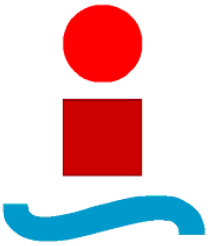
PROYECTO FIN DE CARRERA
Ingeniero de Telecomunicación



ESCUELA SUPERIOR DE INGENIEROS

Universidad de Sevilla

Estudio Teórico sobre Seguridad en Redes Linux

Autor: Antonio Carmona Ramos
Tutor: Ángel García Ramírez

Sevilla, Mayo de 2003

A mis padres

Resumen

El presente proyecto constituye un estudio básico de seguridad en redes de ordenadores. Se estudia la seguridad desde el punto de vista de los niveles más bajos en la pila de protocolos TCP/IP, así como en lo relativo al acceso físico y autenticación de usuario. Se estudian las vulnerabilidades de los sistemas y se apuntan métodos para contrarrestar el efecto de las mismas. En la mayoría de los casos se particulariza este estudio para las redes de sistemas Linux.

En un principio se presenta el sistema operativo Linux como un sistema robusto y fiable, ideal para su funcionamiento en entornos de red. Más tarde se exponen la necesidad de restringir el acceso físico a los equipos de la red y las implicaciones de la distribución de la red en la seguridad del sistema. En un capítulo posterior se manifiesta la importancia de elegir un esquema de autenticación seguro, y el papel tan importante que en éste juega la correcta elección de contraseñas por parte de los usuarios.

Cubiertos los aspectos relativos a la seguridad de los sistemas respecto a su entorno más próximo, se estudian las vulnerabilidades inherentes a la pila de protocolos TCP/IP y niveles inferiores. De este modo, se muestran las escuchas electrónicas, cómo realizar ataques de este tipo y sencillas formas de protegerse contra ellos. Más tarde se tratan los scanners, herramientas que realizan sondeos automatizados para detectar máquinas activas, puertos abiertos e información previa para posteriores ataques. Se describen los fundamentos de los ataques de denegación del servicio y algunos ataques comunes. Por último, se estudian las bases de los cortafuegos, las posibilidades que ofrecen para controlar el tráfico que circula por una red, y se presenta la infraestructura de firewall de Linux, Netfilter, y su herramienta de administración, iptables.

Términos Clave

Sistema Operativo. Linux. TCP/IP. UDP. Ethernet. Internet. LAN. NOC. BIOS. LiLo. Topología. Autenticación. Contraseña. Cifrado. Puerto. Sniffer. Scanner. Denegación del servicio. Firewall. Router. Gateway. Filtrado de paquetes. Traducción de direcciones. Root. Administrador.

Objetivos

Realizar un estudio sobre seguridad en redes, las vulnerabilidades en los niveles físico, de enlace, de red y de transporte de la pila de protocolos de internet, especialmente en entornos de red Linux. Analizar las herramientas de seguridad disponibles para estos niveles. Proponer procedimientos para contrarrestar las vulnerabilidades de los niveles bajos de la pila TCP/IP. Constituir el punto de partida para posteriores estudios.

Indice

Memoria

Pliego de condiciones

Planos

Presupuesto

Memoria

Capítulo 1

EL SISTEMA OPERATIVO LINUX

Linux es un sistema operativo de libre distribución similar a Unix. Por tanto, Linux posee muchas características en común con Unix; Linux es multitarea, multiusuario, multiplataforma y multiprocesador. Además, Linux ofrece protección de memoria entre procesos (lo que impide que un proceso cuelgue todo el sistema), y un control completo de éstos. Además, Linux interacciona de forma sencilla y transparente con otros sistemas operativos del mercado (MS-DOS, Windows 9x/Me/XP/NT/2000, Novell, OS2 y Unix). Linux ofrece soporte para redes TCP/IP, Novell, NT y Appletalk, al tiempo que es compatible con todo tipo de hardware.

1.1. Sistemas Operativos

Veamos algunos conceptos relativos a los ordenadores. Un computador digital es una máquina capaz de procesar información y dar solución a problemas mediante la ejecución de instrucciones. Las instrucciones se agrupan en programas, cada uno de los cuales define una tarea. Pero el conjunto de instrucciones que reconocen los circuitos electrónicos de una computadora es bastante limitado y las operaciones que definen son muy simples. A ese conjunto limitado de instrucciones reconocido por los circuitos electrónicos de una computadora se le llama *lenguaje máquina*. La programación en lenguaje máquina es complicada, y para hacer más cómoda la

programación de computadoras nace el concepto de *sistema operativo*.

A grandes rasgos, el sistema operativo de una computadora puede ser visto como una capa que envuelve al hardware, ocultando los detalles dependientes del mismo, que presenta una interfaz de comunicación, más o menos semejante para distintas máquinas, entre los procesos y el hardware. En definitiva, el sistema operativo realiza una abstracción de las características especificas de cada máquina, de forma que el programador no tiene que preocuparse de ellas. Por ejemplo, un proceso puede necesitar leer un bloque de ficheros. Es el sistema operativo quien maneja el disco o disquete (el proceso no se preocupa de ello) y transfiere los datos. El proceso sólo da la orden de leer.

Otra de las funciones del sistema operativo es la de gestionar los recursos de la máquina. Para ello, conoce cuáles son dichos recursos y lleva el control de los recursos asignados. En caso de conflicto en el acceso a un recurso compartido, es el sistema operativo, quien se encarga de resolverlo.

En sentido estricto, al hablar de sistema operativo, deberíamos referirnos sólo al *kernel* (núcleo) del mismo. Alrededor del núcleo, se ejecutan aplicaciones, comandos y servicios, que debido a lo habitual de su uso, se habla de ellas como si del propio sistema operativo se tratara.

Un sistema operativo multitarea es aquel que permite que varios procesos se ejecuten simultáneamente. Si los procesos que se ejecutan simultáneamente lo hacen sobre un mismo procesador, entonces estaremos ante un sistema en tiempo compartido (a cada proceso se le asigna un periodo de tiempo de procesador y van rotando). En este caso es el sistema operativo quien debe resolver la entrada en ejecución y bloqueo de cada proceso.

Con un sistema operativo multiprocesador, en el que existe más de un procesador, podremos tener varios procesos ejecutandose simultáneamente sin necesidad de recurrir a la multiprogramación (tiempo compartido). Será una vez más el sistema operativo quien se encargue de ubicar cada proceso en uno u otro procesador, realizando esto de forma totalmente transparente para el programador.

Un sistema operativo multiusuario es aquel que permite la

conexión de más de un usuario a la misma máquina. Se debe encargar de gestionar los accesos de los usuarios al sistema y los recursos, así como de proteger a los usuarios de eventuales malintencionados.

Un sistema multiplataforma será aquel que permite su instalación en distintos tipos de hardware.

1.2. Semejanzas entre Linux e Unix

En muchos libros y documentos de internet se dice que Linux es un clon de Unix. Veamos algunas características de Unix.

Unix es un sistema operativo multiusuario y multitarea, que corre en diferentes computadoras, desde supercomputadoras, *mainframes*, minicomputadoras, computadoras personales y estaciones de trabajo.

Unix fue creado a principios de los setenta por los científicos en los laboratorios Bell. Fue específicamente diseñado para proveer una manera de manejar científica y especializadamente las aplicaciones computacionales. Unix ha servido como modelo para la gran mayoría de sistemas operativos para PC y Linux es uno de ellos.

1.3. Características de Linux

Linux utiliza la llamada multitarea preventiva, la cual asegura que todos los programas que se están utilizando en un momento dado serán ejecutados, siendo el sistema operativo el encargado de ceder tiempo de microprocesador a cada programa.

Las plataformas en las que en un principio se puede utilizar Linux son Intel 386, 486, Pentium, Pentium Pro, Pentium II, III y IV, Amiga y Atari. Asimismo, también existen versiones para

su utilización en otras plataformas, como Alpha, ARM, MIPS, PowerPC y SPARC. El soporte para sistemas con mas de un procesador esta disponible para Intel y SPARC.

Linux funciona en modo protegido 386 y proporciona protección de la memoria entre procesos, de manera que uno de ellos no pueda colgar el sistema. Además Linux carga ejecutables por demanda, es decir, sólo lee del disco aquellas partes de un programa que están siendo usadas en un momento dado.

Gracias a la política de copia en escritura para la compartición de páginas entre ejecutables, varios procesos pueden usar la misma zona de memoria para ejecutarse. Cuando alguno intenta escribir en esa memoria, la página se copia a otro lugar. Esta política de copia en escritura tiene dos beneficios: aumenta la velocidad y reduce el uso de memoria.

La memoria virtual usa paginación (sin intercambio de procesos completos) a disco. Además, la memoria se gestiona como un recurso unificado para los programas de usuario y para la caché de disco, de tal forma que toda la memoria libre puede ser usada para caché y ésta puede a su vez reducirse cuando se ejecuten grandes programas.

Linux dispone de librerías compartidas de carga dinámica (DLL's) y librerías estáticas.

Con el objeto de posibilitar los análisis después de un desastre se realizan volcados de estado (core dumas), permitiendo el uso de depuradores sobre los programas no sólo en ejecución sino también tras ser abortados por cualquier motivo.

Linux es compatible con POSIX, System V y BSD a nivel fuente y ofrece control de tareas POSIX.

Linux posee emulación de iBCS2, casi completamente compatible con SCO, SVR3 y SVR4 a nivel binario.

Por último, como Linux es tratado bajo los términos de la licencia GNU, todo el código fuente está disponible, incluyendo el núcleo completo y todos los drivers, las herramientas de desarrollo y todos los programas de usuario; además todo ello se puede distribuir libremente. Hay algunos programas comerciales que están siendo ofrecidos para Linux actualmente sin código fuente, pero todo lo que ha sido gratuito sigue siendo gratuito.

Linux tiene implementada en el núcleo la emulación de coprocesador matemático, de tal forma que los programas no tengan que hacer su propia emulación matemática. Cualquier máquina que ejecute Linux parecerá dotada de coprocesador matemático. Por supuesto, si el ordenador ya tiene una FPU (unidad de coma flotante), esta será usada en lugar de la emulación, pudiendo incluso compilar tu propio kernel sin la emulación matemática y conseguir un pequeño ahorro de memoria.

Linux ofrece además múltiples consolas virtuales: varias sesiones de login a través de la consola entre las que se puede cambiar con las combinaciones adecuadas de teclas (totalmente independiente del hardware de video). Se crean dinámicamente y puede haber hasta 64.

Linux proporciona soporte para varios sistemas de archivo comunes, incluyendo minix-1, Xenix y todos los sistemas de archivo típicos de System V, y tiene un avanzado sistema de archivos propio con una capacidad de hasta 4 Tb y nombres de archivos de hasta 255 caracteres de longitud. Además ofrece soporte en sólo lectura de HPFS-2 del OS/2 2.1. Por otro lado, el sistema de archivos de CD-ROM lee todos los formatos estándar de CD-ROM.

Otra de las características que tiene Linux y que lo hace especialmente atractivo para la incorporación a redes heterogéneas es el acceso transparente a particiones MS-DOS (o a particiones OS/2 FAT) mediante un sistema de archivos especial: no es necesario ningún comando especial para usar la partición MS-DOS, esta parece un sistema de archivos normal de Unix (excepto por algunas restricciones en los nombres de archivo, permisos, y las características del sistema de archivo Unix no presentes en la FAT). El soporte para VFAT (Windows NT, Windows 95) ha sido añadido al núcleo de desarrollo y estará en la próxima versión estable. Existe también la posibilidad instalar Linux en un sistema de archivos especial llamado UMSDOS, de forma que el sistema operativo Linux sea visto por UMSDOS como un gran archivo.

Es bien conocido que Linux es un sistema operativo que ofrece muchos servicios de red. Para ello, muchos protocolos de red están incluidos en el kernel (TCP, IPv4, IPv6, AX.25, X.25, IPX, DDP, Netrom, etc)y ofrece la posibilidad de trabajar como cliente y servidor de muchos servicios de red, incluyendo ftp, nntp, rlogin, http, telnet, NFS, dhcp, etc. También puede

comunicarse mediante el protocolo Appletalk, y posee software cliente y servidor para redes Netware. Para la comunicación en redes Microsoft, existe el paquete SAMBA que permite que un cliente Windows acceda a un servidor Linux y viceversa.

Gracias a todas estas características y a alguna más que probablemente pasemos por alto, Linux se convierte en un perfecto candidato para el desempeño de tareas críticas, entre las que podemos destacar:

- Servidor de comunicaciones: puede trabajar como gateway, firewall, proxy o servidor de módems, entre otros, ofreciendo además otros servicios populares como caché www o autenticación de usuarios.

- Servidor de Intranet: ficheros, web, correo electrónico, DNS, gestión de bases de datos, etc.

Linux proporciona control de acceso a redes o la capacidad para permitir o no a determinados usuarios y máquinas conectarse entre sí. Es posible implantar reglas de acceso muy refinadas, dependiendo tanto del usuario como de la máquina desde la que se intenta acceder.

Por otro lado, Linux proporciona varias opciones de cifrado punto a punto. En el tránsito de los datos a través de las múltiples pasarelas, encaminadores y equipos de la red por los que circulará la información, los datos son susceptibles de ser captados. El cifrado disminuye enormemente la probabilidad de que los datos capturados sean inteligibles.

Otra de las muchas capacidades interesantes de Linux es la de registro. A pesar de que definamos correctamente las políticas de control de acceso a la red, de que configuremos adecuadamente los servicios, de que dispongamos de un restrictivo firewall, existe la posibilidad de que alguien burle la seguridad del sistema. En estos casos, es tremendamente útil disponer de una historia detallada de todos los eventos ocurridos antes de la intrusión. El registro será nuestra única evidencia real de que se ha producido un ataque. Linux graba registros a nivel de red, host y usuario. Entre las funciones de registro de Linux están las siguientes:

- Registra todos los mensajes del sistema y del núcleo.
- Registra todas las conexiones de la red, la dirección IP de cada una, longitud y en casi todos los casos también

registra el nombre de usuario y sistema operativo del atacante.
- Registra los archivos solicitados por los usuarios remotos.
- Puede registrar los procesos que pertenecen a cualquier usuario.
- Puede registrar todos los comandos enviados por cualquier usuario.

Linux posee además capacidad de detección de intrusiones y en consecuencia no permitirlas. Es posible hacer que Linux registre los intentos de intrusión y que avise cuando se produzcan. También es posible establecer las acciones a realizar cuando una evento de este tipo se produzca, en función de las acciones realizadas por el atacante. Por último, es posible que Linux engañe al atacante haciendole creer que el sistema tiene otras características u otro sistema operativo.

En resumen, Linux se presenta como un sistema operativo fiable, potente, que ofrece multitud de posibilidades de trabajo en red y con unas características de seguridad que lo hacen ideal para tareas críticas en una red.

Capítulo 2

SEGURIDAD FÍSICA Y DE ARRANQUE

Debido a la penetración en la población que ha tenido en los últimos años la red internet, cada vez hay más y más potenciales atacantes de nuestro sistema. Sin embargo, este hecho puede hacernos caer en el error de concentrar nuestros esfuerzos en la correcta configuración de nuestra red y de los servicios que ofrecemos al exterior, pasando por alto un aspecto, si cabe, mucho más importante: su seguridad física. De hecho, de nada habrá servido que hayamos pasado interminables horas leyendo documentación y escribiendo extensos archivos de configuración, si el acceso físico al servidor es libre o no está debidamente controlado. De nada servirá todo esto si alguien consigue llegar hasta él y apagarlo, desenchufar cables o, lo que es peor, robarlo o causarle daños irreparables. Es por esta razón por la que, siempre que tratemos de asegurar un sistema, debemos comenzar por asegurar su ubicación física.

Por otro lado, también debemos tener en cuenta antes de diseñar una red que pretenda ser segura, que existen algunas topologías más vulnerables que otras. Nuestra elección deberá estar condicionada por estas consideraciones y por los requerimientos de operatividad. Habrá otros aspectos de seguridad básica de computadoras que examinaremos más tarde, como las contraseñas de la BIOS y la seguridad del hardware.

2.1. Ubicación y acceso físico al sistema

La importancia de dónde estén ubicados tiene su fundamento no sólo en los posibles ataques que podemos recibir de personas externas a nuestra organización; un empleado descontento puede llegar a ser más dañino que el más cruel de los crackers. Pero no sólo debemos protegernos de los usuarios malintencionados. El mismo personal de limpieza podría desenchufar un servidor para conectar su aspiradora o poner un teléfono a cargar. Es por ello que debemos tomarnos muy en serio la tarea de asegurar nuestros servidores frente a ataques físicos, intencionados o no.

En una situación óptima, sólo el administrador de un sistema tendría acceso físico a sus servidores, y estos estarían situados en las plantas más altas de los edificios. Con ello, conseguiríamos tanto que eventuales trabajadores malintencionados pudieran llevar a cabo sus ataques físicos, como proteger los servidores frente a, por qué no, una inundación. Al mismo tiempo estaríamos dificultando el acceso a los mismos a cualquier persona ajena a la organización.

Teniendo en cuenta lo anterior, sería una buena política colocar nuestros servidores en una habitación sin ventanas con una puerta sin cristales, que impida ver la actividad en el interior a quien pase por los alrededores, y con un control de acceso estricto. Lo mejor sería que dicho control de acceso no estuviera implementado mediante una simple autenticación por contraseña. Sería deseable que la identificación a la entrada estuviera autenticada, además de por contraseña, mediante algún control biométrico, por ejemplo por huella dactilar, análisis de retina, voz, o incluso alguna combinación de ellos. Dependiendo del valor de lo que queramos asegurar deberemos endurecer más o menos las medidas de seguridad en el acceso a los servidores. Y por supuesto, debemos dotar al sistema de control de acceso de capacidad de registro, de forma que en caso de que se produzca un ataque, podamos seguir las pistas para descubrir al culpable o bien poner un parche al agujero de seguridad que haya permitido que se produzca el ataque.

Si se trata de diseñar un sistema para una empresa u organización de tamaño medio o grande, quizás sea una buena práctica tomar ejemplo de los proveedores de servicios de

internet y crear un **centro de operaciones de red** (NOC, Network Operation Center). Entre las características que debe poseer (en general, dependiendo del valor de lo que aseguremos, todos los servidores deberían localizarse en lugares con características similares) se encuentran:

- Debe encontrarse en un espacio alejado del público, y si es posible, alejado de la oficina.

- La sala y los pasillos que a ella conduzcan deben ser totalmente opacos.

- Las puertas de acceso deben tener un blindaje que incluya el cerco de la puerta, para que no baste con forzar la cerradura.

- Si se emplea vigilancia por circuito cerrado de televisión, se debe enviar la señal hacia un VCR remoto. De esta forma, los asaltantes eventuales no podrán llevarse las pruebas de su delito.

- Los dispositivos de almacenamiento se deben guardar en un lugar seguro, si puede ser, en otro lugar.

Entre las normas de seguridad más comúnmente impuestas para el acceso a las dependencias del centro de operaciones de la red se encuentran:

- Los empleados son responsables de no entrar en áreas donde se almacena información administrativa, a menos que sean autorizados expresamente para ello.

- Los encargados de personal deben asegurar que aquellos empleados que voluntariamente abandonen su empleo, devuelvan sus llaves o tarjetas de acceso el último día de trabajo en la compañía.

- Los empleados que sean despedidos de la compañía, deben devolver sus llaves o tarjetas en el momento en que son informados de su despido; si esto no ocurriese así, las tarjetas de acceso deben ser inmediatamente canceladas y las áreas controladas por las llaves que puedan estar en sus manos deben someterse a un cambio de cerradura.

- El acceso autorizado de cualquier persona a cualquier información, no es para eliminar ninguna grabación oficial

o informe (o una copia) de la oficina donde se guarda, excepto para ejercer sus responsabilidades en el trabajo.

- El acceso al NOC debe estar restringido a los responsables de su operación y mantenimiento.

- Ningún individuo está autorizado a entrar en el NOC, a no ser bajo la inmediata supervisión de un miembro del equipo de operación y mantenimiento del NOC.

- En ningún caso los medios de acceso (llaves, tarjetas o combinaciones) serán prestados o proporcionados a otros.

- Las instalaciones del NOC proveerán e implementarán unas razonables medidas de seguridad para proteger el sistema contra desastres naturales, accidentes e intentos deliberados de dañarlo.

- Las instalaciones del NOC contarán con las medidas necesarias para proveer una protección razonable contra las circunstancias del entorno, como inundaciones, rayos, temperaturas extremas, y pérdidas o fluctuaciones del suministro eléctrico, así como sabotajes, y contará con procedimientos y planes de recuperación de desastres para los datos críticos del sistema.

- Los servicios del sistema pueden ser asegurados fuera de las horas de apertura a través de la instalación de sistemas de alarmas basadas en la detección de movimientos que deben conectarse directamente a una central de alarmas, en la que sea monitorizado.

2.2. Seguridad y topologías de la red

La topología que adopte una red concreta va a definir la forma en que se interconectan los distintos componentes de la misma y, por tanto, la forma en que la información fluye por ella. Este hecho tiene una importante influencia en la seguridad de la red y en la privacidad de las conexiones en las que tiene lugar. Por lo tanto, examinaremos las topologías de red más comunes en cuanto a: puntos de fallo, susceptibilidad a escuchas electrónicas y robustez o tolerancia a fallos.

Topología en bus

Esta topología se caracteriza por estructurar la red conectando todos los dispositivos a un mismo medio físico, que llamamos bus. De esta forma, como todos los equipos leen y escriben en el mismo medio, debe existir algún método para resolver las contiendas que se producirán al intentar escribir en el bus dos o más dispositivos al mismo tiempo.

Los puntos de fallo de esta topología son, por un lado el bus (el medio físico) y el servidor de la red. El bus es lógicamente un punto de fallo, ya que si, por ejemplo, es cortado, en muchos casos será imposible la comunicación de cualquier par de equipos. Esto se debe a que en los buses se ajustan las impedancias de los cables y las resistencias terminales de forma que no existan reflexiones indeseadas. Si el cable es cortado, el esquema de impedancias cambiará totalmente, con lo que también las señales reflejadas. Si en cambio es el servidor el que falla, en principio, el resto de dispositivos de la red podrán seguir comunicandose entre ellos, pero, evidentemente, no podrán acceder al servidor.

Por otro lado, la topología bus lleva inherentemente asociado el peligro de escuchas electrónicas o sniffing en la red. Esto era de esperar, ya que al ser compartido el medio de comunicación, todos los equipos pueden leer los datos escritos en el bus, independientemente de que sean o no los destinatarios. Por tanto, si queremos dotar de seguridad estructural en cuanto a la protección de los datos en tránsito dentro del área local de la red, evitaremos esta topología.

La tolerancia a fallos que presentan las redes con topología en bus es, por contra, alta; a pesar de que muchas estaciones de trabajo de la red caigan, la red seguirá activa.

Topología en anillo

Las redes con topología en anillo tienen cierta similitud con los estructuran con los equipos conectados a un medio físico común, esta vez formando un anillo.

Al igual que en la topología en anillo, hay al menos dos puntos de fallo, que son el anillo propiamente dicho y el servidor. Sin embargo, en este caso, cada dispositivo conectado a la red

actúa como repetidor, con lo que si uno de ellos cae, puede comprometerse la operatividad de la red.

En cuanto a la susceptibilidad de escuchas electrónicas, volvemos a estar ante una situación en la que los datos pasan por dispositivos que no son el destinatario de los datos, con lo que no ofrece garantías de privacidad. Sin embargo, este aspecto se mejora sensiblemente respecto a la topología en bus, ya que no siempre los datos deben pasar por todas las estaciones de trabajo de la red para llegar a su destino.

Las redes con topología en anillo no son tolerantes a fallos; simplemente con que fallen un par de estaciones de trabajo, la red entera puede quedar sin operatividad.

Topología en estrella

Las redes organizadas en estrella se caracterizan por la existencia de un dispositivo concentrador. Este dispositivo es normalmente un switch o un hub.

En este tipo de redes, como en todas las redes centralizadas, el puntos de fallo es el dispositivo concentrador. Si éste falla, se rompe la comunicación entre todos los dispositivos de la red.

En cuanto la tolerancia a fallos, la red puede seguir funcionando a pesar de que fallen distintas estaciones de trabajo. Sin embargo, si falla el dispositivo concentrador, la red deja de estar operativa.

En cuanto a la privacidad de las comunicaciones, si el dispositivo concentrador es un switch en lugar de un hub de acceso por contienda, las conexiones estarán separadas y no será posible llevar a cabo técnicas de sniffing.

2.3. Hardware de red

Uno de los puntos vitales de la seguridad de los sistemas de comunicación es el hardware de red. Nos referimos a dispositivos tales como routers, bridges, switches o hubs. Un

fallo en uno de estos dispositivos normalmente incapacita las comunicaciones del segmento de red que de él cuelgue.

Normalmente los problemas de seguridad derivados del hardware de red son provocados por una incorrecta administración. Muchos dispositivos de red, proporcionan algún servicio para su configuración. Si no definimos contraseñas de administración y mantenimiento robustas, distintas de otras contraseñas de la red, estaremos poniendo en peligro la configuración de nuestros servicios de red. Muchos de los dispositivos de red cuentan con cuentas de administración y contraseñas bien conocidas entre los agresores; otros, presentan una configuración predeterminada demasiado permisiva. Debemos asegurarnos de configurar adecuadamente nuestro hardware de red, eliminar servicios innecesarios (en la mayoría de los casos, el servicio de administración remota por telnet será prescindible) y establecer contraseñas seguras. Por otro lado, muchos routers y switches están dotados con una opción de cifrado de los datos; debemos también comprobarlo y activarlo en caso de que sea posible.

A continuación se muestran algunos problemas conocidos de varios dispositivos de red:

Switches 3Com:	El acceso mediante usuario "debug" y contraseña "synnet" para el mantenimiento son muy conocidos. Esto afecta entre otros, a los modelos CoreBuilder y SuperStack II.
Motorola CableRouter:	Los productos de la serie CableRouter de Motorola permiten el acceso mediante telnet al puerto 1024 con el usuario "login" y contraseña "router".
Ericsson Tigris:	Algunos routers antiguos de la serie Tigris permiten que usuarios remotos envíen comandos no autenticados. Esto se corrigió a partir de la versión 11.1.23.3 del firmware.

Muchos dispositivos de red que no se han mencionado aquí presentan problemas similares. Es preciso por tanto, mantenerse al día en las actualizaciones del firmware de los equipos, y es también recomendable suscribirse a las listas de correo para seguridad, para conocer con rapidez los problemas que salen a

la luz.

