bit (de izquierda a derecha)		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	Este segmento solicita una operación push
13	RST	Iniciación de la conexión
14	SYN	Sincronizar números de secuencia
15	FIN	El emisor ha llegado al final de su flujo de octetos

Bits del campo CODE 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.

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.

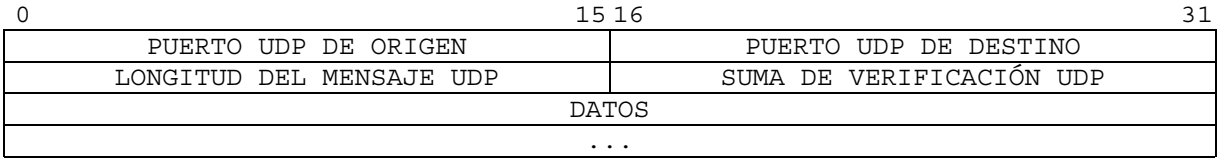
La ventana se utiliza para controlar cuanta información puede estar en tránsito en un momento dado. No es práctico esperar a que cada datagrama enviado sea confirmado antes de transmitir el siguiente, cosa que ralentizaría bastante el proceso. Por otro lado, si se envía la información sin esperar la 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 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 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 la ventana.

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:

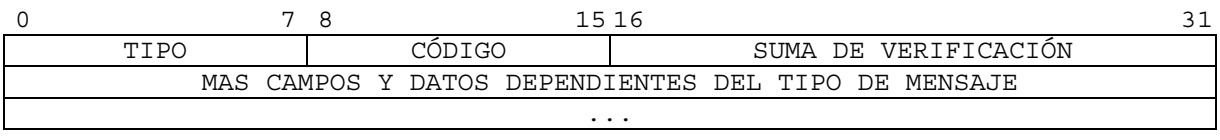


Formato de los campos en un datagrama UDP.

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.

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.



Formato de mensaje ICMP.

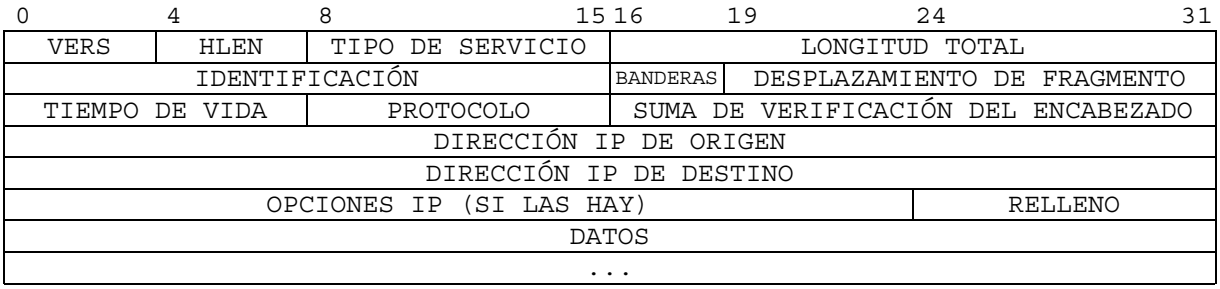
Campo de tipo	Tipo de mensaje ICMP
0	Respuesta de eco
3	Destino inaccesible
4	Disminución de origen
5	Redireccionar (cambiar una ruta)
8	Solicitud de eco
11	Tiempo excedido para un datagrama

12	Problema de parámetros en un datagrama
13	Solicitud de timestamp
14	Respuesta de timestamp
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.

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.



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).

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 reenvíen 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).

Anexo E: algunos puertos “famosos”

Decimal	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
11	TCP / UDP	USERS	Usuarios activos
13	TCP / UDP	DAYTIME	Hora del día
15	TCP / UDP	-	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)
23	TCP	TELNET	Conexión de terminal
25	TCP	SMTP	Protocolo de transporte de correo sencillo
37	TCP / UDP	TIME	Hora
42	TCP / UDP	NAMESERVER	Servidor de nombres de anfitriones
43	TCP / UDP	NICNAME	¿Quién está ahí?
53	TCP / UDP	DOMAIN	Servidor de nombres de dominios
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	TCP	WWW-HTTP	World Wide Web HTTP
93	TCP	DCP	Protocolo de control de dispositivos
95	TCP	SUPDUP	Protocolo SUPDUP
101	TCP	HOSTNAME	Servidor de nombre de anfitrión NIC
102	TCP	ISO-TSAP	ISO-TSAP
103	TCP	X400	Servicio de correo X.400
104	TCP	X400-SND	Envío de correo X.400
110	TCP	POP3	Protocolo de oficina postal v. 3
111	TCP / UDP	SUNRPC	Llamada a procedimiento remoto de SUN
113	TCP	AUTH	Servicio de autenticación
117	TCP	UUCP-PATH	Servicio de trayecto UUCP
119	TCP	NNTP	Protocolo de transferencia de noticias de red
123	UDP	NTP	Protocolo de tiempo de red
129	TCP	PWDGEN	Protocolo generador de clave de acceso
139	TCP	NETBIOS-SSN	Servicio de sesión NETBIOS
161	UDP	SNMP	Monitor de red SNMP
162	UDP	SNMPTRAP	Interrupciones SNMP
512	UDP	BIFF	Comsat UNIX
513	UDP	WHO	Rwho daemon UNIX
514	UDP	CMD	Conexión de sistema
525	UDP	TIMESERVER	Servidor de hora