

# 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](http://www.sindominio.net/hackbcn00)

Alejandro Castán Salinas

[acastan@xtec.cat](mailto:acastan@xtec.cat)



**Análisis activo y pasivo de redes - revisión 3.2 - 15/8/2007**

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 1.2 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.xtec.net/~acastan/textos/>.



# 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 su 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 afinar 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.

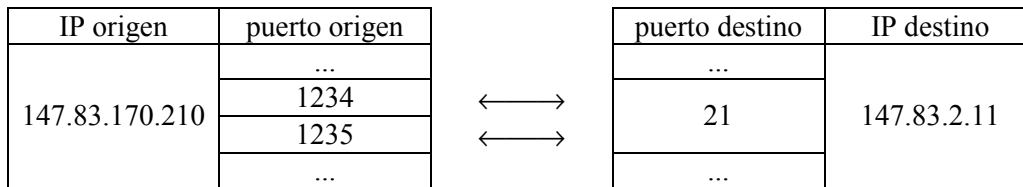
Dos protocolos separados son los encargados de manejar los mensajes TCP/IP (en el anexo D de este documento se entra un poco más en detalle sobre estos protocolos). TCP (“Transmission Control Protocol”) es el responsable de romper el mensaje en datagramas, ensamblar los datagramas en el otro extremo, 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.

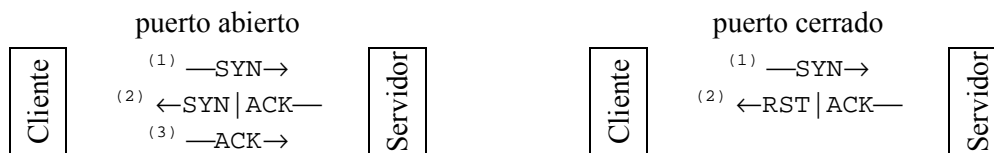
Los números de puertos están divididos en tres rangos: los puertos bien conocidos (del 0 al 1023), los puertos registrados (del 1024 al 49151) y los puertos dinámicos y/o privados (del 49152 hasta el 65535). Los puertos bien conocidos son asignados y controlados por un organismo llamado IANA. En la mayoría 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 simultno 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 del cliente más uno, 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.



### 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/IP, 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 que sólo buscan intentos de conexión secuenciales, realizando el análisis de puertos en orden aleatorio o pseudo-aleatorio. 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, ya que los sistemas de detección de intrusos determinan si un ordenador intenta realizar un análisis de puertos de un sistema detectando todo el tráfico de red generado por la dirección IP de origen en un intervalo fijo de tiempo.

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 algunos 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
WinNT: HKLM\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` (o también `netstat -ie`) 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 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://insecure.org/nmap/nmap\\_doc.html](http://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://insecure.org/nmap/osdetect/>.
- Un pequeño artículo que recopila los métodos para engañar a los analizadores de sistemas operativos es “Un enfoque práctico para engañar la detección remota de SO de Nmap”, que encontrareis en [http://his.sourceforge.net/proy\\_his/papers/](http://his.sourceforge.net/proy_his/papers/).
- 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](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* (<http://insecure.org/nmap/>). Dos herramientas que utilizan un enfoque alternativo a *Nmap* para el análisis de sistemas operativos son *Xprobe2* (<http://sys-security.com/blog/xprobe2>) y *Cron-OS* (<https://gna.org/projects/cronos>). Otra herramienta también valiosa, que permite enviar paquetes a medida, es *hping* (<http://www.hping.org/>).
- Tres herramientas para el análisis pasivo de sistemas operativos son: *SinFP* (<http://www.gomor.org/cgi-bin/sinfp.pl>), *p0f* (<http://lcamtuf.coredump.cx/p0f.shtml>) y *ettercap* (<http://ettercap.sourceforge.net/>).
- Un conocido sniffer para el tráfico TCP es *tcpdump* (<http://www.tcpdump.org/>). Otro conocido sniffer es *Ethereal*, ahora llamado *Wireshark* (<http://www.wireshark.org/>). Por último, *dsniff* (<http://www.monkey.org/~dugsong/dsniff/>) es una imprescindible colección de herramientas para monitorizar redes.