2.4. Seguridad de las estaciones de trabajo

Una vez tratadas tanto la ubicación física de los servidores del sistema y el acceso a ellos, como la relación entre las topologías de red más habituales y la seguridad, debemos ocuparnos de las estaciones de trabajo. A ellas accederán los usuarios, por lo que debemos garantizar que los privilegios que de ellas se obtengan no comprometan la seguridad del sistema.

BIOS y seguridad

Cuando se arranca una computadora, el procesador busca en la parte final de la memoria del sistema la BIOS (Basic Input/Output System) y la ejecuta. El programa de interfaz de la BIOS está escrito en una memoria de solo lectura. La BIOS proporciona la interfaz de más bajo nivel con los dispositivos periféricos y controla los primeros pasos del proceso de arranque.

La BIOS examina el sistema, busca y chequea periféricos y por último busca una unidad para arrancar el sistema. Normalmente se busca el sistema operativo en la unidad de disco flexible o en la unidad de CD-ROM (si existen), y después en la unidad de disco duro. Esta secuencia se puede normalmente cambiar en la interfaz de configuración de la BIOS. Si el disco duro contiene un sistema arrancable, se busca el MBR (Master Boot Record), que comienza en el primer sector de la primera unidad de disco fijo, se carga su contenido en la memoria y se le pasa el control.

Es evidente que deberemos proteger la información de configuración de nuestro sistema, ya que de ella depende que el arranque de las estaciones de trabajo se produzca, y que éste se produzca como nosotros hemos determinado. Por esta razón debemos proteger el acceso a la interfaz de la BIOS, para que los usuarios no autorizados no puedan cambiar la configuración de nuestros equipos y/o activar hardware o servicios no deseados.

La primera medida de seguridad que podemos tomar para evitar accesos no deseados a la estación de trabajo es protegerla con una contraseña BIOS. No debemos confundir esta contraseña, gestionada por la BIOS, de acceso a la estación de trabajo con la contraseña de acceso a la interfaz de la BIOS ni con las contraseñas de acceso gestionadas por el sistema operativo. La contraseña de acceso a la configuración de los parámetros de la BIOS nos permitirá cambiar la configuración de la estación de trabajo.

Muchas BIOS antiguas solían permitir contraseñas predeterminadas como forma de acceso a la BIOS. Es decir, era posible acceder a la interfaz BIOS introduciendo como contraseña la estipulada por el fabricante como predeterminada. Ejemplos de estas puertas traseras son "AMI" y "AMI_SW" en algunas BIOS fabricadas por American Megatrends, "Award" y "AWARD_SW" en las fabricadas por Award, etc.

Los sistemas operativos Linux se caracterizan, entre otras cosas, por ser capaces de adaptarse al hardware en el que se instalan. Este hecho ha posibilitado que muchos ordenadores que se han quedado anticuados, se dediquen a proporcionar servicios de red basados en Linux. Por ello, debemos prestar especial atención en este tipo de equipos, ya que posiblemente tengan chips con BIOS antiguas y permitan el acceso mediante contraseñas predeterminadas.

La BIOS también permite configurar un equipo de forma que el hardware presente sea o no accesible. De esta forma, aunque una estación de trabajo tenga instalado un disco duro, éste no será accesible a menos que en la BIOS así lo estipulemos. Deberemos usar esta facilidad para deshabilitar todo aquello que no sea estrictamente necesario para que el usuario pueda desarrollar su trabajo. De esta forma, por ejemplo, si el equipo dispone de unidad de discos blandos, deberemos deshabilitarla, para evitar que se pueda arrancar desde ella con otro sistema operativo. Lo mismo podemos decir de las unidades de CD-ROM. Otra solución posible en sistemas cuya BIOS lo permita, es limitando la secuencia de búsqueda de sistema operativo al disco duro local. Por otro lado, podemos pensar en deshabilitar los puertos series, para evitar la posible conexión de un modem, y los puertos paralelos, para evitar la conexión de un disco duro externo, unidad ZIP, o CD-ROM. Lo mismo deberemos hacer con los puertos USB.

Por otro lado, es posible mediante el uso de cierto tipo de

software, recuperar contraseñas de acceso a la BIOS, o provocar un reset por software de la misma. Esto sin mencionar la posibilidad que ofrecen casi todas las placas base de resetear la BIOS mediante un jumper o provocando un cortocircuito. Por todo ello, debemos pensar en la BIOS como una primera línea de defensa en la seguridad de nuestro sistema y tener siempre en cuenta en la configuración de la red y los servicios que ésta presta, la política del menor privilegio.

Cargadores de arranque y LILO

Si tenemos un sistema Linux, una vez se ha cargado el MBR en memoria y se está ejecutando su código, éste busca la primera partición activa y lee el contenido del PBR (Partition Boot Record); éste contendrá información sobre como cargar y ejecutar LILO (LInux LOader). Entonces, el MBR cargará LILO y éste tomará el control del proceso (si LILO se instaló en el MBR).

LILO (Linux Loader) es el encargado de cargar el sistema operativo en memoria y pasarle información para su inicio. A su vez, se le pueden pasar parámetros a LILO para modificar su comportamiento. Por ejemplo, si alguien en el indicador de LILO añade "init single", el sistema se inicia en modo monousuario y proporciona una shell de root sin contraseña. Para evitar que esto se produzca, se puede utilizar el parámetro "restricted" en el fichero de configuración de LILO (habitualmente /etc/lilo.conf). Este parámetro permite iniciar normalmente el sistema, salvo en el caso de que se hayan incluido argumentos en la llamada a LILO, que solicita una clave. Esto proporciona un nivel de seguridad razonable: permite iniciar el sistema, pero no manipular el arranque. Si existen mayores necesidades de seguridad se puede incluir la opción "password", para que LILO solicite una clave antes de iniciar el sistema.

El cargador de arranque LILO puede ser configurado con el archivo lilo.conf (ubicado normalmente en /etc/lilo.conf). Por defecto, la distribución instalará LILO con una configuración estándar. Deberemos modificar el archivo lilo.conf para dotar al sistema de una mayor seguridad. Es conveniente por tanto, añadir las siguientes líneas al siguiente archivo lilo.conf:

```
timeout=00
```

Con esta línea indicaremos a LILO que no espere a que el

usuario introduzca una entrada para arrancar el sistema. Esta línea deberá aparecer a menos que existan más sistemas operativos en el sistema.

```
restricted
```

Esta opción pedirá una contraseña (la indicada en la opción "password") al usuario solo si se especifican parámetros en línea de comandos, como por ejemplo "init linux single", para iniciar en modo monousuario.

```
password=<password>
```

Esta opción especifica la contraseña que deberá introducirse cuando el usuario intente iniciar el sistema en modo monousuario. Al incluir esta línea debemos asegurarnos de que el archivo lilo.conf no podrá ser leído más que por el usuario root, ya que la contraseña aparecerá sin encriptar.

Una vez hechas las modificaciones en el archivo lilo.conf, y para que éstas tengan efecto, deberemos ejecutar la siguiente línea de comando:

```
/sbin/lilo
```

Una medida de seguridad adicional que se debe tomar hechos estos cambios en el archivo de configuración de LILO, es hacerlo inmutable, de forma que evitaremos cambios accidentales o por cualquier razón. Para ello, haremos:

```
chattr +i /etc/lilo.conf
```

De esta forma, si queremos cambiar el archivo de configuración, deberemos primero quitar la propiedad de inmutable y después modificarlo. Esta es una buena práctica para todos los archivos de configuración del sistema vitales.

Bloqueo de consola

En los entornos Unix es conocido el truco de ejecutar en una terminal, que alguien ha dejado inocentemente abierto, un programa que simule la pantalla de presentación al sistema. Muy probablemente, un usuario introducirá su nombre y contraseña, y estos se almacenarán o enviarán al atacante para que este tenga una primera vía de acceso al sistema.

Para evitar situaciones como la anterior, debe obligarse a los usuarios del sistema a que si se alejan de sus máquinas, bloqueen sus consolas para que nadie pueda manipularla o mirar su trabajo. Para ello usaremos xlock y vlock.

El programa xlock bloquea la pantalla cuando nos encontramos en modo gráfico. Está incluido en la mayoría de las distribuciones Linux que soportan X. En general puede ejecutar xlock desde cualquier terminal de consola (xterm) y bloqueará la pantalla de forma que necesitará su clave para desbloquearla.

El programa vlock permite bloquear alguna o todas las consolas virtuales de su máquina Linux. Puede bloquear sólo aquella en la que se está trabajando o todas. Si sólo se bloquea una, las otras se pueden abrir y utilizar la consola, pero no se podrá usar su vty hasta que no la desbloquee.

Sin embargo, bloquear una consola prevendrá que alguien manipule el trabajo de otros, o que haga uso de sus privilegios, pero no previene de reinicios de la máquina u otras formas de deteriorar el funcionamiento del sistema.

A pesar de que las medidas anteriores, una correcta configuración tanto del sistema en la BIOS como de LILO y el bloqueo de consola, pueden llegar a dificultar el acceso no autorizado a las estaciones de trabajo, no debe pensarse que con ello queda solucionado el problema de la seguridad de las estaciones. Estas medidas deben considerarse como una línea más de defensa contra accesos físicos no autorizados.

Capítulo 3

SEGURIDAD DE LAS CONTRASEÑAS

En el capítulo dedicado a seguridad física se indicaban algunos métodos de autenticación: basados en características biométricas (retina, voz, huella dactilar, geometría de la mano, etc), otros en la posesión de algún objeto como una tarjeta de acceso, etc. El modelo de autenticación más sencillo que podemos encontrar es el basado en contraseña. Según este modelo, cuando un usuario intenta acceder al sistema o a alguno de sus recursos, se decide si es quien dice ser en función de que introduzca o no una contraseña previamente acordada entre el sistema y el usuario. El modelo basado en contraseña es el más extendido de todos los métodos de autenticación, dada su facilidad de implementación y bajo coste. Por tanto, nos ocuparemos en este capítulo de estudiar los puntos débiles de este modelo y de cómo podemos mejorar su eficacia.

3.1. Autenticación en Linux

Una vez efectuada una instalación segura del hardware y del sistema operativo Linux en nuestra computadora, incluyendo un correcto particionado de los discos, debemos crear los usuarios y grupos para nuestro sistema. Aquellos accederán a éste mediante su identificación con el nombre de usuario y contraseña. En adelante nos referiremos a este proceso como "autenticación". Debemos observar aquí, que este proceso es el que típicamente se realiza, en contraposición con un proceso de identificación. La diferencia entre ambos estriba en lo siguiente: en un proceso de identificación, el sistema debe ser capaz de identificar al usuario sin que éste haga nada, y después asignarle o no privilegios; en que en el proceso de autenticación, al intentar un usuario acceder al sistema, éste le pregunta a aquel sobre su identidad, y después, comprueba mediante algún método que "el usuario es realmente quien dice ser". Este punto

es, por tanto, uno de los más importantes a la hora de asegurar un sistema con autenticación basada en contraseñas; si las contraseñas de los usuarios son débiles, nuestro sistema será débil, ya que cualquiera que rompa una contraseña tendrá un primer acceso al él. Si el sistema tiene un agujero, por mínimo que sea, y el atacante tiene la habilidad suficiente, la seguridad de todo el sistema estará comprometida.

Para evitar situaciones derivadas de lo anterior, trataremos en este capítulo los procedimientos más comunes para realizar ataques a contraseña y las formas de evitar que tengan éxito.

3.2. Ataques a contraseña

Entendemos por ataque a contraseña como todas aquellas acciones encaminadas a descifrar o borrar contraseñas, así como a evitar los mecanismos de seguridad implementados mediante dichas contraseñas.

Normalmente, un ataque a contraseña exitoso pone en peligro a todo el sistema, con lo cual, a pesar de la sencillez de ejecución de este tipo de ataques, no debemos menospreciar su potencia.

Existen programas de libre distribución que realizan ataques a las contraseñas del sistema mediante la comparación con palabras de un diccionario (archivo que consiste en una larga lista de palabras). En las comparaciones que realizan, aplican diversas reglas, como verificar las palabras escritas hacia atrás, mezcladas con números, etc. Estas herramientas, además de constituir un peligro, pueden ayudar a reforzar la seguridad de las contraseñas del sistema, mediante la verificación periodica de las mismas. Si muchas contraseñas son adivinadas por este tipo de programas, éstas no son seguras. Dos de las aplicaciones más populares para los ataques a contraseñas son Crack y John the Ripper.

3.3. La importancia de contraseñas robustas

Las contraseñas son las llaves que abren la puerta principal de los sistemas. Es obvio que, por tanto, deben ser tan seguras

como sea posible para prevenir accesos no autorizados. Los accesos no autorizados son el primer paso hacia problemas de seguridad mucho mayores. El uso de contraseñas robustas y difíciles de romper será una simple (pero efectiva) medida para evitar problemas en el futuro.

Muchos usuarios usan contraseñas muy fáciles de adivinar. A pesar de que el sistema de autenticación que hayamos escogido sea teóricamente muy seguro (contraseñas MD5, suite Shadow, Kerberos, etc), gran parte de esa seguridad dependerá directamente de la complejidad de las contraseñas escogidas por los usuarios. El hecho de que cada vez los equipos sean más potentes y baratos, unido al aumento del número de programas de libre distribución dedicados a romper contraseñas no es más que una razón cada vez más convincente para dediquemos un esfuerzo a la elección de contraseñas robustas.

Debido a la forma en que las contraseñas se almacenan en muchos de los esquemas de autenticación más simples, si un atacante tiene acceso al archivo que contiene las contraseñas de los usuarios del sistema, podrá adivinar alguna de ellas en un relativamente corto espacio de tiempo, mediante la comparación de las contraseñas encriptadas con palabras también encriptadas, tomadas de un archivo diccionario.

A pesar de que los esfuerzos en desarrollar esquemas de autenticación que minimicen el éxito de este tipo de ataques son grandes, ninguno de ellos es infalible. Por tanto, debemos preocuparnos de escoger contraseñas robustas y de cambiarlas a menudo.

3.4. El Sistema de Contraseñas de Linux

El archivo `/etc/passwd`

Tradicionalmente, en los sistemas tipo Unix, las contraseñas se han almacenado en el archivo `/etc/passwd`, con la característica de que dicho archivo debe ser legible por todos. Por tanto, un simple vistazo al archivo puede presentar algo

similar a lo siguiente:

```
root:$1$XvwNGizW$GYQ47yupFvrwD4P9i4wnm.:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/var/ftp:
nobody:*:99:99:Nobody:/:
nscd:!!:28:28:NSCD Daemon:/:/bin/false
mailnull:!!:47:47::/var/spool/mqueue:/dev/null
ident:!!:98:98:pident user:/:/bin/false
rpc:!!:32:32:Portmapper RPC user:/:/bin/false
rpcuser:!!:29:29:RPC Service User:/var/lib/nfs:/bin/false
xfs:!!:43:43:X Font Server:/etc/X11/fs:/bin/false
gdm:!!:42:42::/home/gdm:/bin/bash
postgres:!!:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
apache:!!:48:48:Apache:/var/www:/bin/false
named:!!:25:25:Named:/var/named:/bin/false
amanda:!!:33:6:Amanda user:/var/lib/amanda:/bin/bash
mysql:!!:27:27:MySQL Server:/var/lib/mysql:/bin/bash
ldap:!!:55:55:LDAP User:/var/lib/ldap:/bin/false
pvm:!!:24:24::/usr/share/pvm3:/bin/bash
squid:!!:23:23::/var/spool/squid:/dev/null
antonio:$1$dxmNwVwp$PI88PykZMTWPvt1FY2bs8.:500:500::/home/antonio:/bin/bash
anger:$1$HpF3qE9.$OmKNtKNCx6SWWqy4o6vwf/:501:501::/home/anger:/bin/bash
incauto:$1$cef3R4DA$7zNF50Tnn8U442HjC0RRG0:502:502::/home/incauto:/bin/bash
```

En este archivo, cada linea corresponde a un usuario del sistema. Dentro de cada linea, existen distintos campos, todos separados por ‘:’. Los campos, de izquierda a derecha, son: nombre de usuario, contraseña, UID, GID, campo GECOS, directorio personal, login shell.

Los campos UID y GID son, respectivamente, la identificación de usuario y grupo. En un sistema Unix se identifica a un usuario mediante su UID, siendo el nombre de usuario o login una simple asociación a un UID, ya que es más fácil recordar un nombre de usuario que un número (esto toma mayor importancia cuando se es usuario de varias máquinas y en cada una se tiene un UID distinto). De la misma forma, un grupo se identifica por su GID.

El campo GECOS (General Electric Comprehensive Operating System) contiene información relativa al usuario, como el nombre real, dirección, correo electrónico, etc. El shell del usuario debe estar listado en el archivo /etc/shells para que el usuario pueda hacer login.

Hasta hace relativamente poco tiempo, en el campo de contraseña se almacenaba directamente la contraseña encriptada. El método que se usaba para encriptar las contraseñas Linux estaba basado en el DES (Data Encryption Standard). El Data Encryption Standard es un cifrado de bloque, es decir, trabaja sobre bloques de datos de un tamaño determinado. Las funciones de cifrado y descifrado de DES se basan en una clave, mediante la que los usuarios que no son destinatarios de la comunicación no podrán descifrar los datos. El algoritmo usado para encriptar la contraseña se conoce como "one way hash function". Es un algoritmo fácil de computar en una dirección, pero difícil de calcular en la dirección inversa.

El hecho de que en el archivo /etc/passwd se almacenasen directamente las contraseñas encriptadas era un serio problema de seguridad, ya que este archivo debe ser legible por todo el mundo, o de lo contrario, muchos comandos dejarían de funcionar correctamente.

Sin embargo, si instalamos un sistema de shadowing de contraseñas, por ejemplo Linux Password Shadow Suite, éstas se almacenarán en el archivo /etc/shadow, y en el campo de contraseña del archivo /etc/passwd aparecerá un enlace a la contraseña en el archivo /etc/shadow. La suite Shadow implementa además nuevos conceptos relativos a la gestión de contraseñas, como son el vencimiento de la contraseña y el bloqueo automático de la cuenta. A continuación se lista un archivo típico /etc/shadow:

```
root:$1$KdJ631Vq$2L48QD40Bm1BV.ynkE/Wv/:12169:0:99999:7:::
bin:*:12169:0:99999:7:::
daemon:*:12169:0:99999:7:::
adm:*:12169:0:99999:7:::
lp:*:12169:0:99999:7:::
sync:*:12169:0:99999:7:::
shutdown:*:12169:0:99999:7:::
halt:*:12169:0:99999:7:::
mail:*:12169:0:99999:7:::
news:*:12169:0:99999:7:::
uucp:*:12169:0:99999:7:::
operator:*:12169:0:99999:7:::
games:*:12169:0:99999:7:::
nobody:*:12169:0:99999:7:::
rpm:!!:12169:0:99999:7:::
vcsa:!!:12169:0:99999:7:::
rpc:!!:12169:0:99999:7:::
xfs:!!:12169:0:99999:7:::
apache:!!:12169:0:99999:7:::
postfix:!!:12169:0:99999:7:::
named:!!:12169:0:99999:7:::
rpcuser:!!:12169:0:99999:7:::
sshd:!!:12169:0:99999:7:::
mysql:!!:12169:0:99999:7:::
gdm:!!:12169:0:99999:7:::
ftp:!!:12169:0:99999:7:::
postgres:!!:12169:0:99999:7:::
antonio:$1$zRlQutdA$zssrSG5ho2zb0uWs6nflU/:12169:0:99999:7:::
saned:!!:12170:0:99999:7:::
```

```
nychy:$1$8aRLwlUA$c23hfX/LpXA8k1wPG3E6n/:12173:0:99999:7:::
```

Podemos observar que el formato del archivo /etc/shadow es parecido al de /etc/passwd. Los campos que contiene cada entrada (cada fila) son:

username:passwd:last:may:must:warn:expire:disable:reserved

El campo username es el nombre de usuario o login. Al igual que en el archivo /etc/passwd, se asocia a un UID o identificador de usuario. El campo passwd almacena la contraseña encriptada. El campo last indica el número de días que han pasado hasta que la contraseña fue cambiada por última vez, contados desde el 1 de enero de 1970. El campo may indica el número de días antes de que la contraseña pueda ser cambiada, y must el número de días después de los que la contraseña debe ser cambiada. El campo warn indica los días antes de que se avise al usuario de que la contraseña va a expirar. Por último, el campo expire indica el numero de días desde el 1 de enero de 1970 hasta que la contraseña expire. El campo reserved está reservado para uso futuro.

El shadowing de contraseñas proporciona un nivel más alto que el almacenamiento tradicional de contraseñas. Hemos dicho más arriba que el archivo /etc/passwd debe ser legible por todo el mundo para que los programas funcionen correctamente, y que esto constituía un riesgo para la seguridad de las contraseñas del sistema. Con el shadowing de contraseñas, dicho fichero sigue siendo legible por todo el mundo, pero en los campos de contraseña aparecen enlaces a las contraseñas en el archivo /etc/shadow, el cual es solo legible por el root. A continuación vemos los permisos típicos de ambos archivos:

```
-r--------    1 root     root          916 may  1 12:48 shadow
-rw-r--r--    1 root     root         1472 may  1 12:49 passwd
```

3.5. Comprobación proactiva de contraseñas

Una buena solución a adoptar para asegurar que las contraseñas de los usuarios no utilizan contraseñas débiles es la comprobación proactiva de contraseñas. Esta técnica consiste en verificar que la contraseña introducida por el usuario en el momento de creación o cambio de la misma no es débil. Para ello se compara la entrada con una lista de palabras (diccionario), a las cuales se les aplican una serie de reglas. Si se encuentra alguna coincidencia se decide que la contraseña introducida es débil y se pide al usuario que elija otra. Existen diversos paquetes de comprobación proactiva de contraseñas, entre los que se encuentran passwd+ y npasswd. La comprobación proactiva de contraseñas se puede ver como un ataque a diccionario previo a la introducción de la contraseña en la base de datos.

passwd+

Este paquete ofrece una gran capacidad de registro. De esta forma, quedarán registradas las sesiones, los errores que se produzcan en la introducción de contraseñas, los cambios de contraseña, las reglas que hayan verificado las contraseñas introducidas por los usuarios a la hora de la comprobación, etc. Además, es posible especificar el número de caracteres de la contraseñas que se usarán para la comprobación. Passwd+ permite también especificar el mensaje de error que se enviará al usuario cuando introduzca una clave errónea, de modo que los usuarios puedan conocer las razones por las que la contraseña elegida es débil y aprender a elegir opciones más robustas.

Entre las reglas que usa passwd+ para la comprobación se encuentran:

- Número o teléfono de la oficina, el dominio o el nombre del host no están permitidos.

- Se deben mezclar mayúsculas y minúsculas.

- Las palabras de diccionario están prohibidas.

- El nombre y los apellidos (también escritos hacia atrás) están prohibidos.

Por otro lado, el creador de passwd+, desarrolló un lenguaje para controlar muchos aspectos de las contraseñas, así como las reglas a las que se exponen en la comprobación proactiva.

anlpasswd

Este comprobador de contraseñas permite usar el archivo de diccionario que se escoja y construir reglas de comprobación personalizadas. Entre las reglas predeterminadas que se usan en las comprobaciones están:

- Números con espacios y espacios con números.

- Mayúsculas y minúsculas con espacios.

- Todo en mayúsculas o todo en minúsculas.

- Todo números.

- La primera letra en mayúsculas y números.

- Todo números.

- La primera letra mayúscula y números.

- Todas las combinaciones de las anteriores.

3.6. Pluggable Authentication Modules

Muchas distribuciones de Linux, incluyendo Red Hat y Debian, han incorporado un esquema unificado de autenticación llamado PAM (Pluggable Authentication Modules). Este esquema permite al administrador del sistema cambiar la forma en que las

aplicaciones autentican a los usuarios al momento y unificar todos los métodos locales de autenticación sin recompilar nada.

Los Pluggable Authentication Modules son un conjunto de librerías compartidas que permiten usar métodos de encriptado de contraseñas distintos a DES, haciendolas más difíciles de adivinar en un ataque por fuerza bruta. Por otro lado, también permiten asignar límites a los usuarios en el acceso a los recursos, de forma que no puedan ejecutar ataques de negación de servicio; se puede especificar el número máximo de procesos, cantidad máxima de memoria, etc. Además, podremos especificar qué usuarios pueden acceder al sistema en cada instante y desde qué sitios pueden hacerlo.

Con el método de autenticación tradicional, la asignación de privilegios se basa en el esquema de autenticación mediante la base de datos de contraseñas (/etc/passwd o /etc/shadow). De esta forma, un usuario tendrá los privilegios asociados a su UID (identificador de usuario) o GID (identificador de grupo). Según sus creadores, el objetivo de PAM es separar el desarrollo de software asignación de privilegios del desarrollo de esquemas de autenticación seguros y apropiados. De esta forma, el administrador puede escoger un esquema tan simple como el de autenticar mediante contraseña sin encriptar, o el extremo opuesto de autenticación mediante la combinación de un escáner de retina, el análisis de la voz y una contraseña de un solo uso.

PAM trata con cuatro tipos separados de tareas: gestión de autenticación, gestión de cuenta, gestión de sesión y gestión de contraseña. La asociación entre el esquema de gestión deseado y el comportamiento de las aplicaciones se realiza en el archivo de configuración de Linux-PAM.

3.7. Reglas básicas para escoger una contraseña

Una buena contraseña debe reunir las siguientes características:

- Contiene al menos 8 caracteres. Mientras más corta sea, más fácil de romper será.

- Contiene caracteres alfabéticos, numéricos y símbolos. Los números y símbolos mezclados con las letras aumentan considerablemente el número de posibilidades para cada carácter de la contraseña, lo cual la hace más fuerte.

- Es única. Se deben seleccionar contraseñas distintas de otras que pueda poseer el usuario. Si todas las contraseñas son iguales o muy parecidas, la magnitud del agujero de seguridad, caso de ser rota una de ellas, será mucho mayor.

Se debe evitar el uso de contraseñas con las siguientes características:

- Son palabras de diccionario o derivadas, de cualquier idioma. Aquí se incluyen los nombres geográficos y mitológicos y las variaciones. Al usar palabras de diccionario para elegir una contraseña se multiplica exponencialmente el riesgo de ser rota mediante un ataque por fuerza bruta.