En *tcpdump* se puede conseguir aumentar la velocidad para el análisis pasivo filtrando sólo los paquetes de inicio de conexión (por ejemplo SYN+ACK): `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 las herramientas comerciales *antisniff* (<http://packetstormsecurity.org/sniffers/antisniff/>) o *promiscan* (<http://www.securityfriday.com/products/promiscan.html>) para Windows NT. Como herramientas gratuitas, existen para Unix *Sniffdet* (<http://sniffdet.sourceforge.net/>), y para Windows NT *proDetect* (<http://sourceforge.net/projects/prodetect/>).
- Existen varias herramientas que permiten detectar y responder a análisis de puertos. Mis preferidas son: *snort* (<http://www.snort.org/>), *iplog* (<http://ojnk.sourceforge.net/>), *scanlogd*

(<http://www.openwall.com/scanlogd/>),

y

*portsentry*

(<http://sourceforge.net/projects/sentrytools/>).

- Como curiosidad: una página web que permite analizar el sistema operativo de un ordenador conectado a Internet es <http://www.netcraft.net/>, y una página web que permite analizar los puertos abiertos es <http://nmap-online.com/> y <http://www.tlshopper.com/tools/port-scanner/>.

Todas estas herramientas también las podéis encontrar buscando en el repositorio <http://www.packetstormsecurity.org/>.

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

He aquí una relación de diferentes técnicas de análisis de puertos utilizando paquetes TCP/IP. Para escoger la técnica adecuada a utilizar en un determinado entorno a analizar se deberá tener en cuenta la topología de la red, la presencia de cortafuegos y de sistemas de detección de intrusos, y las características de registro de actividad en los ordenadores destino.

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



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 filtrar y detectar, 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 scan.** 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.



Ventajas: (1) los IDS más modestos (basados en conexión) no registran este intento de conexión 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 scan.** 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.



Ventajas: (1) los paquetes SYN|ACK son capaces de pasar a través de algunos cortafuegos que sólo filtran paquetes SYN a puertos restringidos y (2) los IDS más modestos no registran este intento de conexión.

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 scan.** 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 o igual que 64, o el valor del campo win será diferente de 0.



Ventajas: (1) evita IDS, (2) es difícil de registrar y (3) además 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 BSD.

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



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 scan.** 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.



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 scan.** 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.



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 scan.** 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 sistemas responden con un error ICMP de puerto inalcanzable en sus puertos cerrados. 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), y (2) evita IDS que sólo registran tráfico TCP.

Desventajas: (1) hacen falta privilegios de administrador, (2) es lento, (3) es fácilmente detectable y (4) 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() scan.** 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 scan.** El protocolo IDENT, normalmente asociado al puerto 113, permite descubrir el nombre del 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 scan.** 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.

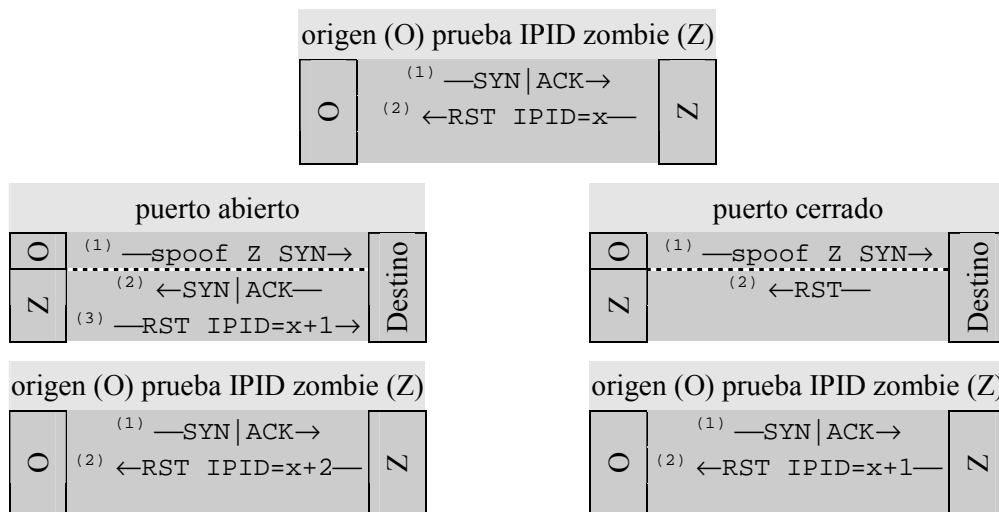
Desventajas: (1) Algunos programas tienen problemas (se cuelgan) al recibir este tipo de paquetes tan pequeños y (2) no funciona con cortafuegos y filtros que encolan y reconstruyen los fragmentos IP antes de procesarlos.

- **FTP bounce scan.** 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 scan.** 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.

- **ICMP echo scan.** No se trata realmente de un análisis de puertos, ya que el protocolo ICMP no tiene una abstracción de puerto. Consiste en determinar qué ordenadores de una red están activos realizando un *ping* (mensaje ICMP de solicitud de eco) directamente sobre ellos o sobre la dirección de *broadcast* de la red. Los ordenadores activos responderán con un mensaje ICMP de respuesta de eco. Variantes de esta técnica, cuando los mensajes ICMP de solicitud de eco (tipo 8) se encuentran filtrados por un cortafuegos, son: utilizar mensajes ICMP de solicitud de marca de tiempo (tipo 13) o de solicitud de máscara de dirección (tipo 17), utilizar paquetes TCP ACK o SYN dirigidos hacia un puerto cualquiera, o enviar un paquete UDP a un puerto cerrado.



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

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

- **IP protocol scan.** 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.



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 66.66.66.66:1078 -> 11.11.11.197:53 SYN
Apr 1 19:02:12 66.66.66.66:1079 -> 11.11.11.198:53 SYN
Apr 1 19:02:12 66.66.66.66:1080 -> 11.11.11.199:53 SYN
Apr 1 19:02:12 66.66.66.66:1081 -> 11.11.11.200:53 SYN
Apr 1 19:02:12 66.66.66.66:1082 -> 11.11.11.201:53 SYN
Apr 1 19:02:12 66.66.66.66:1083 -> 11.11.11.202:53 SYN
```

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 66.66.66.66:1093 -> 11.11.11.49:21 SYN
Apr 1 19:36:01 66.66.66.66:1094 -> 11.11.11.49:23 SYN
Apr 1 19:36:01 66.66.66.66:1095 -> 11.11.11.49:25 SYN
Apr 1 19:36:02 66.66.66.66:1096 -> 11.11.11.49:53 SYN
Apr 1 19:36:02 66.66.66.66:1096 -> 11.11.11.49:79 SYN
Apr 1 19:36:02 66.66.66.66:1097 -> 11.11.11.49:80 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_secuencia (bytes_datos) tamaño_ventana <opciones_TCP>
(señal_no_fragmentación) (tiempo_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

He aquí una relación de diferentes técnicas de análisis de sistemas operativos utilizando paquetes TCP/IP. Debemos ser conscientes de que la calidad de la respuesta de las pruebas puede deteriorarse debido diversos motivos: varios dispositivos pueden tener una misma versión de firmware, un dispositivo puede tener diferentes versiones de firmware, puede no haber cambios en la pila TCP/IP en sucesivas versiones de un sistema operativo, diversos dispositivos de red pueden alterar los campos de los paquetes (un *scrubber* puede limpiar valores maliciosos en ciertos campos, un cortafuegos puede falsear respuestas, un encaminador puede cambiar los valores TOS,...), la presencia de cortafuegos (por ejemplo, si la herramienta *nmap* no encuentra un puerto TCP abierto, un puerto TCP cerrado y un puerto UDP cerrado para realizar sus pruebas, disminuye la precisión de las pruebas), etc.

- **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.
- **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.
- **IPID Sampling.** Algunos sistemas operativos incrementan en 1 el valor del campo IPID para cada nuevo paquete que envían. Windows, en cambio, lo incrementa en 256 para cada nuevo paquete que envía. Otros, como OpenBSD, utilizan un valor aleatorio para dicho campo. Linux y otros sistemas utilizan un IPID de 0 cuando el bit de no fragmentación no está activado.