- Están relacionadas con la información personal del usuario. El uso de contraseñas que contienen (o simplemente son) la fecha de cumpleaños, el nombre de la esposa o marido, la matricula del coche, etc, facilita a los atacantes la tarea de romperlas. Al elegir una contraseña se debe pensar en si una persona que nos conozca podría adivinarla, y en caso de que exista el más mínimo indicio de que sí, no usarla y elegir otra.

- No se puede teclear rápidamente. Las contraseñas que son tan complicadas que no pueden introducirse con rapidez, aumentan la probabilidad de que alguien observe mientras el usuario la teclea y adivinarla por la posición de los dedos.

- Contiene palabras que pueden ser encontradas en el mismo archivo de contraseñas (por ejemplo, alguna perteneciente al campo GECOS).

- Contiene patrones como 123456, qwerty, a1b2c3d4, etc.

En resumen, para elegir una buena contraseña, debemos escoger sus caracteres aleatoriamente de entre el conjunto de letras, números y símbolos de forma que su longitud sea al menos de 8 caracteres.

Capítulo 4
ESCUCHAS ELECTRÓNICAS

En las redes de computadores, los canales de comunicación se suelen compartir. Esto se comenzó a hacer así, sencillamente por cuestiones económicas, ya que dedicar un bucle hasta el concentrador para cada par de ordenadores que se comuniquen resultaba demasiado caro.

La forma más popular de realizar lo anterior es a través del protocolo Ethernet de nivel de enlace. En él, los paquetes se envían a un segmento de red, teóricamente al mismo medio al que todos los hosts de ese segmento están conectados. Como la cabecera de cada paquete contiene información sobre la dirección de destino, la interfaz de cada host con la red podrá decidir, examinando la cabecera, si el paquete es o no para ese host. Teóricamente, sólo el host de destino del paquete lo leerá. No obstante, existe un modo de funcionamiento de las interfaces de red en el que pueden leer todo el tráfico que pase por ellas, sin importar la dirección de destino del paquete.

Existen dispositivos, aplicaciones que pueden ser instalados en una red y recoger toda la información que por ella circula. Estos dispositivos se llaman analizadores de protocolo, aunque también se les conoce como *analizadores de protocolo* o *sniffers*. En este capítulo nos dedicaremos a estudiar el funcionamiento de dichos dispositivos, explicaremos los resultados de algunas pruebas hechas en el laboratorio y veremos las formas de defendernos de ellos.

De lo anterior podemos deducir que, el hecho de que una sola máquina del segmento esté ejecutando una herramienta de sniffing es suficiente para comprometer todas las conexiones de la red.

4.1. Funcionamiento de los sniffers

En condiciones normales, las estaciones de trabajo tendrán sus interfaces de red configuradas de forma que éstas solamente leerán el tráfico de la red que tenga como destino la dirección de dicha interfaz. Sin embargo, es posible configurar las interfaces de red en un modo especial, llamado *modo promiscuo* de forma que la interfaz puede examinar todo el tráfico que circula por su segmento de red. En este modo, tenemos la posibilidad de espiar las conexiones del resto de máquinas de la red.

Por otro lado, podemos imaginar una máquina multiusuario a la que se conectan diferentes máquinas. Un sniffer instalado en esta máquina podría no necesitar el inicio de la interfaz en modo promiscuo, ya que por ella pasarían todas las conexiones de las distintas máquinas, y de esta forma podríamos examinarlas. De todas formas, para escuchar todo el tráfico el segmento de red, es imprescindible que la interfaz se encuentre en modo promiscuo.

La mayoría de los sniffers para Linux realizan los ataques a través de tres tareas: inicio del modo promiscuo de la interfaz de red, lectura del tráfico TCP/IP y escritura en un archivo o en la salida estándar. Antes de escribir en un archivo es posible darle a los datos un formato inteligible. Para estas tareas, los sniffers suelen incluir los archivos de cabecera siguientes:

- linux/if.h Contiene definiciones para controlar las interfaces.

- linux/if_ether.h Contiene definiciones para la interfaz IEEE 803.3 de Ethernet y para muchos de sus protocolos, entre ellos, IP, ARP, AppleTalk, IPX, etc.

- linux/in.h Contiene definiciones de estructuras para las direcciones de internet.

- linux/ip.h Es una implementación de IP para Linux.

- sys/socket.h Para gestionar las operaciones con sockets.

- tcp.h Contiene definiciones relativas a las conexiones TCP.

Debe hacerse notar que los analizadores de protocolo no solo funcionan con interfaces de red del tipo Ethernet. También pueden funcionar en redes Token Ring o en conexiones punto a punto. Cualquier interfaz de red que pueda configurarse en modo promiscuo es capaz de realizar escuchas. Este hecho supone un peligro mayor, ya que si alguien es capaz de arrancar un sniffer en la máquina conectada a internet que proporciona el servicio a toda una red local, esta tendrá su privacidad comprometida en todos los accesos al exterior.

También es importante dejar claro que para capturar toda la información que circula por el segmento de red al que está conectada una determinada interfaz, es imprescindible que ésta se encuentre configurada en modo promiscuo. Para ello son necesarios los privilegios del usuario root. No obstante, en máquinas multiusuario, sobre las que trabajan muchos usuarios, aún sin configurar la interfaz en modo promiscuo, podría capturarse el tráfico de red dirigido a esa máquina pero no necesariamente a un solo usuario. Esto es, si un analizador de protocolo está ejecutandose en una máquina multiusuario, todos los usuarios de la misma verían comprometida la privacidad de sus conexiones.

Para evitar los ataques de este tipo se debe asegurar que las interfaces no se encuentran en modo promiscuo. Para comprobar el estado de las interfaces, y si se encuentran en modo promiscuo o no, puede usarse la utilidad ifconfig. Esta utilidad solo la puede ejecutar el usuario root. No obstante, debe se debe prestar atención a la integridad de dicha aplicación (así como la de todas las aplicaciones de sistema), ya que los atacantes podrían instalar una aplicación modificada, que se comportase como la original pero que mostrase que las interfaces están en estado normal (cuando estuviesen en modo promiscuo). Para asegurar que los archivos de sistema no han sido alterados se puede usar alguna herramienta de verificación de integridad, como MD5 o Tripwire.

4.2. Sniffing en helio.us.es

Para comprobar las posibilidades de los analizadores de protocolos o sniffers, hemos realizado varias pruebas con diversos sniffers en la red del laboratorio. Los sniffers que hemos usado son: Sniffit v0.3.5, Hunt v1.5 y Ethereal. Veamos sus características y los resultados que hemos obtenido.

Hunt v1.5

Hunt es una herramienta de sniffing creada por Pavel Krauz. Con esta herramienta, Krauz pretendía explotar las debilidades de la familia de protocolos TCP/IP. Esta aplicación nos permite realizar espionaje de las conexiones y nos proporcionas varias herramientas de *spoofing* (falsificación o suplantación de identidad).

Comenzamos explicando los pasos que debemos dar para instalar el paquete. Después de descargarlo de internet debemos descomprimirlo. Para ello, ejecutaremos el comando:

```
$tar -xvf hunt-1.5bin.tar
```

Hecho esto, los archivos fuente y los binarios compilados deben haberse copiado en un subdirectorio llamado hunt-1.5, por lo que accedemos a él, y ejecutamos el programa escribiendo:

```
$ cd hunt-1.5
$ ./hunt_static
```

Hecho esto debe habernos aparecido en pantalla un menú como mostrado en la figura 4-1.

Hunt es capaz de capturar conexiones que se establezcan o estén establecidas en cualquier par de puertos TCP (vease Pliego de Condiciones, Puertos TCP), aunque está especialmente indicado para conexiones de telnet y rlogin; en general, cualquier conexión no cifrada a una máquina, ya que la presentación en pantalla de la sesión bajo análisis es muy clara e intuitiva. No obstante, Hunt pone a nuestra disposición un demonio capaz de guardar por si sólo las conexiones robadas en archivos, lo que lo hace especialmente atractivo para dejarlo ejecutándose en segundo plano.

Para probar las capacidades de Hunt, vamos a realizar

varias conexiones desde varios host del laboratorio. Mientras tanto, intentaremos capturarlas desde uno de los host del segmento de red. Almacenaremos los resultados en distintos archivos. Para ello, antes debemos configurar los demonios de sniffing que nos ofrece Hunt de forma que capturen aquello en lo que estamos interesados.

Ataque de sniffing a una sesión de telnet

Vamos a ejecutar Hunt en la estación de trabajo de dirección 172.16.17.1, la primera dentro del segmento del laboratorio. Lo primero que debemos hacer es configurar correctamente Hunt para que realice las operaciones que deseamos. En el menú principal que presenta Hunt al ejecutarse, seleccionamos la opción **d**, para configurar los demonios. Los demonios son tremendamente útiles, ya que permiten que especifiquemos aquellas conexiones en lo que estamos interesado y ellos se encargan de vigilar si se producen o no. En caso de que se produzcan, archivarán el registro en el fichero que indiquemos. Seleccionada esta opción, obtendremos lo mostrado en la figura 4-2.

Como por el momento vamos a realizar ataques de sniffing, seleccionamos la opción **s** correspondiente al demonio de sniffing. Aparece lo que se muestra en la figura 4-3.

Antes de arrancar el demonio de sniffing, vamos a configurarlo de forma que capture, en este caso, todas las sesiones de telnet que se produzcan en el segmento, y le indicaremos que las almacene en distintos archivos. Para ello seleccionamos **a** (add sniff item), para añadir un elemento a espiar. El programa nos pregunta por la dirección o mascara y puerto o puertos de origen de la conexión a espiar. Introducimos `0.0.0.0`, con lo que estamos indicando que estamos interesados en todas las conexiones que partan de cualquier dirección IP y cualquier puerto TCP. Lo siguiente que nos pregunta es la dirección o máscara y el puerto o puertos destino. Introducimos `0.0.0.0 23`, con lo que indicamos que registre las conexiones cuyo destino sea cualquier host al puerto TCP número 23 (el correspondiente a telnet). A continuación debemos indicar al programa si debe buscar alguna cadena en la que estemos interesados. En principio no estamos interesados en ninguna cadena en especial, con lo cual respondemos no a esta pregunta. Hecho esto, debemos introducir el tipo de log, es decir, si

queremos hacer registro de lo enviado por la dirección fuente, de lo enviado por la dirección destino, o de lo enviado por ambos. Seleccionamos **b** (both), para registrar lo que transmiten ambos. Ya sólo nos queda especificar la longitud máxima del registro y el nombre del archivo. Nosotros vamos a indicarle al programa que como máximo registre 1000000 bytes. Por defecto, el número de bytes a registrar es 64. La razón de esto es que en las sesiones telnet y rlogin, la información referente al nombre de usuario y la contraseña se encuentra en los 64 primeros bytes de la conexión, con lo cual, si estamos interesados en capturar este tipo de datos, no es necesario registrar más. Como nombre de archivo dejamos el nombre predeterminado de las conexiones, de forma que los archivos se nombran usando las direcciones y puertos origen y destino. Nos pregunta también en qué posición debe insertar el contenido del registro dentro del archivo; dejamos la opción predeterminada (0). Introducidos todos estos datos, la pantalla del terminal debe tener un aspecto parecido al de la figura 4-4.

Ahora ya estamos en condiciones de arrancar el demonio de sniffing. Para ello, seleccionamos **s** (start sniff daemon). En ese momento, Hunt está listo y atento a las conexiones que hemos especificado. Para verlo, nos sentamos en otro ordenador del laboratorio y hacemos telnet a helio.us.es. En esta sesión vamos a movernos un poco por el sistema de archivos y luego terminamos. Veamos cuales han sido los resultados.

```
ÿý&ÿû&ÿý ÿû ÿû ÿû ÿû!ÿû"ÿû'ÿý ÿû#ÿú  P  ÿðÿú
38400,38400ÿðÿú#    localhost.localdomain:0ÿðÿú'
DISPLAY localhost.localdomain:0ÿðÿú XTERMÿðÿü ÿý
antonio
 incauto
 ll
 telnet 172.16.17.2 2
  [A 1
 ll
 ez xit
```

El registro grabado en el archivo de salida tiene un formato que no resulta tan legible como la presentación en la pantalla del terminal que nos ofrece Hunt. Esto se debe a que, además de las direcciones de origen y destino de cada paquete, se incluyen los caracteres de control. Sin embargo se han resaltado el nombre de usuario (antonio) y la contraseña (incauto) capturadas en la sesión. En la utilidad de monitorización en pantalla de conexiones que nos ofrece Hunt (watch connections) se usan dichos

caracteres para dar formato a la presentación en el terminal.

Con este ataque que hemos realizado a una sesión telnet, hemos experimentado lo vulnerable que es este servicio. Su vulnerabilidad parte de que no usa cifrado, y por tanto, si los datos son capturados, son también inteligibles directamente. Por tanto, siempre que queramos conectar de forma remota a otra máquina, intentaremos usar (Secure Shell). Aplicar cifrado a las sesiones no es una medida infalible, pero reduce notablemente la probabilidad de que los datos eventualmente capturados sirvan para algo. Veremos más adelante las comunicaciones seguras con más detalle.

Ataque de sniffing a una conexión a un servidor POP.

En este apartado vamos a realizar un ataque a una conexión a un servidor de correo pop (Post Office Protocol). De nuevo usaremos el host 172.16.17.3 para realizar la conexión, y el host que nos servirá para ejecutar Hunt y capturarla será el 172.16.17.1.

Primero, como en el caso anterior, configuraremos Hunt para que sea el demonio de sniffing el que se ocupe de registrar automáticamente las conexiones al servidor de correo. En el caso anterior, en el que espiábamos una conexión telnet, Hunt detectaba automáticamente ese tipo de conexión (al puerto 23). En este caso, el puerto destino de la conexión es el 110 (servidor pop), por lo que necesitaremos decirle a Hunt que detecte todas las conexiones, o al menos las que vayan al puerto 110. Para ello, procederemos como sigue. En el menú principal de Hunt, seleccionamos **o** (options). El menú que se presenta es el mostrado en la figura 4-5.

Una vez en el menú de opciones, seleccionamos **a** (add conn policy entry). Aquí introducimos los valores predeterminados en dirección fuente (0.0.0.0), dirección destino (0.0.0.0) y en la posición a insertar. Hecho esto, nos salimos hasta el menú principal, en el que seleccionamos **d** (daemons) como en el apartado anterior. En este menú, seleccionamos **s** (sniff daemon) y añadimos una entrada a la lista de conexiones a espiar. Para ello, seleccionamos **a** (add sniff item). Nos pregunta la dirección IP fuente, a lo que responderemos con la dirección predeterminada 0.0.0.0, para registrar las conexiones

que partan de cualquier dirección, y como destino especificaremos 0.0.0.0 110, para indicar que queremos capturar todas las conexiones que vayan al puerto 110 de cualquier máquina. Diremos que no queremos buscar ninguna cadena y que queremos registrar 1000000 bytes como máximo de ambas direcciones de la conexión. El nombre del archivo de registro lo dejaremos como está para usar el nombrado predeterminado (usando las direcciones y puertos fuente y destino). En la figura 4-6 se muestra la pantalla del terminal después de introducir estas opciones.

Hecho esto, sólo queda arrancar el demonio y esperar. Para ello seleccionamos **s** (start daemon). Para ver los resultados solo tenemos que acceder a los archivos de registro, guardados en el directorio oculto `.sniff`.

Sniffit 0.3.5

En este apartado haremos lo mismo que hicimos en el apartado anterior, pero con otra herramienta de análisis de protocolos: Sniffit. Esta herramienta ha sido desarrollada por Brecht Claerhout y proporciona la opción de funcionar en modo interactivo. Veamos cómo funciona.

Lo primero que debemos hacer para instalar Sniffit es descargarlo de la página de su creador. Este proporciona un archivo empaquetado con la utilidad tar

Una vez lo hemos descargado, debemos desempaquetarlo usando tar. Para ello, abrimos un terminal y escribimos el comando:

```
$ tar xvf sniffit.0.3.5.tar
```

Si todo ha ido bien, el programa debe haberse instalado en el subdirectorio `./sniffit.0.3.5`. Dentro de ese subdirectorio están tanto el código fuente de la aplicación como los archivos compilados.

El modo interactivo de Sniffit

Una de las características principales de Sniffit es la posibilidad de ejecutarlo en modo interactivo. De esta forma,

podremos visualizar una lista de todas las conexiones que se estén produciendo en un determinado momento y seleccionar la que más nos interese. Para entrar en el modo interactivo, ejecutamos Sniffit de la siguiente forma:

```
$ ./sniffit -i
```

La pantalla que presenta Sniffit al entrar en el modo interactivo es parecida a la mostrada en la figura 4-7. En ella podemos observar las distintas conexiones que están teniendo lugar en el segmento de red al que pertenece el host en el que se ejecuta Sniffit. En la columna que aparece más a la izquierda figura la fuente de cada conexión, y en la columna de la derecha el destino. Si queremos filtrar las conexiones en función de su dirección y/o puerto fuente y/o destino, podemos hacerlo usando las teclas F1 a F4. Para ver el tráfico de cada conexión solo tenemos que movernos hasta ella con los cursores y pulsar ENTER.

Dentro de la ventana asociada a una conexión concreta se muestran los datos capturados, que pueden ser inteligibles o no. En los casos en los que los datos no estén cifrados, éstos serán inteligibles. En la figura 4-8 se ilustra uno de estos casos, en los que se captura una sesión de MSN Service. Sin embargo, en otros casos, como por ejemplo las sesiones cifradas o las sesiones FTP los datos capturados no son inteligibles. En la figura 4-9 se puede ver la apariencia de la captura de una sesión de FTP.

Pero lo anterior solo muestra una de las capacidades de Sniffit. Lo que realmente le confiere potencia a Sniffit es la opción de ejecutar las opciones tomadas de un archivo de configuración. Es por ello, que en las pruebas realizadas en el laboratorio hemos usado Sniffit con esta opción. Veamos de forma sucinta la estructura de un archivo de configuración para Sniffit.

El archivo de configuración de Sniffit

Para hacer que Sniffit tome los parámetros de un archivo de configuración, debemos especificarlo en la línea de comandos mediante la opción **-c <archivo de configuración>**. Por ejemplo, lo ejecutaremos en una ocasión de la forma siguiente:

```
$ ./sniffit -c sniff_telnet.conf
```

De esta forma, Sniffit tomará los parámetros de ejecución del archivo sniff_telnet.conf.

Por defecto, cuando Sniffit se ejecuta en modo no interactivo, no captura nada; debemos especificar nosotros en el archivo de configuración lo que queremos registrar.

Cada linea del archivo de configuración consta de 4 o 5 campos, siguiendo la estructura:

```
<campo1> <campo2> <campo3> <campo4> [<campo5>]
```

donde:

<campo1> puede ser `select`, con lo que indicamos que se registre lo que se define en los campos siguientes; `deselect`, con lo que indicamos que no se registre lo que se define en los campos siguientes; o `logfile`, para indicar que en <campo2> damos el nombre del archivo de registro.

<campo2> puede ser `from`, con lo que indicamos que nos referimos a las conexiones procedentes de lo que se indica a continuación; `to`, con lo que nos referimos a las conexiones con el destino que se especifica en el resto de los campos; both, en el caso de que nos queramos referir a conexiones cuya fuente o destino se especifican en los siguientes campos; o un nombre de archivo, si especificamos `logfile` en <campo1>

<campo3> puede ser `host`, para indicar que nos referimos a un host, que indicaremos en <campo4>; `port`, con lo que indicamos que estamos especificando un puerto; o `mhost`, para referirnos a múltiples hosts.

<campo4> puede ser un nombre de host, un número de puerto o una dirección múltiple si hemos indicado `mhost` en <campo3>.

<campo5> este campo estará presente si en <campo3> hemos especificado `host` o `mhost`. Corresponderá a un número de puerto.

Ataque de sniffing con Sniffit

Vamos a realizar un sencillo ataque de sniffing en la red del laboratorio. Para ello, como hacíamos con Hunt, ejecutaremos Sniffit en el ordenador con interfaz de red 172.16.17.1 y realizaremos una conexión a un servidor de correo POP desde otro ordenador.

En este ataque, estaremos interesados en capturar todas las sesiones POP que se produzcan en nuestro segmento de red excepto aquellas que impliquen al host 172.16.17.2. Para ello usaremos el siguiente archivo de configuración:

```
select both port 110
deselect both host 172.16.17.2
```

Con estos parámetros en el archivo de configuración, ejecutamos Sniffit y obtenemos en el archivo de salida lo siguiente:

172.16.17.5 [1050] <-- 194.179.41.3 [110]
+OK Gordano Messaging Suite POP3 server ready <18334432861773@iespana.es>[0xD][0xA]

172.16.17.5 [1050] --> 194.179.41.3 [110]
USER acme[0xD][0xA]

172.16.17.5 [1050] <-- 194.179.41.3 [110]
+OK acme is welcome.[0xD][0xA]

172.16.17.5 [1050] --> 194.179.41.3 [110]
PASS coyote[0xD][0xA]

Como se puede observar, el archivo de salida tiene un formato más fácil de entender que el que ofrecía Hunt. Se detallan los mensajes enviados en cada uno de los sentidos, y puede fácilmente observarse que ha recogido el nombre de usuario (acme) y la contraseña (coyote) usadas en la conexión.

Módulos externos

Por último, debe mencionarse que Sniffit permite la ejecución de módulos externos. Estos módulos pueden

programarse para realizar tareas sobre los datos capturados, como por ejemplo, darles un formato determinado, buscar cadenas concretas (e.g. contraseñas) o cualquier otra que sea necesaria. Esto proporciona a Sniffit una gran versatilidad a la hora de obtener los resultados deseados.

5.1. Concepto de scanners

Algunos autores de libros y artículos sobre seguridad definen los scanners como herramientas que analizan los sistemas en busca de sus debilidades. Desde este punto de vista podemos distinguir dos tipos de scanners: los scanners de sistema y los scanners de red. Los primeros analizarán el sistema en busca de permisos de acceso erróneos, contraseñas débiles, etc; en definitiva, puntos débiles del sistema en cuanto a su configuración individual. Los scanners de red, analizan sistemas (locales o remotos) buscando caracterizar la red, mediante la identificación de su topología u organización, los equipos activos o alcanzables desde internet, el sistema operativo que se ejecuta en ellos, los servicios ofrecidos, etc. Una implementación incorrecta u obsoleta de un servicio tras un puerto TCP abierto puede constituir un posible agujero en la seguridad del sistema, y por tanto, debemos prestarles mucha atención.

A pesar de que en muchos textos se identifican los scanners de red con los scanners de puertos, nosotros aquí vamos a distinguirlos. Nos referiremos a los scanners de red cuando hablemos de las herramientas que tratan de identificar qué máquinas están activas, los puertos TCP abiertos en una máquina determinada o las máquinas con un determinado puerto TCP abierto.

5.2. Métodos de escaneo

De forma general, dentro de los métodos de escaneo se encuentran técnicas como:

- Escaneo de equipos basado en ICMP

- Escaneo de puertos basado en TCP

- Firewalking: Firewalking es una técnica desarrollada para analizar las respuestas de paquetes IP para determinar los filtros de paquetes y desvíos de paquetes.

- Trace Routing. Mediante las técnicas trace-routing podemos determinar el trayecto que siguen nuestros paquetes desde el origen hasta el destino. Estas técnicas pueden ayudar a los atacantes a encontrar la topología de la red objetivo.

- Identificación de sistema operativo. Mediante la explotación y análisis de las características típicas de cada implementación particular de las pilas TCP/IP (lo que llamaremos fingerprinting), es posible determinar con alta probabilidad de acierto el sistema operativo ejecutandose en una máquina concreta.

Escaneo de equipos basado en el protocolo ICMP

El protocolo ICMP se desarrolló inicialmente para la notificación de errores y condiciones inusuales en el protocolo IP. Sin embargo, es posible hacer un uso indebido de este protocolo, con el objetivo de escanear sistemas remotos.

El escaneo de equipos activos en una subred es habitualmente el primer paso a dar por un atacante, para conocer hacia dónde debe encaminar sus esfuerzos.

Mensaje ICMP Echo

Usando los mensajes del protocolo ICMP, ECHO y ECHO REPLY, de solicitud de eco y respuesta respectivamente, es

posible obtener la lista de dispositivos IP activos en una subred determinada. Si se envía un paquete ECHO a una dirección, y se recibe de ella un paquete ECHO REPLY, es que dicha dirección está siendo utilizada. Existen herramientas que ayudan a realizar esta tarea, enviando múltiples paquetes ECHO. Entre ellas están ping, fping, gping y Pinger.

Además de los dispositivos IP activos, mediante el protocolo ICMP podemos extraer información de otro tipo, como por ejemplo, la franja horaria del sistema destino, la máscara de subred de los diferentes interfaces, etc.

Para detectar este tipo de ataques, se puede analizar el archivo de registro del servidor DNS, ya que aparecerán muchas peticiones de resolución de direcciones consecutivas. Asimismo, podremos obtener la dirección IP del atacante.

Mensaje ICMP Echo a dirección Broadcast

Cuando se envía un paquete ICMP Echo a la dirección de broadcast o dirección de una red, basta con un solo paquete de este tipo para que respondan todos los hosts activos de esa red. Es lógico pensar que los sistemas se comportarán de esta manera. Sin embargo, en este caso, el comportamiento dependerá de la implementación particular del protocolo ICMP. De esta forma, esta técnica se empleará con éxito en sistemas Unix, pero no servirá en los sistemas operativos de Microsoft Windows, ya que no responden a este tipo de paquetes. Según el RFC1122, este último comportamiento es el deseado.

Mensaje ICMP Timestamp

Mediante el envío de un paquete ICMP del tipo Timestamp, si el sistema destino del paquete está activo, enviará de vuelta un paquete Timestamp Reply, para indicar que implementa la transferencia de referencia de tiempo. Según el RFC1122, responder a este tipo de paquetes es decisión de cada implementación. La mayoría de los sistemas Unix implementan la respuesta, mientras que hay sistemas Windows que la implementan y otros que no.

Mensaje ICMP Information

Los paquetes ICMP de este tipo tenían como propósito inicial el de permitir que ciertos equipos carentes de disco del que extraer su propia configuración, pudieran configurarse en el momento de arrancar, y entre otras cosas obtener su dirección IP. Nótese que en el paquete, tanto la dirección origen como la destino tienen ambas valor cero.

Según los RFC 1122 y 1812, los sistemas no deberían ni generar ni responder a este tipo de mensajes, aunque en la realidad, algunos sistemas operativos responden a paquetes de este tipo cuando la dirección IP destino del paquete tiene un valor concreto. Algunos sistemas Unix comerciales y equipos Cisco implementan la respuesta a este tipo de paquetes.