- **TCP Timestamp.** Algunos sistemas no soportan la opción TCP de marca de tiempo, mientras que otros sistemas incrementan el valor en frecuencias de 2HZ, 100HZ, o 1000HZ, e incluso otros retornan 0. Esta característica también se puede utilizar para determinar cuanto lleva funcionando sin reiniciarse el equipo analizado.
- **Don't Fragment bit.** Algunos sistemas operativos activan el bit IP de no fragmentación en algunos de los paquetes que envían, para conseguir 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, OpenBSD, FreeBSD) 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  $\frac{1}{4}$  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 aumentado el campo IP de longitud, 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.

- **Retransmission Timeout.** Consiste en crear intencionadamente conexiones medio abiertas (enviando un paquete SYN a un puerto abierto y recibiendo sucesivos paquetes SYN|ACK, sin responder con un ACK). Podemos reconocer fácilmente algunos sistemas operativos

midiendo el número de respuestas SYN|ACK, la espera entre respuestas y la presencia opcional de un paquete RST para cerrar la conexión después de varios reintentos.

Uno de los puntos fuertes de esta técnica es que es muy silenciosa. Tan sólo envía un paquete SYN y no llega a establecer la conexión TCP. Existen variaciones de esta técnica que miden los tiempos de retransmisión entre paquetes FIN|ACK después de haber establecido una conexión.

- **Port 0 probe.** Aunque el puerto cero es un número válido para conexiones TCP y UDP, dicho puerto está reservado para un uso especial (RFC 1700) y el sistema operativo lo reasigna automáticamente cuando se especifica el puerto 0 en una conexión a otro equipo. Debido a que las especificaciones de cómo tratar el tráfico dirigido al puerto 0 no son claras (en un principio dicho tráfico no debería existir), cada sistema operativo lo maneja a su manera, por lo que comprobando las respuestas ante paquetes dirigidos al puerto cero podemos distinguir entre diferentes sistemas operativos.
- **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

## Ettercap

Con mucho, la tabla más completa es la de esta herramienta (más de mil entradas). Por su extensión no la incluyo en el anexo, pero la podéis encontrar siempre actualizada en su CVS:

[http://ettercap.cvs.sourceforge.net/ettercap/ettercap\\_ng/share/etter.finger.os](http://ettercap.cvs.sourceforge.net/ettercap/ettercap_ng/share/etter.finger.os)

## Siphon

Parece ser que el proyecto *siphon* está muerto casi desde sus inicios, así que tampoco incluyo su tabla.

## P0f\*

La siguiente tabla enumera las firmas para conexiones de entrada (paquetes SYN). *P0f* también dispone de una tabla para conexiones de salida (paquetes SYN+ACK) y para conexiones rechazadas (paquetes RST y RST+ACK).

Window	TTL	DF	PS	OO	Q	Sistema Operativo y versión
16384	64	0	44	M512	.	AIX 4.3.2 and earlier
16384	64	0	60	M512,N,W%2,N,N,T	.	AIX 4.3.3-5.2 (1)
32768	64	0	60	M512,N,W%2,N,N,T	.	AIX 4.3.3-5.2 (2)
65535	64	0	60	M512,N,W%2,N,N,T	.	AIX 4.3.3-5.2 (3)
65535	64	0	64	M*,N,W1,N,N,T,N,N,S	.	AIX 5.3 ML1
512	64	0	44	M*	.	Linux 2.0.3x (1)
16384	64	0	44	M*	.	Linux 2.0.3x (2)
2	64	0	44	M*	.	Linux 2.0.3x (MkLinux) on Mac (1)
64	64	0	44	M*	.	Linux 2.0.3x (MkLinux) on Mac (2)
S4	64	1	60	M1360,S,T,N,W0	.	Linux 2.4 (Google crawler)
S2	64	1	60	M*,S,T,N,W0	.	Linux 2.4 (big boy)
S3	64	1	60	M*,S,T,N,W0	.	Linux 2.4.18 and newer
S4	64	1	60	M*,S,T,N,W0	.	Linux 2.4/2.6
S3	64	1	60	M*,S,T,N,W1	.	Linux 2.5 (sometimes 2.4) (1)
S4	64	1	60	M*,S,T,N,W1	.	Linux 2.5/2.6 (sometimes 2.4) (2)
S20	64	1	60	M*,S,T,N,W0	.	Linux 2.2.20 and newer