Mensaje ICMP Address Mask

El proposito de este tipo de mensajes, de forma parecida a los mensajes ICMP Information, era en un principio el que los equipos sin disco pudiesen obtener la máscara de red asociada a la subred en la que estaban en el momento de arrancar. En principio los sistemas no deberían responder ante este tipo de mensajes, salvo que fuese un agente autorizado para notificar la máscara (por ejemplo, el router de la subred).

Mediante el uso de esta información, el atacante puede conocer la estructura interna de la subred, y el esquema de rutado empleado. Asimismo, es posible mediante este tipo de mensajes, identificar los routers que existen entre el atacante y la red objetivo.

Hemos visto hasta ahora técnicas de detección de equipos activos basadas en el envío de mensajes ICMP de distintos tipos. Veremos ahora algunas técnicas que estudiarán el comportamiento de la implementación particular del protocolo ICMP, mediante el análisis de los mensajes de error ICMP. Mediante este tipo de estudio, se pueden conocer datos sobre la existencia o no de dispositivos de filtrado, o la configuración de las listas de acceso empleadas. Estas técnicas incluyen, en general:

- Modificación malintencionada de la cabecera IP de un

paquete.
- Uso de valores indebidos en los campos de la cabecera IP.
- Abuso de la fragmentación.
- Escaneo basado en el protocolo UDP, ya que es el protocolo ICMP el que se encarga de la notificación de las anomalías de aquel.

Campos de cabecera IP no válidos

Al fijar un valor incorrecto de los campos de la cabecera IP se pretende obtener de la máquina objetivo un mensaje ICMP de error: ICMP Parameter Problem. Este mensaje se genera cuando al pasar un paquete por un sistema y éste procesar aquel, se encuentra un problema con la cabecera IP que no está contemplado en ningún otro mensaje de error ICMP. Este mensaje se envía solo si el paquete es descartado debido al error.

Este error debería ser generado por todos los routers. Sin embargo, no todos comprueban la cabecera IP de la misma forma. Por ello, es posible identificar el fabricante del router en función del comportamiento del mismo. Los sistemas implementan generalmente la verificación de la versión, y si no es la 4, descartan el paquete. Del mismo modo, comprueban el valor del checksum para evitar errores derivados del transporte por la red.

Aprovechando esta funcionalidad, un atacante puede escanear un rango de direcciones IP completo asociado a una subred.

Si se pretende conocer la configuración de las listas de control de accesos (ACLs) existente, se debe escanear todo el rango de direcciones IP con todos los protocolos y puertos posibles, para obtener datos detallados sobre los servicios y la topología de la red así como del tráfico que pasa y el que no.

Las redes se pueden proteger frente a este tipo de ataques si los firewalls se encargan de verificar este tipo de errores y no permiten trafico de este tipo. Del mismo modo, si el dispositivo de filtrado no implementa esta característica, es posible filtrar los mensajes ICMP Parameter Problem en su camino de vuelta.

Campos de cabecera IP no validos

Es posible modificar un paquete IP para que contenga valores no válidos en alguno de sus campos. En el caso de que un equipo reciba un paquete de este tipo, generará un mensaje de error ICMP Destination Unreachable.

Un ejemplo de este tipo de modificación es fijar un valor no válido para el campo de protocolo. Cuando el sistema objetivo reciba este mensaje, generará un error ICMP. Si no se recibe esta respuesta, podemos pensar que existe un dispositivo de filtrado, excepto si el sistema objetivo es alguna versión Unix como AIX o HP-UX.

Por tanto, podríamos pensar en realizar un escaneo probando con todos los valores de protocolo posibles, en total 256. Si aparentemente se detectan muchos protocolos abiertos (ausencia de mensajes de error ICMP), podremos pensar que existe un dispositivo de filtrado. Si el dispositivo filtra los mensajes ICMP Port Unreachable, entonces los 256 valores para el protocolo parecerán existir y estar disponibles.

La protección frente a ataques de este tipo es posible, bien configurando un firewall para bloquear la salida de mensajes ICMP Protocol Unreachable, o bien configurando el firewall para bloquear los protocolos que no están permitidos.

Fragmentación IP

Cuando un sistema recibe datagramas IP fragmentados, y algún fragmento de un datagrama concreto no se recibe tras un tiempo estipulado, el sistema descartará ese paquete. Al descartarse el paquete, se envía al remitente del datagrama fragmentado un mensaje ICMP Fragment Reassembly Time Exceeded.

Si aprovechamos esta situación, podemos mandar datagramas IP fragmentados al rango de direcciones IP de la subred a atacar, omitiendo algún fragmento. En el caso de que se reciba el mensaje ICMP de tiempo excedido al reconstruir los fragmentos, el puerto en cuestión está abierto y no filtrado. En caso contrario, o existe un dispositivo de filtrado, o está cerrado.

La forma de protegerse contra este ataque es, una vez

más, no permitir la salida de este tipo de mensajes ICMP hacia el exterior.

5.3. Protocolo TCP

Para comprender en qué están basados los distintos tipos escaneo de puertos, vamos a presentar de forma sucinta las bases del protocolo de transporte de internet, TCP, y más concretamente la forma en la que se establecen y liberan las conexiones. Para una descripción en más detalle veáse el documento RFC-793 del pliego de condiciones, sobre el protocolo TCP.

El protocolo de transporte TCP ofrece un servicio orientado a conexión de transmisión de datos fiable a las aplicaciones de nivel superior mediante el asentimiento numerado y la repetición de paquetes cuando sea necesario.

La cabecera de un segmento TCP (en adelante lo llamaremos así) tiene la estructura mostrada en la figura 5-1.

El campo 'source port' contiene el número de puerto que identifica a la aplicación de la que parte la conexión y el 'destination port' es el número de puerto que identifica a la aplicación destino de la conexión. En las comunicaciones entre aplicaciones distintas en una misma interfaz IP debemos ser capaces de identificar hacia qué aplicación va cada segmento TCP. Para ello se crea una nueva abstracción: el puerto TCP. El puerto TCP no es más que un número entero mediante el cual haremos referencia a distintas conexiones TCP. La asignación de número de puerto se realiza teniendo en cuenta que existen ciertos números reservados a ciertos servicios (números de puerto por debajo de 1024). El sistema operativo, de forma dinámica, asigna a cada aplicación cliente un número de puerto que esté libre (que estará por encima de 1024). Para las aplicaciones servidoras, los números de puerto se asignan de manera estática y son conocidos por las aplicaciones cliente. Para una descripción completa de los puertos asociados a aplicaciones conocidas véase el archivo "nmap-services" que viene con nmap.

El campo 'sequence number' indica el número de secuencia del primer octeto de datos, siempre que el bit SYN no esté activo. Cuando el bit SYN está activo, este campo contendrá el número de secuencia inicial (ISN), y el primer octeto de datos

será ISN+1. El número de secuencia inicial es un número aleatorio entre 0 y 4.294.967.295 (2^{32}).

El campo 'acknowledge number', en las tramas con el bit de control ACK activo, contendrá el número del siguiente número de secuencia que el emisor de este segmento espera recibir. Una vez que la conexión está establecida, este campo siempre se envía.

El campo 'data offset' indica el número de palabras de 32 bits que ocupa la cabecera TCP, lo que indica dónde comienzan los datos.

Después del campo 'reserved', de 6 bits reservados para uso futuro, nos encontramos con el campo de bits de control. Son seis bits y pueden ser:

U (URG): Datos urgentes.

A (ACK): Asentimiento.

P (PSH): Función Push.

R (RST): Reseteo de la conexión.

S (SYN): Sincronización de números de secuencia.

F (FIN): No hay más datos del emisor.

El campo 'window' indica el número de octetos en la ventana del emisor de este segmento, comenzando por el octeto indicado por el bit ACK.

El campo 'checksum' contiene el valor de 16 bits en complemento a uno de la suma en complemento a uno de todas las palabras de 16 bits del segmento TCP. Si el número de octetos del segmento es impar, se rellena un octeto con ceros por la derecha. Este relleno no se transmitirá como parte del segmento, sino que en el destino se calculará el 'checksum' de la misma manera.

El campo 'urgent pointer' es el puntero a datos urgentes. Consiste en un offset positivo, contado a partir del número de secuencia del segmento en el que aparece y solo puede ser interpretado en los segmentos con el bit URG activo.

En el campo 'options' se pueden transmitir las opciones. Este campo debe tener una longitud múltiplo de 8 bits. Cada opción debe comenzar en los límites de un octeto. Hay dos formatos posibles para especificar las opciones:

- Usar solo un objeto para especificar el tipo de opción.

- Usar un octeto para especificar el tipo de opción, otro para especificar la longitud de la opción y otro para la opción propiamente dicha. El octeto de longitud de opción indicará la longitud total de la opción, es decir, incluirá los octetos de tipo, longitud y datos de opción.

La lista de opciones debe ser más corta que lo designado en el campo 'data offset', ya que el contenido de la cabecera más allá del final de las opciones debe ser relleno (esto es, ceros).

El campo 'data' contendrá los datos de protocolo de una capa más alta.

Para los temas que aquí se tratarán, será de especial interés conocer la forma en que se establecen y liberan las conexiones TCP. Por ello presentamos en las siguientes sección cómo se realizan estas tareas.

Establecimiento de Conexiones TCP

El establecimiento de conexiones TCP se lleva a cabo mediante el llamado Three-way Handshake Protocol o Protocolo a Tres Bandas. El intercambio de segmentos que tiene lugar en un establecimiento de conexión TCP entre dos extremos A y B está representado gráficamente en la figura 5-2.

El extremo que desea comenzar una conexión envía un segmento con el bit de control SYN activo y número de secuencia X (que coincidirá con el número de secuencia inicial (ISN) del extremo A, comentado más arriba). El extremo B, al recibir dicho segmento, enviará otro al extremo B con SYN activo, número de secuencia Y (que coincidirá con el número de secuencia inicial (ISN) del extremo B) y el bit ACK activo, conteniendo X+1 el campo 'acknowledgement number' para indicar que se ha asentido el segmento con número de secuencia X y se espera el de número de secuencia X+1. Una vez recibe este segmento el extremo A, envía un segmento de asentimiento al segmento con

número de secuencia Y+1 y la conexión queda establecida.

Liberación de la conexión

En condiciones normales una conexión TCP establecida entre dos extremos A y B se libera en tres fases, de manera consensuada entre ambos extremos, como se indica en la figura 5-3.

En dicha figura se ha supuesto que el extremo que desea finalizar la conexión es el B. Para ello envía un segmento con el bit FIN activo, número de secuencia X y, en su caso, el bit ACK activo conteniendo Y en el campo 'acknowledgement number'. El extremo A, al recibir este segmento, responde activando el bit FIN en el segmento de número de secuencia Y, activando el bit ACK y poniendo X+1 en el campo 'acknowledgement number'. Por último, el extremo B responde en el segmento X+1 con el bit ACK activo y el campo 'acknowledgement number' conteniendo Y+1. La conexión se da entonces por finalizada.

Sin embargo, el cierre de una conexión no siempre se realiza de forma consensuada. Es posible que se produzca un cierre abrupto, algo que no debe ocurrir bajo condiciones normales. Para abortar una conexión de forma abrupta, cualquiera de los extremos envía una segmento con el bit RST activo, y el número de secuencia que corresponda. Esto hace que se aborte inmediatamente la conexión en ambos extremos y se liberen los recursos asociados.

5.4. Escaneo de puertos

El escaneo de puertos de los host de nuestra red proporciona al atacante información sobre los servicios de red disponibles. El atacante puede entonces buscar agujeros de seguridad en dichos servicios, bien por una incorrecta configuración de los mismos, bien porque no tengamos actualizado el servicio y por tanto los posibles fallos de implementación no estén corregidos. El escaneo de puertos puede proporcionar adicionalmente detalles sobre el sistema operativo instalado en cada host, además de algunos detalles sobre la estructura de nuestra red.

Realizar un escaneo de puertos es, en principio, una tarea muy sencilla. De hecho, para comprobar si un hay un puerto abierto en una máquina, podemos intentar realizar conexión a ella con `telnet` a dicho puerto. Por ejemplo, si queremos saber si una máquina concreta ofrece servicio SMTP, podríamos intentar un telnet al puerto 25. En caso de que nos deje conectar, el puerto esta abierto y el servicio está disponible. Será cuestión entonces de buscar agujeros en el servicio que nos permitan llevar a cabo un ataque contra el mismo.

Incluso aunque es posible realizar un rudimentario escaneo de puertos usando `telnet`, los atacantes no van a usar este servicio para buscar puertos abiertos con relativa seriedad. Para realizar esta tarea existen herramientas que automatizan este proceso y que permiten multitud de tipos de escaneo.

5.5. Tipos de escaneo

Podemos clasificar los distintos tipos de escaneos de puertos atendiendo a características como su ámbito, el protocolo objetivo, el servicio al que va dirigido o la técnica usada para llevarlo a cabo.

Atendiendo al ámbito en el que se realiza el escaneo, podemos tener dos tipos:

- **Escaneo horizontal**: se produce cuando buscamos un servicio concreto abierto en varias máquinas. Podemos pensar en un atacante que posea un exploit o conozca un agujero de seguridad para un servicio concreto. Para aprovechar esta situación, buscará máquinas en las que ese servicio esté disponible a lo largo de uno o varios segmentos de red. Cuando encuentre una máquina ejecutando ese servicio, intentará conseguir acceso privilegiado en ella.

- **Escaneo vertical**: se produce normalmente cuando el foco de interés se centra en obtener acceso privilegiado en una máquina concreta y buscamos los servicios disponibles en ella, usándolos después para obtener acceso privilegiado.

Si atendemos a la técnica usada para realizar el escaneo a puertos TCP, nos encontramos con los siguientes:

- **Escaneo Open**: se basa en el establecimiento de una conexión completa mediante el protocolo de establecimiento de conexiones three-way handshake. Por ello son muy sencillos de detectar y detener, ya que un en los archivos de registro de la actividad TCP, aparecerá un conjunto más o menos grande de intentos de conexión, y en los casos en los que tenga éxito, será para liberarla justo después. Este tipo de escaneo nos recuerda a lo que comentábamos anteriormente sobre intentos de conexión con `telnet`. Para llevar a cabo un escaneo Open, el programa hace una llamada a `connect()`. Con ello, el atacante conoce el estado del servicio al que intenta acceder de una forma rápida, fácil y fiable, sin necesidad de ningún privilegio sobre la máquina objetivo, ya que la llamada al sistema `connect()` es accesible por cualquier usuario en la mayoría de máquinas UNIX.

- **Escaneo Half-Open**: en esta técnica el atacante finaliza la conexión antes de que se complete el protocolo de acuerdo de tres vías. Esto dificulta la detección del ataque para algunos programas de detección muy simples. Dentro de esta técnica se encuentra el SYN Scanning, en el que cuando el atacante recibe de la máquina escaneada los bits SYN + ACK, envía el bit RST, en vez del ACK que correspondería al proceso normal del protocolo Three-Way Handshake. Sin embargo, este tipo de escaneo se puede detectar y bloquear fácilmente en cualquier firewall.

- **Stealth Scanning**: Se conoce con este nombre a la familia de técnicas de escaneo que presentan cualquiera de las siguientes características:

    - Eluden firewalls o listas de control de accesos.

    - No son detectados por sistemas de detección de intrusos.

    - Simulan tráfico real o normal, para no levantar sospechas ante cualquier analizador de protocolos.

Técnicas Stealth Scanning

Una técnica dentro de las stealth scanning es la conocida como SYN+ACK, en la que se envía al destino una trama con SYN+ACK, en lugar de SYN. Si el puerto está abierto, la trama se ignora. Si está cerrado, se sabe que no se ha recibido una trama SYN previamente, por lo que se genera un error y se envía una trama RST para finalizar la conexión. Actualmente esta técnica no se usa mucho, ya que los motivos posibles por los que una máquina no responda a una trama SYN+ACK son muchos, entre ellos, las listas de control de acceso en los routers, firewalls, etc.

Otro tipo de escaneo que se engloba dentro de las técnicas Stealth Scanning es el FIN Scanning. En él se envía una trama FIN al objetivo; de esta forma, si el puerto está abierto, no responderá, y si está cerrado, responderá con RST. Como en el caso anterior, este tipo de escaneo puede generar muchos falsos positivos, por lo que actualmente no se usa mucho.

El escaneo XMAS (escaneo en árbol de navidad) consiste en enviar una trama al objetivo con todos los bits TCP puestos a uno (ACK, URG, PST, RST, SYN y FIN). Si el puerto está abierto, la trama se eliminará, ya que viola el protocolo three-way handshake. De nuevo, esta técnica genera muchos falsos positivos por las mismas razones que en los casos anteriores. Además, solo es aplicable contra máquinas Unix, debido a que está basado en el código de red de BSD.

Otra técnica, opuesta al escaneo XMAS, es la llamada NULL o NULL flags scan, en la que se envía al objetivo una trama con todos los bits TCP reseteados. El efecto que provoca es el mismo: no se devuelve nada al origen si el puerto está abierto, y se envía RST al origen si está cerrado. Los problemas de esta técnica, como se puede intuir, son los mismos que en las anteriores.

Existe una técnica conocida como escaneo ACK, usada comúnmente para determinar las reglas de filtrado de un firewall. Es especialmente útil a la hora de determinar si un firewall tiene en cuenta el estado o es un filtro estático que sólo bloquea los paquetes que tengan SYN activado.

Escaneos UDP

Con este nombre se conoce a una técnica de escaneo que difiere radicalmente de los vistos hasta ahora en el uso del protocolo UDP en lugar de TCP. Al enviar un datagrama UDP con cero bytes de datos a un puerto abierto, este no ofrece respuesta alguna; sin embargo, si está cerrado, el sistema responde con un error ICMP: ICMP_PORT_UNREACHABLE. Por un lado, estos ataques son fácilmente detectables y bloqueables, y por otro, al ser UDP un protocolo no orientado a conexión, la pérdida de datagramas puede llevar a un número de falsos positivos demasiado elevado.

5.6. Herramientas de escaneo

Las herramientas de escaneo disponibles en internet proliferan cada vez más. Muchas de ellas son gratuitas y con el código fuente disponible. Esto puede verse como un arma de doble filo, ya que puede ser usado para detectar servicios abiertos y vulnerabilidades con el objetivo de mejorar la seguridad de la red, o por el contrario, para encontrar esas vulnerabilidades y explotarlas de manera malintencionada.

Veremos a continuación las posibilidades que ofrece una de las herramientas de escaneado más populares, que además es de distribución gratuita. Muchas de las distribuciones de Linux actuales, entre ellas Mandrake Linux, incluyen esta herramienta.

Nmap 3.0

Nmap (apócope para Network Mapper) es una herramienta de exploración de redes y de auditoría de seguridad. Permite escanear grandes redes en poco tiempo, al tiempo que es capaz de trabajar contra equipos individuales. Nmap envía paquetes IP de diferentes formas para determinar los equipos activos en una red, los servicios que ofrecen, los sistemas operativos (y sus versiones) que están ejecutando, el tipo de filtros de paquetes o firewalls que están usando, y muchas otras características.

Nmap soporta múltiples técnicas para trazar la estructura de redes que contienen filtros IP, cortafuegos, routers y otro tipo de obstáculos. Para ello implementa mecanismos de escaneado

TCP y UDP, barridos de ping, detección de sistema operativo, etc.

Una vez ejecutado, Nmap devolverá una lista de puertos en la máquina objetivo. Cada uno de esos puertos llevará una información asociada en la salida proporcionada por Nmap, como son: el nombre del servicio tras el puerto (para los "puertos bien conocidos"), número de puerto, estado y protocolo. El estado de un puerto, en la salida de Nmap, podrá ser open, filtered o unfiltered. Si un puerto se encuentra en estado open, la máquina objetivo aceptará conexiones a ese puerto. El estado filtered indica que existe un dispositivo de filtrado protegiendo la máquina objetivo y que impide determinar si el puerto está abierto o no. Un puerto que aparece en estado unfiltered está, según lo que nos indica Nmap, cerrado, y además no existe ningún dispositivo de filtrado que lo proteja. Dependiendo de las opciones que usemos al ejecutar Nmap, podremos obtener además características del host remoto como, por ejemplo, sistema operativo, secuenciabilidad TCP, nombres de usuario de los propietarios de los programas en los puertos remotos, etc. A continuación examinaremos todas estas características de Nmap.

Existe una versión de Nmap para Windows 2000. La funcionalidad es la misma que en la versión Linux, con la diferencia de que la versión Windows está provista de una interfaz gráfica, que facilita la ejecución a usuarios poco familiarizados con la linea de comandos. Sin embargo, esta interfaz gráfica se encarga simplemente de arrancar Nmap con los argumentos adecuados. Por lo tanto, nosotros aquí hablaremos de Nmap en linea de comandos, ya que esta versión existe para Linux y Windows, y solo para este último la versión oficial con interfaz gráfica. No obstante, existen varios front-end no oficiales para la versión Linux.

El formato de la línea de comando para la ejecución de Nmap es el siguiente:

```
nmap [tipo_de_escaneo] [opciones] <máquina/red_1
... máquina/red N>
```

Tipos de escaneo

Los tipos de escaneo que permite Nmap son los siguientes:

-sT: Con esta opción, realizaremos el escaneo que hemos denominado Open o TCP connect, por lo que se realizarán conexiones completas, según el protocolo de establecimiento de conexiones TCP a tres bandas.

-sS: Mediante esta opción, activamos el modo de escaneo SYN, también conocido como Half Open, ya que la conexión TCP solicitada es cerrada de forma abrupta antes de que se termine de establecer.

-sF: Escaneo Stealth FIN.

-sX: Escaneo XMAS Tree.

-sN: Escaneo NULL.

-sP: Ping scanning. Nos hemos referido a él en el apartado de escaneos basados en el protocolo ICMP, cuando hablábamos del escaneo mediante mensajes ICMP Echo. Como muchos sistemas disponen de dispositivos de filtrado que bloquean la salida de los mensajes de respuesta ICMP, Nmap usa una técnica adicional, en paralelo con la tradicional ICMP Echo request, mediante la que se envía un paquete TCP con el bit ACK activo al puerto 80 (http), con lo que si recibimos respuesta con RST activo, el puerto está abierto.

-sU: Escaneo UDP.

-sO: Realiza escaneo IP, es decir, envía los valores posibles en el campo de protocolo del datagrama IP, para comprobar si es soportado en la máquina destino. Si recibimos un mensaje ICMP Protocol Unreachable, entonces el protocolo no está en uso.

-sI: Idlescan. Esta es una técnica avanzada de escaneo en la que se realiza además una falsificación de la dirección IP origen del escaneo. Este método se usa normalmente para determinar las relaciones de confianza entre máquinas. El listado de salida muestra los puertos abiertos con respecto a la máquina cuya dirección se ha usado para enviar los paquetes. Obviamente, para que este método tenga éxito

es de vital importancia la recogida previa de información relativa a la víctima, para confeccionar una lista de posibles máquinas en las que confía.

-sA: Escaneo ACK. Éste es un método avanzado que normalmente se usa para determinar si un firewall es una máquina de estados o si es simplemente un filtro de paquetes. Se envía un paquete ACK al objetivo, y si este devuelve un RST, el puerto no está filtrado.

-sW: Escaneo Window. Éste, al igual que el anterior es un método de escaneado avanzado para determinar reglas en los firewalls, con la particularidad de que puede determinar bajo ciertos casos los puertos que están abiertos, debido a un fallo en la notificación del tamaño de la ventana en algunas implementaciones de la pila TCP/IP.

-sR: Escaneo RPC. Es una combinación de las técnicas de escaneo de Nmap, que además busca, detrás de los puertos encontrados abiertos, puertos RPC. Para ello envía comandos NULL con el programa SunRPC a los puertos en estado Open.

-sL: Genera una lista, sin escanear, de direcciones IP y sus correspondientes nombres. Se resolverá mediante DNS a menos que se indique lo contrario con la opción -n.

-b: Implementa el FTP Bounce attack. Este ataque permitía conectar a un servidor FTP y enviar un archivo a cualquier dirección de internet.

Opciones

Las opciones no son requeridas para ejecutar nmap, si bien pueden ser muy útiles.

-P0: Con esta opción, Nmap no verifica mediante ping el estado de los objetivos antes de atacarlos. Esto permite ataques a objetivos cuyos firewalls filtran los mensajes ICMP Echo Request.

-PT: Según las páginas del manual de Nmap, esta opción provoca que se realice un "Ping TCP". Consiste en enviar paquetes TCP con ACK activo en lugar de mandar mensajes ICMP de petición de respuesta. Si se recibe RST

del destino, el equipo está abierto. Se usa esta opción para evitar los bloqueos de firewalls a mensajes ICMP de echo. Por defecto se hace "ping" al puerto 80, ya que este puerto está normalmente sin filtrar, pero podemos especificar el puerto al que hacer "ping" con -PT<puerto>.

-PS: Usa paquetes con SYN activo en lugar de paquetes con ACK activo; éstos los usan los usuarios no privilegiados y aquellos solo pueden ser construidos por los usuarios privilegiados. Los equipos que estén activos responderán con RST (o con SYN|ACK, poco comúnmente). Es posible establecer el puerto destino de la forma expresada en la opción anterior.

-PI: Realiza un "ping" tradicional y verdadero, es decir, mediante un mensaje ICMP Echo Request.

-PB: Esta opción es el tipo de escaneado de equipos por defecto. Usa las técnicas ACK e ICMP (-PT y -PI) en paralelo. De esta forma se pueden escanear redes cuyos firewalls las protegen de una de las dos técnicas.

-O: Activa la identificación del sistema operativo remoto mediante la técnica de fingerprinting. Además, esta opción habilita la recopilación de otros datos, entre los que están el "Uptime" (tiempo desde la última vez que se arrancó el sistema). Otro de los datos que se pueden obtener mediante la activación de esta opción es una medida de lo predecibles que son los números de secuencia TCP. Esto es de interés para una posible falsificación de conexiones que se basan en las relaciones de confianza entre equipos, o para esconder la fuente de un ataque.

-I: Activa el escaneo "TCP reverse ident". El protocolo ident permite descubrir el nombre de usuario del dueño del proceso que está asociado a un puerto, incluso si ese proceso no inició la conexión. Esto solo se puede hacer con una conexión completa al puerto TCP objetivo. Este tipo de escaneo solo funcionará en caso de que el objetivo tenga en ejecución el servicio identd.

-f: Ejecuta el escaneo seleccionado usando fragmentación en los paquetes IP. La idea consiste en dividir la cabecera de los paquetes TCP de forma que sea más complicada la detección por parte de los filtros de paquetes y sistemas de detección de intrusos.

-v: Modo exhaustivo en la salida. Da mucha más información sobre lo que ocurre durante el escaneado. La opción -vv hace que la salida sea aún más exhaustiva.

-h: Muestra una referencia rápida sobre el uso de Nmap.

-oN <logfile>:
Registra la salida del escaneado de una forma "legible para las personas" en el archivo indicado por "logfile".

-oX <logfile>:
Registra la salida en formato XML en el archivo que se indique como argumento. Esto facilita que otros programas interpreten la salida de Nmap.

-oG <logfile>:
Registra la salida del escaneado en un formato "grepable". Este formato incluye toda la información en una sola linea. Al igual que antes, la salida se almacena en el archivo indicado por "logfile".

-oA <baselog>:
Indica a Nmap que se registre la salida en todos los formatos. Se da como argumento el nombre base de archivo "baselog" y la salida se almacenará en los ficheros "baselog.nmap", "baselog.gnmap" y "baselog.xml".

-oS <logfile>:
Registra la salida en un formato sin interés. Se trata de un formato cuya tipografía mezcla mayúsculas y minúsculas arbitrariamente, cambiando algunas letras por números, de forma que escribiría "filtered" como "fIlt3R3d".

–resume <filename>:
Para continuar con un escaneo que fue anteriormente cancelado mediante Ctrl+C o por errores en la red.

–append_output:
Indica que no se debe sustituir un fichero de registro en el caso que ya exista, sino que se deben añadir los datos nuevos al final del existente.

-iL <inputfile>:
Toma los datos de entrada del archivo "inputfile" en lugar de tomarlos de la línea de comandos. El archivo debe contener una lista de equipos o redes, separados por

espacios, tabuladores o lineas nuevas.

-iR: Esta opción hace que Nmap genere una lista aleatoria de equipos para escanear.

-p <rango>:
Sirve para especificar el rango de puertos que queremos escanear. El formato de un intervalo de puertos se expresará separando los puertos origen y destino con un guión, y con una coma separaremos los distintos intervalos que queramos especificar. Por defecto el intervalo es desde 1 hasta 1024, y todos los puertos superiores cuyo servicio esté listado en los archivos de servicios que vienen con Nmap. Si queremos especificar el protocolo al que asociamos los puertos en los que estamos interesados, bien UDP o TCP, introducimos al principio de la especificación del rango "T:" si nos referimos a TCP y "U:" si nos referimos a UDP. En el caso de que se quiera realizar un escaneado IP (-s0), con esta opción especificamos el rango de protocolos en el que estamos interesados.

-F: Escaneado rápido. Indica que sólo queremos escanear los puertos asociados a los servicios listados en los archivos de servicios que vienen con Nmap. Igualmente, en el caso de un escaneado IP (-s0), solo nos referiremos a los protocolos listados en los archivos de protocolo.

-D <cebo1 [,cebo2][,ME]...>
Provoca que se realice un escaneo con cebos, de forma que al escanear al objetivo, a éste le parezca que los equipos que se especifican como cebos están también realizando un escaneo al objetivo. Usamos "ME" detrás de la dirección que queremos que represente nuestra IP; si no lo usamos, Nmap decidirá esta cuestión aleatoriamente. Los cebos son usados en todos los escaneados (tanto de redes y equipos, como de puertos y protocolos).

-S <direcciónIP>
En algunos casos Nmap no podrá determinar la dirección IP de la máquina desde la que se ejecuta. En esta situación, podremos indicar a Nmap cual es dicha dirección IP mediante esta opción. Podría pensarse que esta opción

se puede usar para hacer pensar a los objetivos que es otra dirección IP la que les está escaneando. Este último no era el objetivo inicial de esta opción.

-e <interface>
En los casos en los que no sea posible para Nmap determinar la interfaz mediante la que enviar y recibir los paquetes, se puede especificar con esta opción.

-g <número_puerto>
Selecciona el puerto fuente usado para los escaneos.

-n Hace que Nmap nunca realice la resolución DNS inversa. Sirve normalmente para agilizar el proceso de escaneado.

-R Hace que Nmap realice siempre la resolución DNS inversa.

-r Hace que el orden en el que los puertos son escaneados no sea aleatorizado.

--randomize_hosts
Con esta opción hacemos que Nmap escoja aleatoriamente el orden de los equipos para ser escaneados, tomandolos en grupos de 2048. Esto se usa para hacer el escaneado menos obvio a los ojos de los sistemas de monitorización de redes.

-M <max sockets>
Especifica el número máximo de sockets que serán usados en paralelo para un escaneo Open.

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>
Selecciona entre una de estas seis políticas de temporización del escaneado. También se pueden seleccionar especificando su número asociado (de 0 a 5) en vez del nombre. Mientras más bajo sea el número asociado, más tiempo se esperará entre cada prueba, y por tanto, menos evidente será el escaneo.

--host_timeout <milisegundos>
Especifica el tiempo que Nmap debe pasar escaneando un equipo antes de rendirse con esa dirección IP. Por defecto, este tiempo es infinito.

--max_rtt_timeout <milisegundos>
Especifica el tiempo máximo que Nmap debe esperar una

respuesta a una prueba particular antes de retransmitir o desecharla por tiempo excedido. Por defecto, esta variable se sitúa en 9000.

--min_rtt_timeout <milisegundos>
Si los equipos objetivos comienzan a responder con rapidez, Nmap reducirá la cantidad de tiempo asignada a cada prueba. Esto agiliza el escaneado pero puede llevar a pérdidas de paquetes. Con esta opción se puede garantizar que Nmap esperará al menos el tiempo especificado antes de rendirse en una determinada prueba.

--initial_rtt_timeout <milisegundos>
Especifica la cantidad de tiempo inicial por prueba. Normalmente Nmap puede obtener una buena estimación de este tiempo mediante el "ping" y las primeras pruebas, con lo que esta opción suele tener sentido solamente en el caso de los equipos protegidos por dispositivos de filtrado.

--max_parallelism <número>
Fija el número máximo de escaneos que Nmap puede realizar en paralelo.

-scan_delay <milisegundos>
Especifica la mínima cantidad de tiempo que Nmap debe esperar entre pruebas.

Especificación de objetivos

Todo aquello que no sea una opción o argumento correspondiente a una opción, será tratado por Nmap como un equipo objetivo. El caso más simple es el de especificar una sola máquina en la linea de comandos mediante su nombre o dirección IP. Si queremos escanear una subred, debemos añadir **/mascara**, donde "mascara" debe estar entre 0 (todo internet) y 32 (solo el equipo especificado). Usaremos pues, /24 para direcciones de clase C y /16 para direcciones de clase B.

Además, Nmap permite usar una notación más potente basada en listas o rangos para cada elemento de la dirección IP. Los rangos para cada octeto de la dirección IP se introducen de la misma forma que cuando tratamos los rangos de puertos con la opción -p. La diferencia está en que aquí podremos usar el asterisco '*' para referirnos a todos los posibles valores que pueda tomar una dirección IP.

Capítulo 6

ATAQUES DE DENEGACIÓN DEL SERVICIO

Los ataques de denegación del servicio explotan la naturaleza asimétrica de ciertos tipos de tráfico de red. Un método de ataque intenta conseguir que el objetivo tenga que utilizar más recursos procesando el tráfico de la red de los que debe usar el atacante. Otro es el de sincronizar a muchos atacantes.

Los ataques de denegación del servicio se pueden agrupar en varias categorías. Una división puede ser: ataques al ancho de banda o al rendimiento del equipo, ataques a protocolo y ataques lógicos.

Los ataques al ancho de banda se caracterizan por intentar consumir recursos como el ancho de banda de la red o tiempo de proceso del equipo. Los ataques con gran volumen de datos pueden consumir todo el ancho de banda disponible entre el proveedor de servicios de internet y el sistema víctima. Otros ataques de este tipo pueden basarse en aumentar la tasa de paquetes enviados más que el ancho de banda. De esta forma, ante una tasa muy alta de paquetes recibidos, los equipos víctima deben pasar mucho tiempo procesando las cabeceras, con lo que puede llegar a colapsar los recursos del equipamiento, tales como recursos de memoria, tiempo de CPU, interrupciones, etc.

Los ataques a protocolo se aprovechan del diseño inherente de algunos protocolos de red comunes. Estos ataques no explotan directamente las debilidades de las pilas TCP/IP, sino que aprovechan el comportamiento esperado de protocolos como

TCP, UDP e ICMP. Ejemplos de este tipo de ataques son el SYN Flooding, smurf o su variante fraggle.

Los ataques lógicos o ataques a vulnerabilidades del software, al contrario que los anteriores, que intentan consumir recursos de red o de las máquinas, tratan de explotar las vulnerabilidades del software de red, como un servidor web o la pila TCP/IP. Algunas de estas vulnerabilidades son tan simples como la caída del sistema ante un paquete mal formado. Ejemplos de este tipo de ataques son el ataque Teardrop, land, el ping de la muerte, etc.

No es posible crear una protección infalible contra ataques de denegación del servicio. Pueden tomar formas tantas formas distintas como la imaginación del atacante permita. Cualquier cosa que se traduzca en una respuesta del sistema local, cualquier cosa que reserve recursos del sistema puede ser usada en un ataque de denegación del servicio.

A continuación trataremos varios de los ataques de denegación del servicio más representativos.

6.1. TCP SYN Flooding

Se conoce con este nombre a los ataques de inundación de paquetes de sincronización. Un ataque SYN Flood consume los recursos del sistema hasta que no se pueden atender más conexiones entrantes. El ataque hace uso del protocolo básico de establecimiento de conexiones TCP a tres bandas, unido a la falsificación de la dirección IP origen.

El atacante falsifica su dirección fuente y pone en su lugar una dirección no encaminable. De esta manera inicia la negociación de una conexión a uno de los servicios de la máquina víctima. Con la apariencia de un cliente intentando abrir una conexión TCP, el atacante envía un mensaje de sincronización malintencionadamente generado. La máquina víctima responde enviando un paquete con los bits SYN y ACK activados. Sin embargo, la dirección a la que se envía este paquete no es la dirección del atacante, sino una dirección de otra máquina o incluso una dirección que no puede ser encaminada (o que simplemente no existe). Por tanto, no se recibirá la respuesta RST para liberar la conexión en proceso de negociación.

El tercer paso en el protocolo de establecimiento de conexiones TCP, en el que se envía un paquete con el bit ACK activo, no se completará. En consecuencia, se consumen recursos de la red. La conexión queda en estado semiabierto hasta que el intento de conexión caduca. El atacante inunda el puerto del sistema víctima con petición tras petición de conexión, más rápidamente que el tiempo de caducidad para liberar los recursos asociados a ellas. Si esto continúa, todos los recursos estarán en uso y no se podrán aceptar más conexiones entrantes, y esto no solo para las conexiones a ese servicio, sino para todas las conexiones entrantes. En la figura 6-1 se muestra un esquema de este ataque.

Los usuarios de Linux tienen varias formas de intentar defenderse de estos ataques. Por un lado, el filtrado de paquetes basado en la dirección origen. Mediante esta técnica se filtran todos los paquetes cuya dirección origen caiga dentro de los rangos más comúnmente usados de direcciones falsificadas, pero nada garantiza que la dirección falsificada por el atacante caiga dentro de los rangos filtrados. Se verá con más detalle el filtrado de estas direcciones en el capítulo dedicado a firewalls.

Otra opción es la de habilitar el módulo syn cookie en el kernel. Con este módulo, cuando la cola de conexión comienza a llenarse, el sistema comienza a responder a las peticiones de sincronización con cookies SYN, en lugar de con SYN-ACKs, y libera esa posición en la cola. De esta forma, la cola nunca se llena completamente. La cookie tiene un pequeño tiempo de caducidad; si el cliente no responde dentro de ese tiempo, el servidor no envía el paquete SYN-ACK. La cookie tiene un número de secuencia que está generado en base al número de secuencia original del paquete SYN, las direcciones y puertos origen y destino y un valor secreto. Si la respuesta a la cookie coincide con el resultado del algoritmo de cifrado, el servidor tiene una razonable seguridad de que el SYN inicial era válido.

Casi todas las distribuciones de Linux desde Red Hat 6.0 traen las cookies SYN compiladas en el kernel por defecto. Si el sistema no trae activada la protección de cookies SYN, deberá configurarse dicha opción y recompilar e instalar el nuevo kernel.

6.2. Ping Flooding

Cualquier mensaje que obtiene una respuesta de la máquina a la que se envía puede ser usado para degradar la calidad de la conexión de red, forzando al sistema a pasar la mayor parte del tiempo respondiendo. El mensaje ICMP echo request enviado por el comando ping suele ser el responsable de muchos ataques de denegación del servicio. Un ataque llamado smurf y sus variantes, obligan a un sistema a consumir sus recursos respondiendo a mensajes ICMP echo request. Una forma de llevarlo a cabo es falsificar la dirección origen de la victima y enviar un mensaje de petición de respuesta a toda una red mediante broadcast. Un solo mensaje falsificado puede conseguir que cientos o miles de respuestas se envíen a la victima. Otra forma de llevarlo a cabo es instalar un troyano en muchas máquinas de internet y sincronizarlas para que envíen peticiones ICMP de respuesta a la victima al mismo tiempo. Un esquema de este ataque se muestra en la figura 6-2.

6.3. Ping of Death

El ataque así llamado, ping de la muerte, consiste en enviar paquetes ICMP echo request muy grandes. Algunos sistemas vulnerables a este ataque se caen al recibirlos. Linux no es vulnerable a este tipo de ataque, y tampoco lo son los sistemas Unix actuales. Sin embargo, si en la red existen otros equipos más antiguos, estos podrían serlo.

El ping de la muerte da una idea de como los protocolos más simples o los mensajes en apariencia más inofensivos pueden ser usados por un atacante con imaginación. No todos los ataques son intentos de entrar en el sistema de la víctima. Algunos son simplemente destructivos. En estos casos, el objetivo es tumbar la máquina víctima.

El comando ping es una utilidad muy básica y útil para la comprobación de las conexiones de red. Podría no ser deseable deshabilitarlo. Los expertos en seguridad recomiendan deshabilitar las respuestas a las peticiones de eco, filtrar los paquetes ICMP echo request entrantes en los firewalls o, al menos, limitar estos paquetes a las máquinas de confianza.

Sin embargo, descartar el paquete de petición de

respuesta no es una solución para la máquina víctima. No importa como los equipos traten este tipo de paquetes, el sistema siempre podrá ser sobrecargado con el procesado, inspección y descarte de una multitud de peticiones.

6.4. UDP Flooding

El protocolo UDP puede ser usado como una útil herramienta de ayuda para realizar ataques de denegación del servicio. Al contrario que el protocolo TCP, UDP es un protocolo sin estado. Es decir, al ser no orientado a conexión, no está dotado de mecanismos de control de flujo. No posee indicadores del estado de la conexión ni números de secuencia del datagrama, por lo que no se mantiene información acerca del siguiente paquete que se espera. En el protocolo UDP, no siempre existe una forma de distinguir los paquetes del cliente de los paquetes del servidor basada en los números de puerto. Sin la información del estado, no hay manera de distinguir un paquete que llega como respuesta a una solicitud previa de un paquete que llega de forma no esperada y que no se ha solicitado. Por estas razones, resulta relativamente sencillo mantener a un sistema ocupado atendiendo a peticiones UDP entrantes y no dejar ancho de banda para el tráfico de red legítimo.

Dado que los servicios UDP son inherentemente menos seguros que los servicios TCP, muchos sitios deshabilitan todos los puertos UDP que no son absolutamente imprescindibles. Casi todos los servicios más comunes de internet están basados en el protocolo TCP. Será, por tanto, una buena política, especificar en el firewall de la red, que solo se permita la entrada a los paquetes UDP destinados a servicios necesarios que se vayan a ofrecer solo a las máquinas en las que se confía.

El ataque clásico de inundación UDP o bien involucra a dos máquinas víctimas o funciona de la misma forma que el ataque smurf (ping flooding). Esto se muestra en la figura 6-3. Un solo paquete falsificado desde el puerto UDP echo del atacante, dirigido al puerto UDP chargen de una de las victimas puede llevar a un bucle infinito de tráfico en la red. El servicio echo devuelve en un mensaje de respuesta los datos recibidos. El servicio chargen genera una cadena ASCII y descarta los datos recibidos.

6.5. Bombas de Fragmentación

Las diferentes tecnologías usadas bajo la capa de red, en el nivel de enlace (Ethernet, ATM, Token Ring) definen el tamaño máximo de los paquetes en ese nivel. Cuando un paquete pasa de un router a otro en su camino desde la máquina origen hasta la destino, los routers pasarelas de red podrían necesitar dividir el paquete en partes más pequeñas, a las que llamamos fragmentos, antes de enviarlos a la siguiente red. Bajo fragmentación normal o legítima, el primer fragmento contiene los puertos origen y destino normales contenidos en la cabecera UDP o TCP de transporte. Los fragmentos siguientes no contienen tal información. Para ilustrar esto, pensemos que el tamaño máximo teórico de un paquete es de 65535 bytes, mientras que el tamaño máximo de una trama Ethernet es de 1500 bytes.

Cuando un paquete se fragmenta, los routers intermedios en el camino no lo recomponen antes de volverlo a enviar. Los paquetes se recomponen en la máquina destino o en el router adyacente.

Debido a que a fin de cuentas la fragmentación intermedia es más costosa que el envío de paquetes más pequeños, normalmente los sistemas fijan el valor de MTU (unidad máxima de transmisión) de acuerdo con el otro extremo en la negociación de la conexión. Esto se lleva a cabo enviando un paquete con la opción "Don't fragment" activada en el campo de opciones de la cabecera del datagrama IP. En estas condiciones, si un router intermedio necesita fragmentar el paquete para reenviarlo, envía un mensaje ICMP fragmentation required al origen.

Un tipo de ataque mediante bombas de fragmentación consiste en construir artificialmente paquetes muy pequeños. Paquetes de un byte hacen que muchos sistemas operativos caigan. Los sistemas operativos actuales verifican esta condición.

Otro uso de fragmentos pequeños es el de construir el fragmento inicial de forma que los puertos TCP origen y destino estén contenidos en el segundo fragmento (los valores de MTU de todas las redes son suficientes para llevar los 40 bytes de las cabeceras normales IP y TCP). Los firewalls normalmente permiten el paso de estos paquetes porque la información en la que se basan para realizar el filtrado no está presente en ellos. Esta forma de ataque es útil para conseguir que estos paquetes

crucen el firewall, ya que de otra forma no lo conseguirían.

El ataque de denegación del servicio llamado Ping of Death, comentado más arriba, es un ejemplo del uso de la fragmentación para transportar un mensaje ICMP ilegalmente sobredimensionado. Cuando se reconstruye la petición de echo, el paquete tiene un tamaño total mayor que 65535 bytes, haciendo que algunos sistemas se cuelguen.

Un ejemplo clásico de ataque mediante fragmentación es el ataque Teardrop. Este método se puede usar para saltarse las reglas de un firewall o para tumbar una máquina. El primer fragmento se construye para ser enviado a un servicio permitido. Como la mayoría de los firewalls no inspeccionan los fragmentos después de que el primero haya pasado, si el primero es aceptado, los siguientes paquetes pasarán también y serán reensamblados en la máquina destino. Si el paquete inicial es descartado, el resto de los paquetes pasarán, aunque en la máquina destino no habrá nada que reconstruir y posiblemente se descarten los fragmentos.

El campo data offset en los fragmentos siguientes puede ser alterado para sobre escribir la información del puerto destino y así acceder a servicios no permitidos. El offset también puede ser modificado para que los offsets usados en el reensamblado de los paquetes se conviertan en números negativos. Como las rutinas de copia de bytes del kernel usan normalmente números sin signo, el número negativo sería tratado como un número positivo muy grande, y la copia resultante ensuciaría la memoria del kernel, haciendo caer a la máquina.

Las máquinas que actúan de firewall y las que realizan la traducción de direcciones de red (NAT) deben ser configuradas para que realicen el reensamblado de paquetes antes de entregarlos a los objetivos locales. La traducción de direcciones se tratará con mayor detalle en el capítulo dedicado a firewalls.

6.6. Buffer Overflows

Los ataques de buffer overflow se agrupan en dos categorías. La primera consiste en hacer que un sistema caiga, sobreescribiendo los datos en la pila de ejecución. La segunda requiere conocimientos especiales del hardware y del software del sistema que se vaya a atacar. El propósito del overflow es

escribir en la pila de ejecución del programa de forma que la pila de retorno contenga un programa y éste pase a ejecución. Este programa ejecutará normalmente un shell con privilegios de root.

Los scripts CGI usados en muchas aplicaciones web, son especialmente vulnerables, a menos que se tomen precauciones. Muchas de las vulnerabilidades de los servidores son resultado de un buffer overflow. Es importante instalar y tener al día las ultimas versiones y actualizaciones de software.

6.7. Bombas ICMP Redirect

El mensaje ICMP redirect pide al sistema al que se envía que cambie sus tablas de rutas internas para que se le asigne una ruta más corta. Estos mensajes son enviados por los routers a las máquinas adyacentes. La intención es informar de que una ruta más corta está disponible.

Si una máquina Linux funciona de router usando routed o gated, cualquier mensaje ICMP redirect se ignorará en condiciones normales. El resto de máquinas deben obedecer estos mensajes y añadir la nueva ruta a su caché.

Los mensajes ICMP redirect llegan casi todos los días a las máquinas. Rara vez llegan creadas por el router adyacente. Para los sitios conectados a un proveedor de servicios de internet, es muy improbable que el router adyacente genere un mensaje de este tipo y lo envíe a esos sitios.

Si un equipo usa encaminamiento estático y obedece a los mensajes ICMP redirect, es posible que alguien engañe al sistema haciendole creer que una máquina remota pertenece a la red local o a la red del proveedor de servicios de internet, o incluso hacer que reenvíe todo el tráfico a alguna otra máquina remota.

6.8. Otros ataques de denegación del servicio

La operatividad de la red no es el único ámbito en el que se producen ataques de denegación del servicio. Por ejemplo, el sistema de ficheros puede caer en overflow si se le obliga a escribir gran cantidad de mensajes de error en los registros de error, o si el sistema es inundado con muchas copias de e-mails extremadamente grandes. Por estas razones deben configurarse limites en la asignación de recursos y preparar una partición separada para las aplicaciones que puedan generar un crecimiento o cambio rápido en el sistema de archivos.

Por otro lado, la memoria de sistema, las entradas en las tablas de procesos, ciclos de proceso y otros recursos, pueden ser rápidamente agotados mediante la invocación rápida y repetida de servicios de red. Ante esto, poco se puede hacer que no sea asignar recursos limitados para cada servicio individual, habilitar las SYN cookies en el kernel y descartar en lugar de rechazar paquetes enviados a puertos no ofrecidos.

Capítulo 7
FIREWALLS

Cuando una red debe conectarse a otra (normalmente internet), el número de posibles atacantes aumenta considerablemente, por lo que deberemos dotar a la red de mayores medidas de seguridad en el punto de conexión con el exterior. Es aquí donde entra en juego un firewall o cortafuegos.

7.1. ¿Qué es un cortafuegos?

Según la definición del RFC 2647, "Benchmarking Terminology for Firewall Performance", un firewall es "un dispositivo, un servicio del sistema operativo, o aplicación que asegura las políticas de control de acceso entre redes". Más concretamente, el término firewall tiene muchos significados diferentes dependiendo de los mecanismos utilizados para implementarlo, el nivel en la pila TCP/IP en que el firewall opera y las arquitecturas de la red y rutado usadas. Los significados más comunes se refieren a un filtro de paquetes sin estados, un router con capacidades de filtrado, un firewall dinámico o con estados, una pasarela de nivel de aplicación y un proxy.

Un filtro de paquetes sin capacidades de estado es normalmente implementado en el núcleo del sistema operativo o en el hardware del router y funciona a nivel de la capa de red IP o en la capa TCP de transporte. La protección que proporciona se basa en las decisiones tomadas en función de un análisis de las cabeceras de los paquetes. Estas decisiones se toman paquete a paquetes. En sistemas que soportan una gran

cantidad de tráfico se suele situar un filtro de paquetes antes del firewall para liberar a éste último de esfuerzo de proceso.

Un filtro de paquetes con capacidades de estado mantiene información acerca de la conexión TCP o el intercambio UDP. Los paquetes se analizan dentro de su entorno en lugar de hacerlo individualmente. Por ejemplo, un filtro de paquetes de este tipo podría desechar un paquete TCP con el bit ACK activo si antes de él no ha recibido un paquete con el bit SYN, con lo que estaríamos evitando un escaneo ACK. Un efecto colateral de mantener el estado de la conexión es la posibilidad de saltar las reglas de filtrado si el paquete es identificado como parte de una conexión establecida y permitida. Los filtros de paquetes dinámicos necesitan a menudo para funcionar satisfactoriamente información acerca las aplicaciones que escuchan detrás de los puertos TCP.

Cuando hablamos de un proxy de aplicación, la pasarela de nivel de aplicación es el punto de terminación para ambos extremos de la conexión. El cortafuegos proxy se implementa como aplicaciones separadas para cada servicio en el proxy. La pasarela de nivel de aplicación entiende el lenguaje de la aplicación especifica y puede realizar un análisis del tráfico del nivel de aplicación. Las aplicaciones proxy pueden asegurar la integridad de los datos, es decir, que los datos son apropiados para el servicio, libres de virus, etc. La pasarela mantiene dos conjuntos de conexiones: unas con el cliente y otras con el servidor. Cada aplicación proxy es vista como servidor por el cliente, y como el cliente para el verdadero servidor. La pasarela inicia la conexión en representación del cliente. Del mismo modo, las respuestas del servidor terminan en la pasarela. Esta responde al cliente directamente, en representación del servidor. De este modo, el tráfico local nunca sale al exterior ni el tráfico externo entra en la red local.

En cualquier caso, el objetivo de un firewall es asegurar el cumplimiento de las políticas de acceso que se definan: quien puede usar los servicios y archivos del sistema, y qué tráfico puede pasar de la red local a internet y viceversa.

Entre los ataques de los cuales puede proteger un firewall de filtrado de paquetes está el spoofing de direcciones, escaneado de puertos, ataques de denegación del servicio, acceso a servicios locales privados. En general, proporcionará un nivel de protección adicional sobre una configuración local poco segura.