S22	64	1	60	M*,S,T,N,W0	.	Linux 2.2 (1)
S11	64	1	60	M*,S,T,N,W0	.	Linux 2.2 (2)
S4	64	1	48	M1460,N,W0	.	Linux 2.4 in cluster
T4	64	1	60	M1412,S,T,N,W0	.	Linux 2.4 (late, uncommon)
32767	64	1	60	M16396,S,T,N,W0	.	Linux 2.4 (local)
S8	64	1	60	M3884,S,T,N,W0	.	Linux 2.2 (local)
16384	64	1	60	M*,S,T,N,W0	.	Linux 2.2 (Opera?)

### Leyenda:

Window = *window size* - tamaño de ventana (Snn = multiplo de MSS y Tnn = multiplo de MTU)

TTL = *time to live* - tiempo de vida

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

PS = *packet size* - tamaño de paquete

OO = *option value and order* - valor de las opciones IP y su orden (N = NOP, E = EOL, Wnnn = escala de ventana, Mnnn = tamaño máximo de segmento, S = SACK OK, T = marca de tiempo, T0 = marca de tiempo con valor 0, ?n = número de opción no reconocido, . = sin opciones)

Q = *quirks* - peculiaridades o fallos de la pila TCP/IP (P = opciones detrás de EOL, Z = IP ID cero, I = opciones IP especificadas, U = puntero de urgencia diferente de cero, X = campo sin uso diferente de zero, A = número de ACK diferente de cero, T = segunda marca de tiempo diferente de cero, F = bits poco usuales activos (PUSH, URG, etc), D = data payload, ! = segmento de opciones roto, . = ningún rasgo característico)

32767	64	1	60	M*,S,T,N,W0	.	Linux 2.4 (Opera?)
S4	64	1	52	M*,N,N,S,N,W0	.	Linux 2.4 w/o timestamps
S22	64	1	52	M*,N,N,S,N,W0	.	Linux 2.2 w/o timestamps
16384	64	1	44	M*	.	FreeBSD 2.0-4.1
16384	64	1	60	M*,N,W0,N,N,T	.	FreeBSD 4.4 (1)
1024	64	1	60	M*,N,W0,N,N,T	.	FreeBSD 4.4 (2)
57344	64	1	44	M*	.	FreeBSD 4.6-4.8 (no RFC1323)
57344	64	1	60	M*,N,W0,N,N,T	.	FreeBSD 4.6-4.8
32768	64	1	60	M*,N,W0,N,N,T	.	FreeBSD 4.8-5.1 (MacOS X 10.2-10.3)
65535	64	1	60	M*,N,W0,N,N,T	.	FreeBSD 4.7-5.1 (MacOS X 10.2-10.3)(1)
65535	64	1	60	M*,N,W1,N,N,T	.	FreeBSD 4.7-5.1 (MacOS X 10.2-10.3)(2)
65535	64	1	60	M*,N,W0,N,N,T	Z	FreeBSD 5.1-current (1)
65535	64	1	60	M*,N,W1,N,N,T	Z	FreeBSD 5.1-current (2)
65535	64	1	60	M*,N,W2,N,N,T	Z	FreeBSD 5.1-current (3)
16384	64	0	60	M*,N,W0,N,N,T	.	NetBSD 1.3
65535	64	0	60	M*,N,W0,N,N,T0	.	NetBSD 1.6 (Opera)
16384	64	1	60	M*,N,W0,N,N,T0	.	NetBSD 1.6
65535	64	1	60	M*,N,W1,N,N,T0	.	NetBSD 1.6W-current (DF)
65535	64	1	60	M*,N,W0,N,N,T0	.	NetBSD 1.6X (DF)
16384	64	1	64	M*,N,N,S,N,W0,N,N,T	.	OpenBSD 3.0-3.4
57344	64	1	64	M*,N,N,S,N,W0,N,N,T	.	OpenBSD 3.3-3.4
16384	64	0	64	M*,N,N,S,N,W0,N,N,T	.	OpenBSD 3.0-3.4 (scrub)
65535	64	1	64	M*,N,N,S,N,W0,N,N,T	.	OpenBSD 3.0-3.4 (Opera)
S17	64	1	64	N,W3,N,N,T0,N,N,S,M*	.	Solaris 8 (RFC1323 on)
S17	64	1	48	N,N,S,M*	.	Solaris 8 (1)
S17	255	1	44	M*	.	Solaris 2.5 to 7
S6	255	1	44	M*	.	Solaris 2.6/7
S23	64	1	48	N,N,S,M*	.	Solaris 8 (2)
S34	64	1	48	M*,N,N,S	.	Solaris 9
S44	255	1	44	M*	.	Solaris 7
4096	64	0	44	M1460	.	SunOS 4.1.x
49152	60	0	44	M*	.	IRIX 6.4
61440	60	0	44	M*	.	IRIX 6.2-6.5
49152	60	0	52	M*,N,W2,N,N,S	.	IRIX 6.5 (RFC1323) (1)
49152	60	0	52	M*,N,W3,N,N,S	.	IRIX 6.5 (RFC1323) (2)
61440	60	0	48	M*,N,N,S	.	IRIX 6.5.12-6.5.21 (1)
49152	60	0	48	M*,N,N,S	.	IRIX 6.5.12-6.5.21 (2)
32768	60	1	48	M*,N,W0	.	Tru64 4.0 (or OS/2 Warp 4)
32768	60	0	48	M*,N,W0	.	Tru64 5.0 (or OpenVMS 7.x on Compaq
5.0 stack)						
8192	60	0	44	M1460	.	Tru64 5.1 (no RFC1323) (or QNX 6)
61440	60	0	48	M*,N,W0	.	Tru64 v5.1a JP4 (or OpenVMS 7.x on
Compaq 5.x stack)						
6144	64	1	60	M*,N,W0,N,N,T	.	OpenVMS 7.2 (Multinet 4.3-4.4 stack)
S2	255	1	48	M*,W0,E	.	MacOS 8.6 classic
16616	255	1	48	M*,W0,E	.	MacOS 7.3-8.6 (OTTCP)
16616	255	1	48	M*,N,N,N,E	.	MacOS 8.1-8.6 (OTTCP)
32768	255	1	48	M*,W0,N	.	MacOS 9.0-9.2
32768	255	1	48	M1380,N,N,N,N	.	MacOS 9.1 (1) (OT 2.7.4)
65535	255	1	48	M*,N,N,N,N	.	MacOS 9.1 (2) (OT 2.7.4)
32768	64	0	60	M*,N,W0,N,N,T	.	MacOS X 10.2
8192	32	1	44	M*	.	Windows 3.11 (Tucows)
S44	64	1	64	M*,N,W0,N,N,T0,N,N,S	.	Windows 95
8192	128	1	64	M*,N,W0,N,N,T0,N,N,S	.	Windows 95b
S44	32	1	48	M*,N,N,S	.	Windows 98 (low TTL) (1)
8192	32	1	48	M*,N,N,S	.	Windows 98 (low TTL) (2)
%8192	64	1	48	M536,N,N,S	.	Windows 98 (13)
%8192	128	1	48	M536,N,N,S	.	Windows 98 (15)
S4	64	1	48	M*,N,N,S	.	Windows 98 (1)
S6	64	1	48	M*,N,N,S	.	Windows 98 (2)
S12	64	1	48	M*,N,N,S	.	Windows 98 (3)
T30	64	1	64	M1460,N,W0,N,N,T0,N,N,S	.	Windows 98 (16)
32767	64	1	48	M*,N,N,S	.	Windows 98 (4)
37300	64	1	48	M*,N,N,S	.	Windows 98 (5)
46080	64	1	52	M*,N,W3,N,N,S	.	Windows 98 (RFC1323)
65535	64	1	44	M*	.	Windows 98 (no sack)
S16	128	1	48	M*,N,N,S	.	Windows 98 (6)

S16	128	1	64	M*,N,W0,N,N,T0,N,N,S	.	Windows 98 (7)
S26	128	1	48	M*,N,N,S	.	Windows 98 (8)
T30	128	1	48	M*,N,N,S	.	Windows 98 (9)
32767	128	1	52	M*,N,W0,N,N,S	.	Windows 98 (10)
60352	128	1	48	M*,N,N,S	.	Windows 98 (11)
60352	128	1	64	M*,N,W2,N,N,T0,N,N,S	.	Windows 98 (12)
T31	128	1	44	M1414	.	Windows NT 4.0 SP6a (1)
64512	128	1	44	M1414	.	Windows NT 4.0 SP6a (2)
8192	128	1	44	M*	.	Windows NT 4.0 (older)
65535	128	1	48	M*,N,N,S	.	Windows 2000 SP4, XP SP1
%8192	128	1	48	M*,N,N,S	.	Windows 2000 SP2+, XP SP1, 98 4.10.222
S20	128	1	48	M*,N,N,S	.	Windows 2000 SP3
S45	128	1	48	M*,N,N,S	.	Windows 2000 SP4, XP SP 1 (2)
40320	128	1	48	M*,N,N,S	.	Windows 2000 SP4
S6	128	1	48	M*,N,N,S	.	Windows XP, 2000 SP2+
S12	128	1	48	M*,N,N,S	.	Windows XP SP1 (1)
S44	128	1	48	M*,N,N,S	.	Windows XP Pro SP1, 2000 SP3
64512	128	1	48	M*,N,N,S	.	Windows XP SP1, 2000 SP3 (2)
32767	128	1	48	M*,N,N,S	.	Windows XP SP1, 2000 SP4 (3)
S52	128	1	48	M1260,N,N,S	.	Windows XP/2000 via Cisco
65520	128	1	48	M*,N,N,S	.	Windows XP bare-bone
16384	128	1	52	M536,N,W0,N,N,S	.	Windows 2000 w/ZoneAlarm?
2048	255	0	40	.	.	Windows .NET Enterprise Server
*	128	1	48	M*,N,N,S	U	Windows XP/2000 downloading (leak!)
32768	64	1	44	M*	.	HP-UX B.10.20
32768	64	1	48	M*,W0,N	.	HP-UX 11.00-11.11
0	64	0	48	M*,W0,N	.	HP-UX B.11.00 A (RFC1323)
16384	64	1	68	M1460,N,W0,N,N,T,N,N,?12	.	RISC OS 3.70-4.36 (inet 5.04)
12288	32	0	44	M536	.	RISC OS 3.70 inet 4.10
4096	64	1	56	M1460,N,N,T T	.	RISC OS 3.70 freenet 2.00
8192	64	1	60	M1460,N,W0,N,N,T	.	BSD/OS 3.1-4.3 (or MacOS X 10.2)
4096	64	0	44	M1420	.	NewtonOS 2.1
S8	64	0	44	M512	.	NeXTSTEP 3.3
1024	255	0	48	M*,N,W0	.	BeOS 5.0-5.1
12288	255	0	44	M*	.	BeOS 5.0.x
8192	64	1	60	M1440,N,W0,N,N,T	.	OS/400 V4R4/R5
8192	64	0	44	M536	.	OS/400 V4R3/M0
4096	64	1	60	M1440,N,W0,N,N,T	.	OS/400 V4R5 + CF67032
28672	64	0	44	M1460	A	OS/390 ?
16384	64	0	40	.	.	ULTRIX 4.5
S16	64	0	44	M512	.	QNX demodisk
16384	128	1	44	M1460	.	Novell NetWare 5.0
6144	128	1	44	M1460	.	Novell IntranetWare 4.11
6144	128	1	52	M*,W0,N,S,N,N	.	Novell Netware 6 SP3
S3	64	1	60	M1460,N,W0,N,N,T	.	SCO UnixWare 7.1
S23	64	1	44	M1380	.	SCO OpenServer 5.0
2048	255	0	44	M536	.	DOS Arachne via WATTCP/1.05
S56	64	0	44	M512	.	OS/2 4
0	64	0	44	M1460	A	TOPS-20 version 7
S32	64	1	56	M*,N,N,S,N,N,?12	.	AMIGA 3.9 BB2 with Miami stack
# ----- Firewalls / routers -----						
S12	64	1	44	M1460	.	@Checkpoint (unknown 1)
S12	64	1	48	N,N,S,M1460	.	@Checkpoint (unknown 2)
4096	32	0	44	M1460	.	ExtremeWare 4.x
60352	64	0	52	M1460,N,W2,N,N,S	.	Clavister firewall 7.x
S32	64	0	68	M512,N,W0,N,N,T,N,N,?12	.	Nokia IPSO w/Checkpoint NG FP3
S4	64	1	60	W0,N,S,T,M1460	.	FortiNet FortiGate 50
# ----- Switches and other stuff -----						