Las máquinas que actúan de firewalls pueden además proveer otros servicios de seguridad. Entre estos servicios se encuentra la traducción de direcciones (NAT), el chequeo de virus, filtrado de URLs, autenticación de usuario y encriptado del tráfico.

Los firewalls son normalmente puntos únicos de conexión entre redes de área local e internet. De este modo, toda la información que se transfiera entre ambas redes deberá pasar por él, con lo que el control sobre los paquetes puede ser tan estricto como se defina en las reglas del firewall.

Los cortafuegos actuales realizan distintas tareas, entre las que se encuentran:

- Análisis y filtrado de paquetes. Pueden analizar paquetes entrantes y salientes de múltiples protocolos. Basandose en este análisis realizan acciones, que pueden descartarlo, rechazarlo enviando un mensaje de error al origen del mismo, dejarlo pasar, etc.

- Bloqueo de protocolo y contenido. Se puede configurar un firewall para bloquear Java, Javascript, VBScript, etc. Se puede también configurar para bloquear firmas de ataque particulares.

- Autenticación y encriptación de usuario, conexión y sesión. Muchos firewalls usan algoritmos y sistemas de autenticación (DES, SSL, MD5, etc) para verificar la integridad de las sesiones y de los usuarios, y proteger los datos en transito de los escaneos.

En realidad, podríamos pensar en un firewall como el conjunto de reglas que definen las políticas de acceso a la red que protege.

Existen dos tipos principales de firewalls: firewalls de nivel de red o filtros de paquetes y pasarelas de aplicaciones. Estudiaremos aquí los filtros de paquetes por encontrarse las pasarelas de aplicaciones por encima del nivel de transporte, y quedar fuera de este estudio.

7.2. Filtros de paquetes

Un firewall de filtro de paquetes consiste en un conjunto de reglas para aceptar y rechazar determinados paquetes. Estas reglas definen explícitamente qué paquetes si y qué paquetes no están permitidos a través de la interfaz de red. Esas reglas usan los campos de cabecera de los paquetes para decidir si los reenvían a su destino, los descartan silenciosamente o los bloquean y envían un mensaje con un código de error al origen.

El firewall filtra de forma independiente lo que sale de la red y lo que entra a ella. Es decir, el filtrado de entrada y el de salida pueden estar regidos por reglas completamente distintas. Cada lista de reglas que definen lo que puede entrar y lo que puede salir se llama cadena o *chain*. Se llaman así porque cada paquete se compara con cada regla hasta que cumpla alguna o se agoten todas las reglas.

Las ideas relacionadas con los firewalls a menudo llevan a conclusiones equivocadas, aquellas de los que piensan que un filtro de paquetes en sí mismo puede responsabilizarse de la seguridad de un sistema. Nada más lejos de la realidad. Un filtro de paquetes es sólo una capa de la arquitectura completa de seguridad. Un filtrado de paquetes es una función de muy bajo nivel como para ocuparse de ataques elaborados. Es fácil darse cuenta de esto si tenemos en cuenta que, por ejemplo, el protocolo IP no cuenta con formas de autentificar que el remitente es quien dice ser, dado que la información disponible sobre la identidad del origen del paquete es el campo origen de la cabecera IP. Ni la capa de red ni la de transporte puede verificar que los datos del nivel de aplicación son correctos. Sin embargo, los niveles a los que trabajan los filtros de paquetes proveen un mayor y más simple control sobre el acceso directo a los puertos. El control sobre el contenido de los paquetes y los protocolos de comunicación pueden ser fácilmente realizados en capas más altas.

Sin el control de bajo nivel que ofrecen los filtros de paquetes, posiblemente el filtrado de alto nivel y la seguridad de los proxies serían inefectivas. Hasta cierto punto, estos deben asentarse en la confianza sobre los protocolos de comunicación inferiores.

Política de filtrado por defecto

Cada cadena de un firewall tiene una política de filtrado por defecto, y un conjunto de acciones a tomar en respuesta a tipos de paquetes específicos. Cada paquete se compara con cada regla en la cadena, y si no se cumple ninguna de ellas, se toma la acción predeterminada. Esta es la política por defecto.

Hay dos formas de actuar básicas en un firewall:

- Denegar el acceso a todos por defecto y seleccionar explícitamente qué paquetes pasarán.

- Aceptar todo por defecto y seleccionar explícitamente a qué paquetes se deniega el acceso.

Denegar a todos es la política por defecto recomendada. Usando esta política predeterminada es más sencillo construir un firewall seguro, pero tiene el inconveniente de que cada servicio y su transacción de protocolo asociada debe estar explícitamente permitida. Por lo tanto, para cada servicio que se vaya a permitir, el administrador debe conocer y entender el protocolo de comunicación. Con la política por defecto de denegar a todos, requiere más trabajo habilitar el acceso a internet, pero a cambio, suele dejar menos cabida a errores en la configuración de las cadenas del firewall.

La política predeterminada de aceptar todo facilita la tarea de puesta en marcha del sistema, pero obliga a anticiparse a todos los accesos imaginables a los que no se les quiera conceder el acceso. El peligro de esta política es que nadie se puede anticipar a un acceso no deseado hasta que éste se produce. En definitiva, a la larga, el desarrollo de un firewall seguro con una política predeterminada de aceptarlo todo requiere más trabajo, es mucho más difícil y mucho menos fiable.

Rechazar frente a denegar el acceso

La mayoría de los filtros de paquetes actuales ofrecen la posibilidad de denegar el acceso a un paquete o rechazarlo. La diferencia entre estas dos acciones estriba en si se envía o no al origen una notificación de error. Cuando un paquete se rechaza, se envía al origen una notificación de error, indicando que el paquete no se ha entregado. Por el contrario, al denegar el

acceso a un paquete, simplemente se descarta silenciosamente, sin enviar ningún mensaje al origen. Esta política suele ser, por varias razones, la recomendada en la mayor parte de los casos. La primera es que la respuesta dobla el tráfico de la red. La mayor parte de los paquetes no aceptados, no lo son porque son malintencionados, no porque representan un intento inocente de acceso a un servicio que casualmente no es ofrecido por el sistema. Por otro lado, cada paquete al que se responde puede ser usado en un ataque de denegación del servicio. Por último, cualquier respuesta, incluso un mensaje de error, da al eventual atacante información útil.

Filtrado de paquetes entrantes

La cadena de entrada de un firewall es la más interesante para la seguridad de la red que protege el cortafuegos. El filtrado, como se ha dicho anteriormente se puede hacer atendiendo a la dirección origen, la dirección destino, el puerto origen, el puerto destino y los indicadores de estado TCP.

Filtrado basado de la dirección remota origen

En el nivel de paquete, la única forma de identificar quién envía un paquete es a través del campo de dirección origen de la cabecera del datagrama. Esto posibilita la falsificación de direcciones origen (spoofing). El hecho es que un atacante puede poner una dirección incorrecta en el campo fuente en lugar de la suya verdadera; esa dirección puede ser una dirección que no existe o, por el contrario, puede ser una dirección legítima que pertenece a alguien. Estos ataques pueden llevar a que alguien envíe paquetes a nuestro sistema con direcciones locales, haciendo que parezca un paquete del entorno local, un paquete en el que se confía. Puede provocar también que los atacantes usen la identidad de otros cuando realizan intrusiones. Y lo peor, hacer creer que el origen de los paquetes que llegan es el mismo sistema.

Es importante tener en cuenta que normalmente no se pueden detectar las direcciones suplantadas. Las direcciones pueden ser legitimas y perfectamente encaminables pero no pertenecer al origen del paquete. Sin embargo, en ciertos casos, es posible detectarlas. Cuando los paquetes que vengan del exterior tengan como dirección origen una de las siguientes,

debemos denegar el acceso, ya que son direcciones ilegales. Estas son:

- La dirección IP de la red que pretende proteger el firewall. Por motivos obvios, en condiciones normales, nunca llegará a la interfaz de red un paquete legal cuya dirección origen sea la de la propia interfaz. Por tanto, estos paquetes deben ser descartados.

- Direcciones de la red local. Rara vez llegarán paquetes legales externos a la interfaz de red diciendo que son enviados desde el interior de la red local. Esto es posible en redes de área local con varios puntos de acceso a internet, pero en cualquier caso será fruto de una mala configuración local de la red. En la mayoría de los casos, un paquete de esas características será un paquete malintencionado parte de un intento de obtener acceso al sitio basandose en las relaciones locales de confianza.

- Direcciones Privadas de clase A, B y C. Estos tres conjuntos de direcciones tienen rangos reservados para el uso en redes privadas de área local. Su uso no está pensado para internet. Como tales, las direcciones pertenecientes a uno de esos grupos pueden ser usadas internamente en cualquier red de área local sin la necesidad de comprar direcciones IP registradas. Nunca deben llegar paquetes legales entrantes de ninguna de estas direcciones. Los rangos asociados a cada clase son:

  Clase A: desde 10.0.0.0 hasta 10.255.255.255

  Clase B: desde 172.16.0.0 hasta 172.31.255.255

  Clase C: desde 192.168.0.0 hasta 192.168.255.255

- Direcciones de difusión de clase D. Las direcciones del rango de la clase D se reservan para el uso como direcciones destino cuando se participa en una red de difusión como pueden ser la transmisión de video o audio, por ejemplo en una cadena de radio por internet. Por tanto, el firewall no debe dejar pasar nunca paquetes cuyo campo de dirección fuente sea una de estas direcciones. El rango de estas direcciones va desde 224.0.0.0 hasta 239.255.255.255.

- Direcciones de clase E reservadas. Estas direcciones se reservaron para uso futuro y experimental, no para uso público. Por tanto, el firewall no deberá dejar pasar paquetes con un campo de dirección fuente que pertenezca a este rango. Dicho rango comienza en 240.0.0.0 y termina teóricamente en 247.255.255.255, aunque queda reservado el rango de direcciones IP hasta 255.255.255.255, por lo que en algunos libros se toma el rango de direcciones de clase E desde 240.0.0.0 hasta 255.255.255.255.

- Direcciones de retorno local. La interfaz de retorno es una interfaz de red usada por los sistemas tipo Unix para servicios de red locales. El tráfico dirigido a esta interfaz es, por definición, enviado por el sistema que lo genera, por lo que no debemos encontrarnos con paquetes con este origen intentando entrar en nuestra red. El rango de direcciones de retorno va desde 127.0.0.0 hasta 127.255.255.255.

- Direcciones de broadcast usadas fuera de contexto. Las direcciones de broadcast son direcciones aplicadas a todas las máquinas en una red. La dirección 0.0.0.0 es una dirección de broadcast origen especial. Una dirección de broadcast origen será o bien 0.0.0.0 o bien una dirección IP normal. Los clientes y servidores DHCP verán paquetes de broadcast entrantes desde la dirección origen 0.0.0.0. Éste es el único uso legal de esta dirección origen. No es una dirección legítima para comunicaciones punto a punto. Si se ve como dirección origen en un paquete normal, no broadcast, punto a punto, dicha dirección esta falsificada.

- Direcciones 0.x.x.x de redes de clase A. Como se dijo más arriba, cualquier dirección fuente en el rango 0.0.0.0 a 0.255.255.255 es ilegal en comunicaciones punto a punto.

- Direcciones de red de enlace local. Los clientes DHCP a veces se asignan a sí mismos una dirección de enlace local cuando no pueden conseguir una dirección del servidor. Estas direcciones están contenidas en el rango 169.254.0.0 a 169.254.255.255.

- Direcciones de redes de prueba. El espacio de direcciones desde 192.0.2.0 hasta 192.0.2.255 está reservado para

redes de prueba.

Existen otros bloques de direcciones reservados por IANA (Internet Assigned Numbers Authority). No es mala idea filtrar estas direcciones. Pero dado que estos bloques cambian de rango bastante dada la escasez de direcciones IP y el crecimiento de los sitios de internet, es necesario estar informado acerca de los cambios que se producen en los rangos de direcciones reservados, para evitar bloquear direcciones legítimas.

Filtrado de sitios problemáticos

Otro tipo de filtrado común, aunque menos usado es el bloqueo de todo acceso que se intente llevar a cabo desde una determinada máquina. Este tipo de filtrado es el que se tiende a hacer cuando un sitio de internet desarrolla una reputación de mal vecino; el resto de sitios lo bloquean.

Limitar los paquetes entrantes a determinados hosts

Es posible definir reglas en el firewall que permitan la entrada a ciertos paquetes procedentes de direcciones IP específicas o de un rango limitado de direcciones IP origen.

Un tipo de estos paquetes entrantes está formado por aquellos paquetes procedentes de servidores remotos que están respondiendo a peticiones de máquinas locales. Por ejemplo, dentro de este grupo se encuentran los paquetes DHCP de asignación dinámica de dirección IP enviados por el servidor de un proveedor de servicios de internet cuando un usuario intenta conectar a internet.

Otra clase de estos paquetes entrantes son los enviados por clientes remotos que quieren acceder a los servicios ofrecidos en el sitio local. A pesar de que la procedencia de algunas conexiones entrantes puede esperarse que sea cualquier sitio, por ejemplo en el acceso a un servidor web, otras, sin embargo, solo se ofrecerán a unos pocos usuarios en los que se confía. Ejemplos de estos servicios son telnet y ssh.

Filtrado basado de la dirección local destino

El filtrado basado en la dirección local destino no es algo que deba preocupar en exceso. Esto es debido a que, bajo condiciones normales de operación, las interfaces de red ignoran los paquetes que no van dirigidos hacia ellas (hemos visto en el capitulo sobre escuchas electrónicas que si funcionan en modo promiscuo no, pero vamos a asumir que las interfaces funcionan en modo normal). La excepción la plantean los paquetes de broadcast, que deben ser enviados a toda la red.

La dirección 255.255.255.255 es la dirección destino general para broadcast. Con ella se hace referencia a todas las máquinas en el segmento físico de red, y se llama broadcast limitado. Las direcciones de broadcast se pueden definir de forma más explicita como una dirección de red, seguida por 255 en cada octeto que quede libres hasta completar la dirección IP. Por ejemplo, si la dirección de una red de área local es 192.168.0.0, la dirección de broadcast de dicha red será 192.168.255.255, independientemente de que los equipos pertenezcan al segmento o no.

Con la dirección destino de broadcast 0.0.0.0 tenemos algo similar a la situación en la que paquetes en conexiones punto a punto dicen venir de la dirección de la dirección fuente de broadcast que comentábamos más arriba. Aquí, los paquetes de broadcast son dirigidos a la dirección fuente 0.0.0.0 en lugar de a su dirección destino, 255.255.255.255. En este caso, hay una pequeña cuestión acerca de la intención del paquete. Esto es un intento de identificar si un sistema es un máquina Unix. Por razones históricas, las máquinas BSD Unix y sus derivados envían un error ICMP de tipo 3 en respuesta al uso de la dirección 0.0.0.0 como dirección de destino de broadcast. Otros sistemas operativos descartan silenciosamente el paquete. Este es un buen ejemplo de por qué es mejor descartar silenciosamente los paquetes que rechazarlos. En este caso, el mensaje de error es precisamente lo que el paquete buscaba.

Filtrado basado en el puerto origen

El puerto origen en los paquetes entrantes identifica a la aplicación que envía el mensaje (al menos, esa es la intención, aunque no se pueda garantizar). Generalmente, todas las peticiones entrantes de los clientes remotos de los servicios

ofrecidos siguen el mismo patrón, y todas las respuestas entrantes de los servidores remotos a los clientes locales siguen un patrón diferente.

Las peticiones y conexiones de clientes remotos al servidor local pueden tener un puerto fuente dentro de un rango de puertos no privilegiados. Por ejemplo, todas las conexiones entrantes a un servidor web deben tener puertos fuentes desde 1024 hasta 65535.

Las respuestas entrantes y conexiones desde servidores remotos que se hayan contactado desde el área local tienen el puerto origen que se haya asignado al servicio concreto. Si se realiza una conexión a un servidor web remoto, todos los mensajes entrantes del servidor remoto tendrán como puerto origen el 80, el puerto asignado al servicio http.

Filtrado basado en el puerto local destino

El puerto destino en los paquetes entrantes identifica el programa o servicio al que el paquete va dirigido. Como con el puerto fuente, todas las peticiones entrantes de clientes remotos a los servicios locales siguen generalmente el mismo patrón, y todas las respuestas de servicios remotos a los clientes locales siguen un patrón distinto.

Las peticiones y conexiones entrantes de clientes remotos a los servidores locales pondrán en el puerto destino el número de puerto asignado al servicio concreto. Un paquete entrante destinado al servidor web local tendrá el puerto destino establecido como 80.

Las respuestas entrantes desde servidores remotos que se contacten tendrán un puerto destino en el rango de puertos no privilegiados. Si se realiza una conexión a un sitio web remoto, todos los mensajes entrantes desde el servidor remoto tendrán un puerto destino en el rango de 1024 a 65535.

Filtrado basado en el estado de la conexión TCP

Las reglas de aceptación de los paquetes TCP pueden hacer uso de los indicadores de estado de la conexión asociados con las conexiones TCP. Todas las conexiones TCP se adhieren al mismo

conjunto de estados de la conexión. Estos estados difieren entre cliente y servidor a causa del protocolo de establecimiento de conexión "three-way handshake". Como tal, el firewall puede distinguir entre tráfico entrante desde clientes remotos y tráfico entrante desde servidores remotos.

Los paquetes TCP entrantes procedentes de clientes remotos tendrán el indicador SYN activado en el primer paquete recibido como parte del protocolo de establecimiento a tres bandas. La primera petición de conexión tendrá el indicador SYN activado, pero no el indicador ACK. Todos los paquetes después de la primera petición de conexión tendrán solo el indicador ACK activado. Las reglas del firewall para el servidor local permitirán paquetes entrantes, sin tener en cuenta el estado de los indicadores SYN o ACK.

Los paquetes entrantes de los servidores remotos serán siempre respuestas a la petición inicial de conexión iniciada por los programas clientes locales. Cada paquete recibido de un servidor remoto tendrá el indicador ACK activado. Las reglas del firewall para el cliente local requerirán que todos los paquetes entrantes desde los servidores remotos tengan el indicador ACK activo, pero nunca el indicador SYN. Los servidores legales no suelen intentar iniciar conexiones con programas cliente.

Sondeos y escaneos

Un sondeo a un puerto es un intento de conexión o de obtención de respuesta de un servicio que opera en un puerto concreto. En escaneo de puertos es un conjunto de sondeos a puertos. Los escaneos se realizan normalmente de forma automatizada, como ya vimos en el capitulo dedicado a scanners, y también vimos que existen reglas que podemos incluir en el firewall para bloquearlos. No obstante, los escaneos no son dañinos en sí mismos, pero son normalmente el punto de partida para un ataque posterior.

Paquetes encaminados por la fuente

Los paquetes encaminados por la fuente usan una muy poco usada opción IP que permite al origen definir la ruta seguida entre dos máquinas, en lugar de dejar que los encaminadores intermedios determinen la ruta. Como en el caso de las redirecciones ICMP, esta característica puede permitir que

alguien engañe al sistema haciendole pensar que se comunica con una máquina local, una máquina del proveedor de servicios de internet, o cualquier otra máquina de confianza, o creando el tráfico de paquetes necesario para un ataque "man-in-the-middle".

El encaminamiento realizado por la fuente tiene pocos usos legítimos en las redes actuales. Algunos encaminadores ignoran esta opción. Algunos firewalls descartan los paquetes que contienen esta opción.

Filtrado de paquetes salientes

Si el entorno a proteger representa un entorno de confianza, el filtrado de paquetes salientes puede no parecer tan crítico como el filtrado de paquetes entrantes. El sistema no responderá a mensajes entrantes que el firewall no haya dejado pasar. Los sitios pequeños a menudo adoptan esta solución. Sin embargo, incluso para los sitios pequeños o domésticos el filtrado simétrico es importante, especialmente si el firewall protege máquinas con Microsoft Windows.

Si el firewall protege una red de área local formada por máquinas Microsoft Windows, controlar el tráfico saliente se convierte en algo mucho más importante. Las máquinas Windows son muy abundantes, y por ello, múltiples troyanos abundan para ellas. Se han dado casos en los que miles de máquinas Windows comprometidas han trabajado de forma coordinada en ataques de denegación del servicio sin que los usuarios lo supieran. Por esta razón especialmente es importante filtrar lo que sale del entorno local.

El filtrado de mensajes salientes también permite ejecutar servicios en la red local sin que haya escapes a internet. No solo es cuestión de evitar el acceso externo a los servicios locales, sino de evitar que la información local salga al exterior.

Otro aspecto es el de bloquear travesuras intencionadas originadas en las máquinas locales.

En resumen, la principal razón para filtrar los paquetes salientes es la de conservar el tráfico local dentro del área local.

Filtrado basado en la dirección origen local

El filtrado de paquetes salientes basado en la dirección origen es sencillo. Para un sitio pequeño o una única máquina conectada a internet, la dirección fuente es siempre, bajo condiciones normales, la dirección IP del sitio o de la computadora. No hay razón para permitir un paquete saliente con cualquier otra dirección de origen, y el firewall debe asegurar que no ocurre.

Para las máquinas que obtienen su dirección IP de forma dinámica de su proveedor de servicios de internet, existe una pequeña excepción durante la asignación de la dirección. La excepción es específica de DHCP y es el único caso en el que una máquina envía mensajes de broadcast usando la dirección 0.0.0.0 como su dirección origen.

Para las redes LAN cuya máquina firewall tiene una dirección IP asignada dinámicamente, limitar los paquetes salientes para contener la dirección de la máquina firewall es obligatorio. Esto protege de varios errores de configuración que aparecen como casos de falsificación de direcciones o direcciones origen ilegales a los equipos remotos.

Si los usuarios o el software que usan no es de total confianza, es importante asegurar que el tráfico local contiene solamente direcciones locales legítimas y encaminables, para evitar la participación involuntaria en ataques de denegación de servicio mediante falsificación de direcciones.

Filtrado basado en la dirección destino remota

Como en el caso de los paquetes entrantes, puede ser necesario permitir ciertos tipos de paquetes salientes, que sean direccionados solamente a redes remotas o máquinas específicas. En estos casos, las reglas del firewall definirán tanto las direcciones IP especificas o el rango de direcciones IP destino para las que se permitirá la salida de paquetes.

La primera clase de paquetes salientes que se deben filtrar según su dirección destino es la de los paquetes destinados a servidores remotos que se han contactado. A pesar de que algunos paquetes, tales como aquellos destinados a servidores web o FTP, normalmente tendrán como destino cualquier

dirección de internet, otros servicios remotos solamente serán ofrecidos de forma legítima por el proveedor de servicios de internet o determinados equipos en los que se confía. Ejemplos de estos servicios son el correo POP o el servicio DHCP.

La segunda clase de paquetes salientes que se deben filtrar es la constituida por los paquetes destinados a clientes remotos que acceden a servicios ofrecidos por el sistema local. De nuevo, a pesar de que algunas conexiones salientes a servicios, como respuestas del servidor web al cliente remoto, puede esperarse que vayan dirigidas hacia cualquier parte, otros servicios locales se ofrecerán solo a unos pocos sitios remotos en los que se confía. El firewall no solo denegará el acceso de las conexiones entrantes a estos servicios, sino que las reglas tampoco permitirán respuestas de estos servicios a nadie.

Filtrado basado en el puerto local origen

Definir explícitamente qué puertos se pueden usar para las conexiones salientes tiene dos objetivos. Por un lado, especificar los puertos permitidos para las conexiones salientes ayuda a asegurar que las aplicaciones funcionan correctamente, y protege a los demás de cualquier trafico de red que no pertenece a internet.

Las conexiones salientes de los clientes locales serán siempre originadas desde un puerto no privilegiado. Limitar los clientes a los puertos no privilegiados en las reglas del firewall ayuda a proteger a otra gente de errores potenciales en el extremo local asegurando que las aplicaciones locales se comportan como se espera de ellas.

Los paquetes salientes de las aplicaciones servidoras locales serán originados siempre desde su puerto asignado. Limitar a los servidores a sus puertos asignados en las reglas del firewall asegura que las aplicaciones servidoras están funcionando correctamente, al menos a nivel de protocolo. Y más importante es que de esta forma se protege cualquier servicio privado y local que se pueda estar ejecutando desde un acceso externo. También ayuda a proteger los sitios remotos de ser molestados con tráfico de red que no debería haber salido de la red local.

Filtrado basado en el puerto destino remoto

Las aplicaciones locales cliente están diseñadas para conectar a servidores de red que ofrecen sus servicios en los puertos asignados. Desde este punto de vista, limitar a los clientes locales a conectar solo con los puertos de sus servidores asociados asegura el correcto funcionamiento del protocolo. Limitar las conexiones de los clientes a puertos destino específicos sirve además para la protección contra aplicaciones locales privadas que intentan acceder inadvertidamente a servidores en internet, y por otro lado inhabilita posibles errores enviados al exterior, escaneo de puertos y otros ataques potenciales desde el sitio local.

Los servidores locales casi siempre participaran en el establecimiento de conexiones originadas desde puertos no privilegiados. Las reglas del firewall deberán limitar el trafico saliente de los servidores solo a puertos no privilegiados.

Filtrado basado en el estado de la conexión TCP saliente

Las reglas de aceptación de los paquetes TCP salientes pueden hacer uso de los indicadores de estado de la conexión asociados con las conexiones TCP, del mismo modo en que lo hacen las entrantes. Todas las conexiones TCP se adhieren al mismo conjunto de estados de la conexión. Estos estados difieren entre cliente y servidor a causa del protocolo de establecimiento de conexión "three-way handshake".

Los paquetes TCP salientes procedentes de clientes locales tendrán el indicador SYN activado en el primer paquete enviado como parte del protocolo de establecimiento a tres bandas. La primera petición de conexión tendrá el indicador SYN activado, pero no el indicador ACK. Todos los paquetes salientes después de la primera petición de conexión tendrán solo el indicador ACK activado. Las reglas del firewall para el servidor local permitirán paquetes salientes, sea cual sea el estado de los indicadores SYN y ACK.

Los paquetes salientes de los servidores locales serán siempre respuestas a la petición inicial de conexión iniciada por los programas clientes remotos. Cada paquete recibido de un

servidor local tendrá el indicador ACK activado. Las reglas del firewall para el servidor local requerirán que todos los paquetes salientes desde los servidores locales tengan el indicador ACK activo.

Servicios de red privados y públicos

Una de las formas más sencillas en que se pueden permitir intrusiones no deseadas es permitiendo el acceso externo a servicios locales que están diseñados para el uso dentro del área de la red local. Algunos servicios, ofrecidos a los usuarios locales, nunca deben cruzar la barrera entre la red local e internet. Algunos de estos servicios pueden provocar molestias a otros sitios de internet, otros pueden representar agujeros de seguridad si están disponibles fuera del área local, otros pueden ofrecer información que se desea que se mantenga en secreto, etc.

Algunos de los primeros servicios de red, especialmente los comandos remotos de BSD, fueron diseñados para compartir recursos de forma sencilla en el área local, en el entorno de confianza. Otros servicios posteriores se diseñaron para el acceso a internet, pero nacieron en un tiempo en el que internet era un ámbito al que solo accedía personal universitario e investigadores. Entonces se consideraba a internet un sitio seguro. Al tiempo que internet se iba convertía en una red global, incluyendo acceso general al público, se convirtió también en un entorno de completa desconfianza.

Muchos servicios de red Unix están diseñados para ofrecer información local sobre cuentas de usuario en el sistema, los programas que se están ejecutando y los recursos en uso, el estado del sistema, el estado de la red e información del estilo sobre otras máquinas conectadas a la red. No todos estos servicios que ofrecen información son agujeros de seguridad por sí mismos. No significa que alguien que los use obtenga directamente acceso no autorizado al sistema. Simplemente ofrecen información sobre el sistema y las cuentas de usuario que puede ser muy útil para alguien que busca debilidades en el sistema. También pueden proporcionar información tal como nombres de usuario, direcciones, teléfonos, etc.

Algunos de los servicios de red más peligrosos están diseñados para ofrecer acceso local a sistemas de ficheros y

dispositivos compartidos, tales como una impresora o un fax de red. Algunos servicios son difíciles de configurar correctamente, y algunos son difíciles de configurar para que sean seguros. Algunos servicios simplemente no tienen sentido en un hogar o en una oficina pequeña. Algunos sirven para gestionar grandes redes, ofrecer servicio de encaminamiento en internet, ofrecer servicio de acceso a bases de datos, soportar encriptado bidireccional y autenticación, etc.

Protección de los servicios locales inseguros.

La forma más sencilla de protegerse es no ofrecer el servicio, aunque es posible que sea necesario ofrecerlo. No todos los servicios pueden ser protegidos adecuadamente en el nivel de filtrado de paquetes. Por ejemplo, los servicios multimedia, ICQ y los servicios RPC son realmente complicados de proteger en este nivel.

Un método de asegurar un sistema es no alojar en la máquina firewall servicios de red que no se desee ofrecer fuera de la red local. Si el servicio no está disponible, no hay nada a lo que el cliente remoto se pueda conectar.

Un firewall de filtrado de paquetes no ofrece una seguridad completa, como ya hemos mencionado anteriormente. Algunos programas requieren medidas de seguridad de más alto nivel de las que se pueden ofrecer en el nivel de filtrado de paquetes. Algunos programas son demasiado problemáticos para correr el riesgo de ejecutarlos en una máquina firewall.

Los entornos pequeños, como los de un hogar, a menudo no disponen de las máquinas necesarias para asegurar el cumplimiento de las políticas de seguridad mediante la ejecución de servicios en otras máquinas. Se debe llegar a soluciones de compromiso. Sin embargo, en sitios locales pequeños con una LAN, los servicios para compartir archivos (por ejemplo, Samba) y ningún otro servicio privado local, no deben ejecutarse en la máquina firewall.

Los entornos mayores, simplemente no deben ejecutar los servicios en la máquina firewall. Esta máquina no debe tener cuentas de usuario y todo el software que no sea necesario debe ser desinstalado de ella. Dicha máquina no debe tener otra función que no sea la de ser una pasarela de seguridad.

Selección de los servicios a ofrecer

El primer paso en el proceso de asegurar un sistema es decidir qué servicios y demonios se pretenden ejecutar en la máquina firewall, así como detrás de ella, en la red que protege. Cada servicio tiene sus propias peculiaridades en cuanto a seguridad. Cuando se trata de elegir los servicios de red en sistemas Unix, la regla general es ofrecer solamente los servicios estrictamente necesarios y cuyo funcionamiento se conozca suficientemente. Es importante conocer un servicio de red, qué hace y a quién va dirigido antes de ofrecerlo, especialmente en una máquina conectada directamente a internet.

7.3. El Firewall Netfilter

Los sistemas Linux con núcleos posteriores al 2.4.x incorporan el firewall Netfilter en lugar del antiguo IPFW. La aplicación de administración del firewall Netfilter es iptables, mientras que para los antiguos IPFW las herramientas eran ipfwadm e ipchains. Las diferencias entre IPFW y Netfilter radican en cómo se encaminan o reenvían los paquetes a través del sistema operativo, con las consiguientes diferencias en cómo las reglas del firewall se construyen.

Las aplicaciones ipchains e iptables se parecen mucho desde fuera, pero en la práctica son muy distintas. Por lo tanto, hay que tener presentes estas diferencias, sobre todo en el caso de que se quiera migrar un firewall IPFW a Netfilter, ya que reglas parecidas pueden tener efectos muy distintos.

IPFW: recorrido de paquetes

En IPFW, administrado con ipfwadm o ipchains, se usaban tres cadenas de filtrado. Todos los paquetes que llegaban a una interfaz se filtraba en la cadena de entrada (input chain). Si el paquete se aceptaba, pasaba al módulo de encaminamiento. La función de encaminamiento determinaba si el paquete debía ser entregado localmente o si debía reenviarse a otra interfaz para la salida. El flujo de los paquetes se muestra en la figura 7-1.

Si el paquete se reenviaba, se filtraba una segunda vez con la cadena de reenvío (forward chain). Si el paquete se aceptaba,

pasaba a la cadena de salida (output chain). Tanto los paquetes que debían ser reenvíados como los paquetes generados localmente para la salida debían pasar por la cadena de salida. Si el paquete era aceptado, se enviaba hacia el exterior por la interfaz de red.

Los paquetes recibidos y los envíados al bucle local de retorno pasaban por dos filtros. Los paquetes reenvíados pasaban por tres filtros.

La ruta de retorno involucraba a dos cadenas. Como se muestra en la figura 6-1, cada paquete de retorno pasaba por el filtro de salida antes de "salir" a la interfaz de retorno. Entonces se aplicaba el filtro de entrada (ya que entra a la interfaz de retorno).

En el caso de los paquetes que debían ser desenmascarados antes de reenviarlos a la red LAN, los filtros de entrada se aplicaban. Más que pasar por la función de encaminamiento, el paquete se enviaba directamente a la cadena de filtrado de salida. De esta forma, los paquetes entrantes desenmascarados se filtraban dos veces. Los paquetes enmascarados para la salida se filtraban tres veces.

Netfilter: recorrido de paquetes

Con Netfilter, y su herramienta de administración iptables, se usan las cadenas de filtrado INPUT, OUTPUT y FORWARD incorporadas. Los paquetes entrantes pasan por la función de encaminamiento, que determina si el paquete se debe entregar a la cadena de entrada de la máquina local o a la cadena de reenvío. El flujo de paquetes con Netfilter se representa en la figura 7-2.

Si un paquete es aceptado por la cadena de entrada, el paquete se entrega localmente. Si un paquete con destino remoto es aceptado por la cadena de reenvío, el paquete se envía a la interfaz apropiada.

Los paquetes salientes generados por los procesos locales se pasan a la cadena de salida. Si el paquete es aceptado, se envía a la interfaz apropiada. De esta forma, cada paquete se filtra sólo una vez, excepto para los paquetes de retorno, que se filtran dos veces.

Características de iptables

La aplicación de administración de Netfilter, iptables, mantiene tablas de reglas separadas para cada clase de función de procesamiento realizada sobre paquetes. Estas tablas de reglas se implementan como módulos de funcionalidad separada. Los tres principales módulos son la tabla filter, la tabla nat y la tabla mangle. Cada uno de estos módulos tiene sus propias extensiones de módulo asignadas, las cuales se cargan automáticamente cuando se les hace referencia por primera vez.

La tabla filter es la tabla por defecto. Las otras tablas se especifican mediante una opción en la línea de comandos.

La tabla filter básica incluye:

- Operaciones relacionadas con cadenas sobre las tres cadenas incluidas (INPUT, OUTPUT y FORWARD) y sobre cadenas definidas por el usuario.

- Ayuda.

- Acciones a tomar (ACCEPT y DROP, aceptar y desechar).

- Operaciones de comparación de protocolo, dirección origen y destino, interfaz de entrada y de salida, manejo de la fragmentación.

- Operaciones de comparación para los campos de cabecera de TCP, UDP e ICMP.

La tabla filter, además, tiene dos tipos de módulos de extensión: extensión de acciones a tomar (target) y extensión de operaciones de comparación (match). La extensión target permite rechazar paquetes mediante la adicción de la acción REJECT, y la funcionalidad LOG. La extensión match, permite ampliar las operaciones de comparación con lo siguiente:

- El estado actual de la conexión.

- Listas de puertos.

- La dirección fuente MAC de las interfaces Ethernet.

- El identificador de usuario, grupo, proceso o grupo de

procesos que envía el paquete.

- El campo tipo de servicio de la cabecera IP (TOS).

- El campo de marca iptables.

- Comparación de paquetes con tasa limitada.

La tabla mangle tiene dos extensiones de acciones a tomar. El módulo MARK soporta la asignación de un valor al campo mark mantenido por iptables. El módulo TOS soporta la asignación del valor del campo TOS en el encabezado IP.

La tabla nat tiene extensiones de acciones a tomar para la traducción de direcciones origen y destino y para traducción de puertos. Los módulos que suporta son:

- SNAT (Source NAT), traducción de direcciones origen.

- DNAT (Destination NAT), traducción de direcciones destino.

- MASQUERADE, una forma de traducción de direcciones origen en la que se asigna a las conexiones direcciones IP temporales y cambiables asignadas dinámicamente.

- REDIRECT, una forma de traducción de direcciones destino en la que se redirige el paquete a la máquina local, sin tener en cuenta el contenido del campo de dirección de la cabecera IP.

La tabla filter

Las nuevas características de filtrado de iptables incluyen:

- Listas de puertos origen y destino.

- Acceso a los indicadores de estado TCP.

- Acceso al campo de opciones TCP.

- Mantenimiento del estado en conexiones TCP y en intercambios UDP e ICMP.

- Acceso al campo TOS de la cabecera IP.

- Acceso a la dirección MAC origen.

- Mayores capacidades de registro.

- Comparación de paquetes con limitación en tasa.

- Múltiples formas de rechazo paquetes.

- Colas de paquetes para programas de usuario.

- Filtrado de paquetes salientes mediante la identidad del usuario, del grupo o del grupo de procesos.

La tabla nat

En general hay tres formas de realizar la traducción de direcciones:

- Traducción de direcciones unidireccional hacia fuera. Usada por las redes que usan direcciones privadas. Dentro de la traducción de direcciones unidireccional hacia fuera existe un subtipo básico, caracterizado por el mapeo de todas las direcciones locales en una de entre un bloque de direcciones públicas. Existe otro tipo de traducción de direcciones denominado NAPT (Network Address Port Translation), en la que se mapean direcciones locales y privadas origen a una sola dirección pública. Ejemplo de este último tipo es el enmascaramiento de direcciones de Linux.

- Traducción de direcciones bidireccional. La traducción de direcciones bidireccional permite tanto conexiones entrantes como salientes. Un uso de esta forma de traducción es el mapeo de direcciones entre los espacios de direcciones de IPv4 e IPv6.

- Traducción doble. La traducción de direcciones origen y destino permite tanto conexiones entrantes como salientes. La traducción dos veces puede ser usada cuando se produce una colisión entre las direcciones de red del origen y el destino. Esta colisión podría ser el resultado de que un sitio por error esté usando una dirección pública ya asignada a otra máquina.

La traducción de direcciones de iptables soporta tanto SNAT y DNAT. La tabla de traducción permite modificar la dirección origen o la dirección o puerto destino. Esta tabla tiene tres cadenas ya incorporadas:

- PREROUTING: cadena que especifica los cambios de destino de los paquetes entrantes antes de pasarlos a la función de rutado (DNAT). Los cambios de la dirección destino pueden ser a la máquina local (funcionalidad proxy transparente) o a una máquina distinta (con lo que tendríamos funcionalidad de enmascaramiento).

- OUTPUT: cadena que especifica los cambios en el destino de los paquetes salientes generados localmente antes de que se realice la decisión de encaminamiento (DNAT, REDIRECT). Esto se hace habitualmente para redirigir los paquetes salientes de forma transparente hacia un proxy local, pero también puede ser usado para el reenvío a una máquina distinta.

- POSTROUTING: cadena para especificar cambios en el origen de los paquetes salientes rutados por el equipo (SNAT, MASQUERADE). Los cambios se aplican después de que se haya realizado la decisión de encaminamiento.

La tabla mangle

La tabla mangle permite asociar al paquete un valor que Netfilter mantiene, así como cambiar el valor del campo TOS del paquete para propósitos especiales antes de realizar la decisión de encaminamiento o reenvío. La tabla mangle tiene, desde el Kernel 2.4.18 las cinco siguientes cadenas ya incorporadas:

- PREROUTING: cadena que especifica los cambios en los paquetes entrantes conforme llegan a la interfaz, antes de que ninguna decisión de encaminamiento o reenvío se haya realizado.

- OUTPUT: cadena que especifica los cambios en los paquetes salientes generados localmente.

- INPUT: cadena que especifica los cambios en los paquetes entrantes a la máquina local.

- FORWARD: cadena que especifica las modificaciones en los paquetes reenviados a otras máquinas.

- POSTROUTING: cadena que especifica las modificaciones en los paquetes sobre los que ya se ha tomado la decisión de encaminamiento.

Con respecto al campo TOS, el router Linux local puede ser configurado para dejar los flags TOS asignados por la tabla mangle o como los han establecido las máquinas locales.

7.4. La sintaxis de iptables

Ya se mencionó anteriormente que iptables introduce el concepto de tablas de reglas separadas para diferentes tipos de funcionalidad de procesamiento de paquetes. Las tablas no predeterminadas se especifican mediante una opción de la línea de comandos. Tres tablas están disponibles: filter, nat y mangle.

La tabla filter es la tabla predeterminada. Contiene las verdaderas reglas de filtrado del firewall. Entre las cadenas incluidas están: INPUT, OUTPUT Y FORWARD.

La tabla nat contiene las reglas para la traducción de direcciones de direcciones y puertos fuente y destino. Estas reglas pertenecen a una función distinta de la de filtrado en el firewall. Las cadenas que incluye son: PREROUTING (DNAT / REDIRECT), OUTPUT (DNAT / REDIRECT) y POSTROUTING.

La tabla mangle contiene las reglas para el ajuste de los indicadores especializados para el encaminado de paquetes. Estos indicadores se inspeccionan más tarde mediante las reglas de la tabla filter. Las cadenas que incluye son: PREROUTING (paquetes encaminados), OUTPUT (paquetes generados localmente).

En cada tabla se podrán efectuar distintas acciones mediante los comandos en ella permitidos.

Comandos sobre la tabla filter

Los comandos que podemos ejecutar sobre la tabla filter están soportados por el módulo ip_tables. Para que puedan ejecutar es preciso que dicho módulo esté cargado, cosa que ocurre de forma automática al invocar por primera vez al comando iptables.

Operaciones sobre cadenas completas

Las acciones que implican a las cadenas de la tabla filter pueden ser las siguientes:

-N <cadena>: Crea una cadena de usuario.

-F [<cadena>]: Vacía la cadena especificada o todas las cadenas si no se especifica ninguna.

-X [<cadena>]: Borra la cadena especificada o todas si no se especifica ninguna.

-P <cadena><política>: Define la política predeterminada para una de las cadenas incluidas (INPUT, OUTPUT o FORWARD). La política puede ser ACCEPT o DROP.

-L [<cadena>]: Presenta una lista de las reglas presentes en la cadena especificada o en todas las cadenas si no se especifica ninguna.

Las opciones que podemos especificar para el comando -L son las siguientes:

-n: Muestra las direcciones IP y los números de puerto en formato numérico.

-v: Muestra la información con nivel de detalle alto. Esto incluirá información para cada regla como los contadores de paquete, sus opciones, interfaces de red, etc.

-x: Da los valores exactos de los contadores, en lugar de valores aproximados.

-line-numbers: Muestra la posición de las reglas

dentro de sus cadenas.

-Z: Resetea los contadores de paquetes y de bytes asociados a cada cadena.

[<comando>] -h: Muestra los comandos disponibles y sus opciones. Si especificamos un comando, se muestra la sintaxis y las opciones para dicho comando. Notese que realmente este comando no efectúa ninguna operación sobre la tabla filter ni sobre ninguna.

Operaciones sobre reglas individuales

Las operaciones más comunes son las de crear o borrar reglas en una cadena. Estas son:

-A <cadena>: Añade una regla al final de la cadena.

-I <cadena> [posición]: Inserta una regla en la posición dada al principio de la cadena especificada (si no se especifica la posición, ésta se toma como 1, y se introduciría al principio de la cadena.

-D <cadena> <número>: Borra la regla en la posición indicada por número en la cadena especificada.

Operaciones de comparación básicas

La operaciones de comparación que podemos realizar son las siguientes:

-i [!] [<interfaz>]: Para los paquetes entrantes a las cadenas INPUT o FORWARD, o sus subcadenas, especifica la interfaz de red a la que se aplicará la regla. Si no se especifica la interfaz, se aplicará a todas.

-o [!] [<interfaz>]: Para los paquetes salientes de las cadenas OUTPUT o FORWARD, o sus subcadenas, especifica la interfaz a la que se aplica la regla. Si no se especifica ninguna, la regla se aplica a todas las interfaces.

-p [!] [<protocolo>]: Especifica el protocolo al que se aplica la regla. Los protocolos incluidos son tcp, udp, icmp y all. El valor <protocolo> puede ser o bien el nombre o

bien su valor numérico (de los listados en /etc/protocols).

Para cada uno de los protocolos se pueden realizar más operaciones de comparación especificas del protocolo. La ayuda de iptables presenta una relación de todas las posibles operaciones específicas de cada protocolo con la orden: iptables -p <protocolo> -h

-s [!] <dirección><máscara>: Especifica la dirección origen de máquina o red a la que se aplica la regla.

-d [!] <dirección><máscara>: Especifica la dirección destino de máquina o red a la que se aplica la regla.

-j <acción>: Especifica qué hacer con el paquete si se cumple la regla. Los valores pueden ser ACCEPT, DROP o una cadena de usuario. Notese que especificar una acción es algo opcional, ya que si una regla no especifica acción, servirá para actualizar los contadores asociados a la cadena, y por tanto, permite la contabilidad TCP e IP.

Extensiones de acción para la tabla filter

Las extensiones de acción disponibles para la tabla filter son la función de registro y la capacidad de rechazar un paquete en lugar de descartarlo.

Para la acción REJECT, podemos añadir las siguientes opciones:

--log-level <syslog level>: Puede ser o bien un valor numérico o un valor simbólico, listado en /usr/include/sys/syslog.h, para la prioridad de registro. Los valores posibles son emerg (0), alert (1), crit (2), err (3), warn (4), notice (5), info (6) y debug (7).

--log-prefix <"cadena de descripción">: Permite imprimir un prefijo al principio del mensaje de registro para la regla correspondiente.

--log-ip-options: Incluye en el registro cualquier opción en el encabezado IP.

--log-tcp-sequence: Incluye el número de secuencia del paquete IP en el registro.

--log-tcp-options: Incluye cualquier opción del encabezado TCP en el registro.

Para la acción REJECT, las opciones que se pueden añadir son las siguientes:

--reject-with- <mensaje tipo 3 ICMP>: Especifica un mensaje ICMP de tipo 3 a devolver al origen, si debe ser distinto del predeterminado icmp-port-unreachable.

--reject-with tcp-reset: Los paquetes entrantes se rechazan con un paquete TCP RST, para cerrar la conexión.

--reject-with echo-reply: Puede usarse esta opción para responder a las peticiones de echo desde el firewall, sin enviar la petición a la máquina destino.

Extensiones de comparación para la tabla filter

Las extensiones de comparación de la tabla filter proporcionan acceso a los campos de las cabeceras TCP, UDP e ICMP, así como a las nuevas características incorporadas por iptables, como el mantenimiento del estado de la conexión, identidad del propietario del proceso que envía el paquete, listas de puertos, acceso a la dirección MAC origen y acceso al campo TOS (tipo de servicio) de la cabecera IP. Para cargar una extensión deberemos usar la opción:

-m <extensión>

Las extensiones soportadas son multiport, limit, state, owner, mark, tos y unclean. Una de las más interesantes es state, ya que permite el mantenimiento de información sobre el estado de las conexiones, y por tanto permite la implementación de cortafuegos dinámicos.

La extensión state

Los filtros de paquetes estáticos analizan el tráfico de paquetes de forma individual y separada para cada paquete. En este tipo de filtros no se tiene en cuenta el contexto en el que los indicadores de la cabecera TCP están activados o no, y los mensajes ICMP se tratan sin tener en cuenta las relaciones entre ellos o, los eventos ocurridos en la capa 3. La extensión state permite usar la información del estado en las comparaciones.

Con la extensión state, la información del estado se almacena cuando se comienza una conexión TCP o un intercambio UDP. Los paquetes siguientes se examinan no solo en base a la información de su cabecera, sino también en base al contexto del intercambio en curso. En definitiva, se incluye en el filtro información correspondiente a las capas superiores de transporte TCP o de aplicación UDP.

En términos de monitorización de las sesiones, las ventajas de mantener información del estado son menos obvias para TCP que para UDP, ya que TCP mantiene el estado de la conexión en sí mismo. Para UDP una ventaja directa es la capacidad de distinguir las respuestas a peticiones de otro tipo de datagramas. Por ejemplo, en el caso de una consulta DNS saliente, que implica un intercambio UDP nuevo, el concepto de sesión establecida en UDP permite dejar pasar las respuestas a través del firewall procedentes de la máquina y puerto correspondientes, con un tiempo de expiración determinado. Podemos entonces, no permitir los datagramas UDP entrantes que provengan de otras máquinas. En el caso del protocolo TCP, una de las ventajas es la de permitir los mensajes ICMP solo a las conexiones establecidas que se hayan permitido.

No obstante, a pesar de las ventajas que presentan los filtros dinámicos, la cantidad de recursos que precisan son mucho mayores, tanto de memoria como de tiempo de proceso. En cualquier caso, los filtros dinámicos deben estar provistos de un mecanismo capaz de seguir filtrando el tráfico de manera estática en el caso de que no existan recursos en un instante determinado para filtrar de forma dinámica.

Puede ser una opción recomendable la de implementar un filtro dinámico solo en los casos en los que sea una ventaja con respecto al ahorro de recursos. Por ejemplo, puede ser recomendable mantener el estado en conexiones a servicios FTP, ya que la cantidad de paquetes asociados a este tipo de

conexiones puede ser extremadamente grande, y se liberaría de tiempo de proceso al filtro estático si los paquetes se asociasen directamente a una conexión ya establecida y permitida.

Extensiones de acción para la tabla nat

Ya se mencionó antes que iptables soporta cuatro tipos de traducción de direcciones de red: SNAT, DNAT, MASQUERADE, y REDIRECT. Como parte de la tabla nat, cada una de estas acciones están disponibles cuando la regla pertenece a la tabla nat, especificado mediante la opción -t nat.

Extensión SNAT

La traducción de direcciones y puertos origen (NAPT) es el tipo de traducción de direcciones más común. Como se muestra en la figura 7-3, la traducción de dirección origen se realiza después de que la decisión de encaminamiento se haya tomado. La acción SNAT es solo valida por tanto en la cadena POSTROUTING. Como la acción SNAT se aplica inmediatamente antes de enviar el paquete, solo puede ser especificada una interfaz de salida.

La traducción de direcciones y puertos origen se usa cuando solo se dispone de una única dirección pública. El valor del origen puerto se cambia al de otro libre en la máquina firewall/NAT, dado que como la máquina realiza la traducción para un conjunto de máquinas locales, el puerto podría estar ya ocupado. Cuando llega la respuesta, el número de puerto es lo único que la máquina que realiza la traducción tiene para determinar el destino de ese paquete dentro de la red interna.

La sintaxis general para la traducción de direcciones origen con iptables es:

```
iptables    -t nat -A POSTROUTING ...\
            --out-interface <interfaz>  ...\
            -j SNAT --to-source <address> ...\
            [-<address>][:<port>-<port>
```

La dirección origen puede ser mapeada a un rango de posibles direcciones IP, si es que hay más de una. El puerto origen puede ser mapeado a un rango especifico de puertos

origen en el encaminador.

Extensión MASQUERADE

La traducción de direcciones origen se ha implementado en iptables de dos formas distintas: SNAT y MASQUERADE. La diferencia entre ambas es que esta última está indicada para el uso en conexiones con interfaces cuyas direcciones IP se asignen de forma dinámica, especialmente en aquellas conexiones temporales en las que la dirección IP pública será probablemente distinta en cada nueva conexión.

Como MASQUERADE es un caso particular de SNAT, solo puede ser usada esta acción en las reglas de la cadena POSTROUTING, y la regla solo puede referirse a una interfaz de salida. Al contrario que en el caso SNAT más general, MASQUERADE no toma como argumento la dirección fuente que imprimir al paquete. La dirección IP de la interfaz de salida es la que se usa, de forma automática.

La sintaxis general de MASQUERADE es:

```
iptables    -t nat -A POSTROUTING ...\
            - -out-interface <interfaz> ...\
            -j MASQUERADE [--to-ports <port>[-<port>]]
```

El puerto origen puede ser mapeado a un rango especifico de puertos origen en el router.

Extensión DNAT

La traducción de dirección y puerto origen es un tipo especial de traducción de direcciones. Se le puede sacar partido a este tipo de traducción en los casos en los que un sitio tiene una dirección IP pública asignada dinámicamente o una única dirección IP, y el administrador desea reenviar las conexiones entrantes a servidores internos que no son públicamente visibles. Si nos fijamos en la figura 6-3, la traducción de dirección y puerto destino se realiza antes de la decisión de encaminamiento. DNAT es válida en las cadenas PREROUTING y OUTPUT. En la cadena PREROUTING, DNAT puede ser una acción a tomar cuando se especifica la interfaz de entrada. En la cadena OUTPUT, DNAT puede ser una acción a tomar cuando se

especifica una interfaz de salida.