```

4128 255 0 44 M* Z Cisco 7200, Catalyst 3500, et
S8 255 0 44 M* . Cisco 12008
60352 128 1 64 M1460,N,W2,N,N,T,N,N,S . Alteon ACEswitch
64512 128 1 44 M1370 . Nortel Contivity Client

# ----- Caches and whatnots -----

8192 64 1 64 M1460,N,N,S,N,W0,N,N,T . NetCache 5.2
16384 64 1 64 M1460,N,N,S,N,W0,N . NetCache 5.3
65535 64 1 64 M1460,N,N,S,N,W*,N,N,T . NetCache 5.3-5.5
20480 64 1 64 M1460,N,N,S,N,W0,N,N,T . NetCache 4.1
32850 64 1 64 N,W1,N,N,T,N,N,S,M* . NetCache Data OnTap 5.x
65535 64 0 60 M1460,N,W0,N,N,T . CacheFlow CacheOS ?
8192 64 0 60 M1380,N,N,N,N,N,N,T . CacheFlow CacheOS 1.1
S4 64 0 48 M1460,N,N,S . Cisco Content Engine
27085 128 0 40 . Dell PowerApp cache (Linux-based)
65535 255 1 48 N,W1,M1460 . Inktomi crawler
S1 255 1 60 M1460,S,T,N,W0 . LookSmart ZyBorg
16384 255 0 40 . Proxyblocker (what's this?)

# ----- Embedded systems -----

S9 255 0 44 M536 . PalmOS Tungsten C
S5 255 0 44 M536 . PalmOS 3/4
S4 255 0 44 M536 . PalmOS 3.5
2948 255 0 44 M536 . PalmOS 3.5.3 (Handera)
S23 64 1 64 N,W1,N,N,T,N,N,S,M1460 . SymbianOS 7
8192 255 0 44 M1460 . SymbianOS 6048 (on Nokia 7650?)
8192 255 0 44 M536 . SymbianOS (on Nokia 9210?)
5840 64 1 60 M1452,S,T,N,W1 . Zaurus 3.10
32768 128 1 64 M1460,N,W0,N,N,T0,N,N,S . PocketPC 2002
S1 255 0 44 M346 . Contiki 1.1-rc0
4096 128 0 44 M1460 . Sega Dreamcast Dreamkey 3.0
T5 64 0 44 M536 . Sega Dreamcast HKT-3020 (browser disc)
S22 64 1 44 M1460 . Sony Playstation 2 (SOCOM?)
S12 64 0 44 M1452 . AXIS Printer Server 5600 v5.64

# ----- Scanners -----

1024 64 0 40 . -*NMAP syn scan (1)
2048 64 0 40 . -*NMAP syn scan (2)
3072 64 0 40 . -*NMAP syn scan (3)
4096 64 0 40 . -*NMAP syn scan (4)
1024 64 0 60 W10,N,M265,T,E P -*NMAP OS detection probe (1)
2048 64 0 60 W10,N,M265,T,E P -*NMAP OS detection probe (2)
3072 64 0 60 W10,N,M265,T,E P -*NMAP OS detection probe (3)
4096 64 0 60 W10,N,M265,T,E P -*NMAP OS detection probe (4)
1024 64 0 60 W10,N,M265,T,E PF -*NMAP OS detection probe w/flags (1)
2048 64 0 60 W10,N,M265,T,E PF -*NMAP OS detection probe w/flags (2)
3072 64 0 60 W10,N,M265,T,E PF -*NMAP OS detection probe w/flags (3)
4096 64 0 60 W10,N,M265,T,E PF -*NMAP OS detection probe w/flags (4)
56922 128 0 40 . A -@Mysterious port scanner (?)
5792 64 1 60 M1460,S,T,N,W0 T -@Mysterious NAT device (2nd tstamp)

```

## 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.

Tipo	Código	Significado
0	0	Respuesta de eco
3	0	Red inalcanzable
3	1	Equipo inalcanzable
3	3	Puerto inalcanzable
4	0	Origen acallado
5	0	Redireccionar (cambiar una ruta)
8	0	Solicitud de eco
11	0	Tiempo excedido para un datagrama
12	0	Problema de parámetros en un datagrama
13	0	Solicitud de marca de tiempo
14	0	Respuesta de marca de tiempo
15	0	Solicitud de información (obsoleto)
16	0	Respuesta de información (obsoleto)
17	0	Solicitud de máscara de dirección
18	0	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”

He aquí una lista con algunos de los puertos más utilizados y sus servicios asociados. Encontrareis la lista completa en <http://www.iana.org/assignments/port-numbers>.

Tan importante como esta lista de puertos, es la lista de puertos por defecto utilizados por troyanos. Encontrareis varias listas actualizadas buscando “trojan port list” en <http://www.google.com>, y en <http://www.chebucto.ns.ca/~rakerman/trojan-port-table.html>.

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
11	TCP	SYSTAT	Usuarios activos
13	TCP / UDP	DAYTIME	Hora del día (en formato humano)
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	Sincronización de la hora (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
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 / REXEC	Notific. correo / Ejecución remota de procesos
513	UDP / TCP	WHO / RLOGIN	Quien está conectado / Conexión remota a telnet
514	UDP / TCP	SYSLOG / RSHELL	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”)