La sintaxis general de DNAT es:

```
iptables    -t nat -A PREROUTING ...\
            --in-interface <interfaz> ...\
            -j DNAT ...\
            --to-destination <dirección>[-<dirección>] ...\
            [:<puerto>[-<puerto>]]


iptables    -t nat -A OUTPUT ...\
            --out-interface <interfaz> ...\
            -j DNAT ...\
            --to-destination <dirección>[-<dirección>] ...\
            [:<puerto>[-<puerto>]]
```

La dirección destino puede ser mapeada en un rango de posibles direcciones IP, si están disponibles. El puerto destino puede ser mapeado en un rango específico de puertos alternativos en la máquina destino.

Extensión REDIRECT

La redirección de puertos es un caso especial de DNAT. En él, el paquete se redirige a un puerto en la máquina local. Los paquetes entrantes, que de otra forma serían reenviados, son redirigidos a la cadena de entrada de la interfaz de entrada. Los paquetes salientes generados por la máquina local son redirigidos a un puerto en la interfaz de retorno de la máquina local.

REDIRECT es simplemente una forma corta para el caso de redirección a la máquina local. No ofrece funcionalidad adicional. Podría usarse DNAT con el mismo efecto.

REDIRECT es una acción válida solo en las cadenas OUTPUT y PREROUTING. En la cadena PREROUTING, REDIRECT puede ser la acción cuando se especifica a la interfaz de entrada. En la cadena OUTPUT, la acción REDIRECT puede ser un objetivo cuando se especifica la interfaz de salida.

La sintaxis general de REDIRECT es:

```
iptables    -t nat -A PREROUTING ...\
```

```
                --in-interface <interfaz> ...\
                -j REDIRECT ...\
                [:<puerto>[-<puerto>]]

iptables        -t nat -A OUTPUT ...\
                --out-interface <interfaz> ...\
                -j REDIRECT ...\
                [:<puerto>[-<puerto>]]
```

Comandos sobre la tabla mangle

Las acciones y extensiones de la tabla mangle se aplican a las cadenas OUTPUT y PREROUTING. Para usar las características de la tabla mangle debe especificarse la directiva -t mangle en la invocación a iptables.

Extensiones de acción para la tabla mangle

Las extensiones disponibles para la tabla mangle son MARK, que especifica el valor de la marca de Netfilter mark(entero largo sin signo) para el paquete, y TOS, que escribe un valor en el campo TOS de la cabecera IP.

Un ejemplo de uso de MARK es:

```
iptables        -t mangle -A PREROUTING ...\
                -s <dirección> --sport 1024:65535 ...\
                -d <dirección> --dport 23 ...\
                -j MARK --set-mark 0x00010070
```

La extensión TOS aporta la función de especificación de los bits del campo TOS en el encabezado IP. Un ejemplo de uso puede ser:

```
iptables -t mangle -A OUTPUT ... -j TOS - -set-tos <tos>
```

Los valores tos posibles son los mismos que los disponibles en el módulo de extensión TOS de la tabla filter.

(1) Anónimo
Edición Especial Linux. Máxima Seguridad
PEARSON EDUCACIÓN, S.A. Madrid, 2000

(2) Robert L. Ziegler
Linux Firewalls Second Edition
New Riders

(3) Chris Hare, Haranjit Siyan
Internet Firewalls and Network Security Second Edition
Prentice Hall, New Riders

(4) Antonio Villalón Huerta
Seguridad en Unix y Redes
http://es.tldp.org

(5) Karanjit S. Siyan, Ph.D.
Tim Parker, Ph.D.
TCP/IP Unleashed Third Edition
SAMS Publishing

(6) Christofer Klaus
The ISS Sniffer FAQ
http://morehouse.org/secure/sniffaq.html

(7) FAQ de Linux de "The Linux Documentation Project"
Características de Linux
http://es.tldp.org/FAQ/FAQ_Linux/

(8) Recomendaciones CERT de seguridad
http://www.cert.org/advisories/

(9) CERT/CC Vulnerability Notes Database
http://www.kb.cert.org/vuls/

(10) Vulnerability Catalog
https://www.kb.cert.org/vulcatalog/

(11) Trends in Denial of Service Attack Technology
CERT Coordination Center in collaboration with Neil Long and Rob Thomas
http://www.cert.org/archive/pdf/DoS_trends.pdf

(12) Hunt Sniffer
http://lin.fsid.cvut.cz/~kra/hunt/

(13) Sniffit
http://reptile.rug.ac.be/~coder/sniffit/

(14) Netfilter
http://www.netfilter.org

Pliego

Prefacio

Este documento especifica el Protocolo Internet Estándar del DoD (Department of Defense, USA). Este documento está basado en seis ediciones anteriores de la Especificación del Protocolo Internet de ARPA, y el presente texto se basa en gran medida en ellas. Han habido muchos colaboradores en este trabajo en términos de conceptos y texto. Esta edición revisa aspectos de direccionamiento, tratamiento de errores, códigos de opción, y de las características de seguridad, prioridad, compartimientos y manejo de restricciones del protocolo Internet.

Jon Postel
Editor

1.Introducción

1.1. Motivación

El Protocolo Internet está diseñado para su uso en sistemas interconectados de redes de comunicación de ordenadores por intercambio de paquetes. A un sistema de este tipo se le conoce como "catenet" [1]. El protocolo internet proporciona los medios necesarios para la transmisión de bloques de datos llamados datagramas desde el origen al destino, donde origen y destino son hosts identificados por direcciones de longitud fija. El protocolo internet también se encarga, si es necesario, de la fragmentación y el reensamblaje de grandes datagramas para su transmisión a través de redes de trama pequeña.

1.2. Ambito

El Protocolo Internet está específicamente limitado a proporcionar las funciones necesarias para enviar un paquete de bits (un datagrama internet) desde un origen a un destino a través de un sistema de redes interconectadas. No existen mecanismos para aumentar la fiabilidad de datos entre los extremos, control de flujo, secuenciamiento u otros servicios que se encuentran normalmente en otros protocolos host-a- host. El protocolo internet puede aprovecharse de los servicios de sus redes de soporte para proporcionar varios tipos y calidades de servicio.

1.3.Interfaces

Este protocolo es utilizado por protocolos host-a-host en un entorno internet. Este protocolo utiliza a su vez protocolos de red locales para llevar el datagrama internet a la próxima pasarela ("gateway") o host de destino.

Por ejemplo, un módulo TCP llamaría al módulo internet para tomar un segmento TCP (incluyendo la cabecera TCP y los datos de usuario) como la parte de datos de un datagrama internet. El módulo TCP suministraría las direcciones y otros parámetros de la cabecera internet al módulo internet como

argumentos de la llamada. El módulo internet crearía entonces un datagrama internet y utilizaría la interfaz de la red local para transmitir el datagrama internet.

En el caso de ARPANET, por ejemplo, el módulo internet llamaría a un módulo de red local el cual añadiría el encabezado 1822 [2] al datagrama internet creando así un mensaje ARPANET a transmitir al IMP. La dirección ARPANET sería deducida de la dirección internet por la interfaz de la red local y sería la dirección de algún host en ARPANET, el cual podría ser una pasarela a otras redes.

1.4.Operación

El protocolo internet implementa dos funciones básicas: direccionamiento y fragmentación.

Los módulos internet usan las direcciones que se encuentran en la cabecera internet para transmitir los datagramas internet hacia sus destinos. La selección de un camino para la transmisión se llama encaminamiento.

Los módulos internet usan campos en la cabecera internet para fragmentar y reensamblar los datagramas internet cuando sea necesario para su transmisión a través de redes de "trama pequeña".

El modelo de operación es que un módulo internet reside en cada host involucrado en la comunicación internet y en cada pasarela que interconecta redes. Estos módulos comparten reglas comunes para interpretar los campos de dirección y para fragmentar y ensamblar datagramas internet. Además, estos módulos (especialmente en las pasarelas) tienen procedimientos para tomar decisiones de encaminamiento y otras funciones.

El protocolo internet trata cada datagrama internet como una entidad independiente no relacionada con ningún otro datagrama internet. No existen conexiones o circuitos lógicos (virtuales o de cualquier otro tipo).

El protocolo internet utiliza cuatro mecanismos clave para prestar su servicio: Tipo de Servicio, Tiempo de Vida, Opciones, y Suma de Control de Cabecera.

El Tipo de Servicio se utiliza para indicar la calidad del servicio deseado. El tipo de servicio es un conjunto abstracto o generalizado de parámetros que caracterizan las elecciones de servicio presentes en las redes que forman Internet. Esta indicación de tipo de servicio será usada por las pasarelas para seleccionar los parámetros de transmisión efectivos para una red en particular, la red que se utilizará para el siguiente salto, o la siguiente pasarela al encaminar un datagrama internet.

El Tiempo de Vida es una indicación de un límite superior en el periodo de vida de un datagrama internet. Es fijado por el remitente del datagrama y reducido en los puntos a lo largo de la ruta donde es procesado. Si el tiempo de vida se reduce a cero antes de que el datagrama llegue a su destino, el datagrama internet es destruido. Puede pensarse en el tiempo de vida como en un plazo de autodestrucción.

Las Opciones proporcionan funciones de control necesarias o útiles en algunas situaciones pero innecesarias para las comunicaciones más comunes. Las opciones incluyen recursos para marcas de tiempo, seguridad y encaminamiento especial.

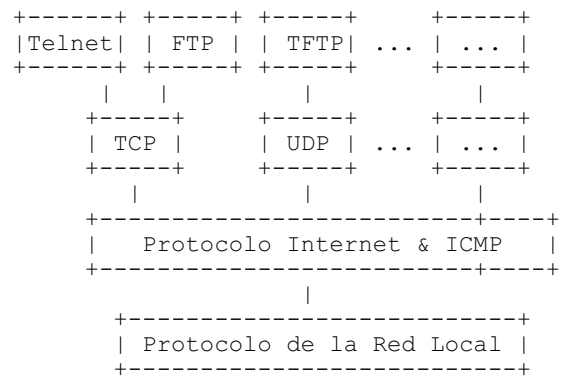
La Suma de Control de Cabecera proporciona una verificación de que la información utilizada al procesar el datagrama internet ha sido transmitida correctamente. Los datos pueden contener errores. Si la suma de control de cabecera falla, el datagrama internet es descartado inmediatamente por la entidad que detecta el error.

El protocolo internet no proporciona ningún mecanismo de comunicación fiable. No existen acuses de recibo ni entre extremos ni entre saltos. No hay control de errores para los datos, sólo una suma de control de cabecera. No hay retransmisiones. No existe control de flujo.
Los errores detectados pueden ser notificados por medio del Internet Control Message Protocol (ICMP) (Protocolo de Mensajes de Control de Internet) [3] el cual está implementado en el módulo del protocolo internet.

2. Panorama General

2.1. Relación con Otros Protocolos

El siguiente diagrama ilustra el lugar del protocolo internet en la jerarquía de protocolos:



Relación entre Protocolos

Figura 1.

El protocolo Internet interactúa por un lado con los protocolos host- a-host de alto nivel y por otro con el protocolo de la red local. En este contexto una "red local" puede ser una pequeña red en un edificio o una gran red como ARPANET.

2.2.Modelo de Operación

El modelo de operación para transmitir un datagrama de una aplicación a otra se ilustra en el siguiente escenario:

Suponemos que esta transmisión involucra a una pasarela intermedia.

La aplicación remitente prepara sus datos y llama a su módulo internet local para enviar esos datos como un datagrama y pasa la dirección de destino y otros parámetros como argumentos de la llamada.

El módulo internet prepara una cabecera de datagrama y adjunta los datos a él. El módulo internet determina una dirección de la red de área local para esta dirección internet, que en este caso es la dirección de una pasarela.

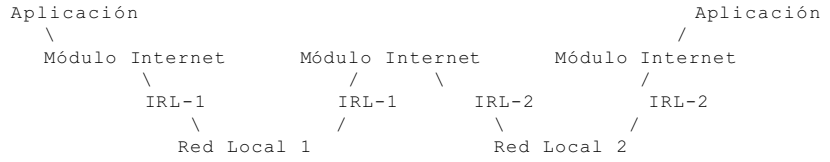
Envía este datagrama y la dirección de red local a la interfaz de red local.

La interfaz de red local crea una cabecera de red local, le adjunta el datagrama y entonces envía el resultado a través de la red local.

El datagrama llega a un host pasarela encapsulado en la cabecera de red local, la interfaz de red local desprende esta cabecera y dirige el datagrama hacia el módulo internet. El módulo internet determina a partir de la dirección internet que el datagrama debe ser reenviado a otro host en una segunda red. El módulo internet determina una dirección de red local para el host de destino. Llama a la interfaz de red local de esa red para enviar el datagrama.

Esta interfaz de red local crea una cabecera de red local y le adjunta el datagrama enviando el resultado al host de destino. En este host de destino la interfaz de red local le quita al datagrama la cabecera de red local y se lo pasa al módulo internet.

El módulo internet determina que el datagrama va dirigido a una aplicación en este host. Pasa los datos a la aplicación en respuesta a una llamada al sistema, pasando la dirección de origen y otros parámetros como resultado de la llamada.



Trayectoria de la transmisión

Figura 2

2.3. Descripción de Funciones

La función o propósito del Protocolo Internet es mover datagramas a través de un conjunto de redes interconectadas. Esto se consigue pasando los datagramas desde un módulo internet a otro hasta que se alcanza el destino. Los módulos internet residen en hosts y pasarelas en el sistema internet. Los datagramas son encaminados desde un módulo internet a otro

a través de redes individuales basándose en la interpretación de una dirección internet. Por eso, un importante mecanismo del protocolo internet es la dirección internet.

En el enrutamiento de mensajes desde un módulo internet a otro, los datagramas pueden necesitar atravesar una red cuyo tamaño máximo de paquete es menor que el tamaño del datagrama. Para salvar esta dificultad se proporciona un mecanismo de fragmentación en el protocolo internet.

Direccionamiento

Se establece una distinción entre nombres, direcciones y rutas [4]. Un nombre indica qué buscamos. Una dirección indica dónde está. Una ruta indica cómo llegar allí. El protocolo internet maneja principalmente direcciones. Es tarea de los protocolos de mayor nivel (es decir, protocolos host-a-host o entre aplicaciones) hacer corresponder nombres con direcciones. El módulo internet hace corresponder direcciones de internet con direcciones de red local. Es tarea de los procedimientos de menor nivel (es decir, redes locales o pasarelas) realizar la correspondencia entre direcciones de red local y rutas.

Las direcciones son de una longitud fija de 4 octetos (32 bits). Una dirección comienza por un número de red, seguido de la dirección local (llamada el campo "resto"). Hay 3 formatos o clases de direcciones internet: En la Clase A, el bit más significativo es 0, los 7 bits siguientes son la red, y los 24 bits restantes son la dirección local; en la Clase B, los dos bits más significativos son uno-cero ("10"), los 14 bits siguientes son la red y los últimos 16 bits son la dirección local; en la Clase C, los tres bits más significativos son uno-uno-cero ("110"), los 21 bits siguientes son la red y los 8 restantes son la dirección local.

Se debe tener cuidado al relacionar direcciones internet con direcciones de red local; un host individual físicamente hablando debe ser capaz de actuar como si fuera varios hosts distintos, hasta el punto de usar varias direcciones internet distintas. Algunos hosts tendrán también varios interfaces físicos (multi-homing).

Esto quiere decir que se debe establecer algún mecanismo que permita a un host tener varios interfaces físicos de red, cada uno de ellos con varias direcciones lógicas internet.

Se pueden encontrar ejemplos de correspondencias de direcciones en "Correspondencias de Direcciones" [5].

Fragmentación

La fragmentación de un datagrama internet es necesaria cuando éste se origina en una red local que permite un tamaño de paquete grande y debe atravesar una red local que limita los paquetes a un tamaño inferior para llegar a su destino.

Un datagrama internet puede ser marcado como "no fragmentar". Todo datagrama internet así marcado no será fragmentado entre distintas redes bajo ninguna circunstancia. Si un datagrama internet marcado como "no fragmentar" no puede ser entregado en su destino sin fragmentarlo, entonces debe ser descartado.

La fragmentación, transmisión y reensamblaje a través de una red local invisible para el módulo del protocolo internet se llama fragmentación intranet y puede ser utilizada [6].

El procedimiento de fragmentación y reensamblaje en internet tiene que ser capaz de dividir un datagrama en un número casi arbitrario de piezas que puedan ser luego reensambladas. El receptor de los fragmentos utiliza el campo de identificación para asegurarse de que no se mezclan fragmentos de distintos datagramas. El campo posición ("offset") le indica al receptor la posición de un fragmento en el datagrama original. La posición y longitud del fragmento determinan la porción de datagrama original comprendida en este fragmento. El indicador "más-fragmentos" indica (puesto a cero) el último fragmento. Estos campos proporcionan información suficiente para reensamblar datagramas.

El campo identificador se usa para distinguir los fragmentos de un datagrama de los de otro. El módulo de protocolo de origen de un datagrama internet establece el campo identificador a un valor que debe ser único para ese protocolo y par origen-destino durante el tiempo que el datagrama estará activo en el sistema internet. El módulo de protocolo de origen de un datagrama completo pone el indicador "más-fragmentos" a cero y la posición del fragmento a cero.

Para fragmentar un datagrama internet grande, un módulo de protocolo internet (p. ej. , en una pasarela) crea dos nuevos

datagramas internet y copia el contenido de los campos de cabecera internet del datagrama grande en las dos cabeceras nuevas. Los datos del datagrama grande son divididos en dos trozos tomando una resolución mínima de 8 octetos (64 bits) (el segundo trozo puede no ser un múltiplo entero de 8 octetos, pero el primero sí debe serlo). Llamemos al número de bloques de 8 octetos en el primer trozo NFB (Number of Fragment Blocks: Número de Bloques del Fragmento). El primer trozo de datos es colocado en el primer nuevo datagrama internet y el campo longitud total se establece a la longitud del primer datagrama. El indicador "más fragmentos" es puesto a uno. El segundo trozo de datos es colocado en el segundo nuevo datagrama internet y el campo longitud total se establece a la longitud del segundo datagrama. El indicador "más fragmentos" lleva el mismo valor que en el datagrama grande. El campo posición del segundo nuevo datagrama se establece al valor de ese campo en el datagrama grande más NFB.

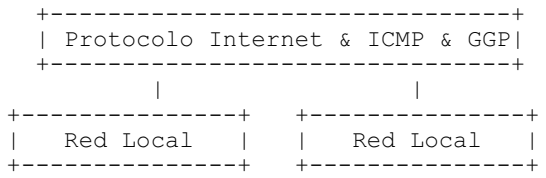
Este procedimiento puede generalizarse para una n-partición, mejor que para la división en dos partes descrita.

Para ensamblar los fragmentos de un datagrama internet, un módulo de protocolo internet (por ejemplo en un host de destino) combina todos los datagramas internet que tengan el mismo valor en los cuatro campos: identificación,origen, destino y protocolo. La combinación se realiza colocando el trozo de datos de cada fragmento en su posición relativa indicada por la posición del fragmento en la cabecera internet de ese fragmento. El primer fragmento tendrá posición cero, y el último fragmento tendrá el indicador "más fragmentos" puesto a cero.

2.4.Pasarelas

Las pasarelas implementan el protocolo internet para reexpedir datagramas entre redes. Las pasarelas también implementan el Protocolo Pasarela a Pasarela (Gateway to Gateway Protocol, GGP) [7] para coordinar el encaminamiento y otra información de control internet.

En una pasarela los protocolos de nivel superior no necesitan ser implementados y las funciones GGP son añadidas al módulo IP.



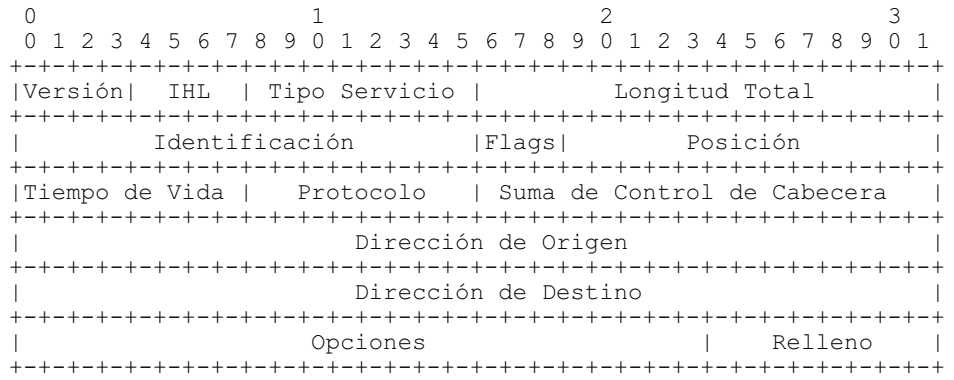
Protocolos de Pasarela

Figura 3.

3. Especificación

3.1.Formato de la Cabecera Internet

A continuación vemos un resumen del contenido de la cabecera internet.



Ejemplo de Cabecera de un Datagrama Internet
Figura 4

Nótese que cada marca (-) corresponde a una posición de un bit.

Versión:4 bits

El campo Versión describe el formato de la cabecera internet. Este documento describe la versión 4.

IHL:4 bits

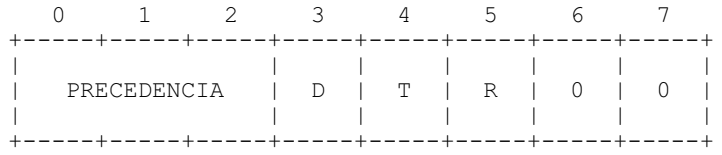
Longitud de la Cabecera Internet (Internet Header Length), es la longitud de la cabecera en palabras de 32 bits, y por tanto apunta al comienzo de los datos. Nótese que el valor mínimo para una cabecera correcta es 5.

Tipo de Servicio:8 bits

El Tipo de Servicio proporciona una indicación de los parámetros abstractos de la calidad de servicio deseada. Estos parámetros se usarán para guiar la selección de los parámetros de servicio reales al transmitir un datagrama a través de una red en particular. Algunas redes ofrecen prioridad de servicio, la cual trata de algún modo el tráfico de alta prioridad como más importante que el resto del tráfico (generalmente aceptando sólo tráfico por encima de cierta prioridad en momentos de sobrecarga). La elección más común es un compromiso a tres niveles entre baja demora, alta fiabilidad, y alto rendimiento.

Bits 0-2: Prioridad.

Bit3:0 = Demora Normal, 1 = Baja Demora. Bit4:0 = Rendimiento Normal , 1 = Alto rendimiento. Bit 5:0 = Fiabilidad Normal, 1 = Alta fiabilidad. Bits 6-7:Reservado para uso futuro.



Precedencia

111 - Control de Red
110 - Control Entre Redes
101 - CRITICO/ECP
100 - Muy urgente (Flash Override)
011 - Urgente (Flash)
010 - Inmediato
001 - Prioridad
000 - Rutina

El uso de las indicaciones de Retraso, Rendimiento y fiabilidad puede incrementar el coste (en cierto sentido) del servicio. En muchas redes un mejor rendimiento para uno de estos parámetros conlleva un peor rendimiento en algún otro. Excepto para casos excepcionales no se deben establecer más de dos de estos tres indicadores.

El tipo de servicio se usa para especificar el tratamiento del

datagrama durante su transmisión a través del sistema internet. Se dan ejemplos de relación entre el tipo de servicio internet y el servicio real proporcionado por redes como AUTODIN II, ARPANET, SATNET y PRNET en "Correspondencias de Servicios" [8]

La denominación de precedencia 'Control de Red' está pensada para ser usada dentro de una sola red. El uso y control efectivos de este modo es responsabilidad de cada red. El modo 'Control Entre Redes' está pensado para su uso exclusivo por parte de generadores de control de pasarela. Si el uso efectivo de estos modos de precedencia concierne a una red en particular, es responsabilidad de esa red controlar el acceso a, y el uso de, esos modos de precedencia.

Longitud Total: 16 bits

La Longitud Total es la longitud del datagrama, medida en octetos, incluyendo la cabecera y los datos. Este campo permite que la longitud máxima de un datagrama sea de 65,535 octetos. Los datagramas de tal longitud no son prácticos para la mayoría de hosts y redes. Todos los hosts deben estar preparados para aceptar datagramas de hasta 576 octetos (tanto si llegan completos como en fragmentos). Se recomienda que los hosts envíen datagramas mayores de 576 octetos sólo si tienen la seguridad de que el destinatario está preparado para aceptarlos.

El número 576 se ha seleccionado para permitir que un bloque de datos de tamaño razonable sea transmitido junto a la información de cabecera necesaria. Por ejemplo, este tamaño permite que un bloque de datos de 512 octetos más 64 octetos de cabecera quepa en un datagrama. La cabecera internet de tamaño máximo son 60 octetos, y una cabecera internet típica son 20 octetos, admitiendo así un margen para cabeceras de protocolos de nivel superior.

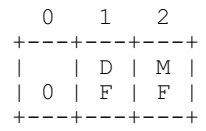
Identificación:16 bits

Es un valor de identificación asignado por el remitente como ayuda en el ensamblaje de fragmentos de un datagrama.

Flags (indicadores): 3 bits

Son diversos indicadores de control.

Bit 0: reservado, debe ser cero.
Bit 1: (DF) No Fragmentar (Don't Fragment) 0 = puede fragmentarse, 1 = No Fragmentar.
Bit 2: (MF) Más Fragmentos (More Fragments) 0 = Último Fragmento, 1 = Más Fragmentos.



Posición del Fragmento: 13 bits

Este campo indica a que parte del datagrama pertenece este fragmento.

La posición del fragmento se mide en unidades de 8 octetos (64 bits). El primer fragmento tiene posición 0.

Tiempo de Vida: 8 bits

Este campo indica el tiempo máximo que el datagrama tiene permitido permanecer en el sistema internet. Si este campo contiene el valor cero, entonces el datagrama debe ser destruido. Este campo es modificado durante el procesamiento de la cabecera internet. El tiempo es medido en segundos, pero como todo módulo que procese un datagrama debe decrementar el TTL (Time To Live: Tiempo de Vida) al menos en uno, incluso si procesa el datagrama en menos de un segundo, se debe pensar en el TTL sólo como un límite superior del tiempo durante el cual un datagrama puede existir. La intención es hacer que los datagramas imposibles de entregar sean descartados, y limitar el máximo periodo de vida de un datagrama.

Protocolo:8 bits

Este campo indica el protocolo del siguiente nivel usado en la parte de datos del datagrama internet. Los valores de varios protocolos son especificados en "Números Asignados" [9].

Suma de Control de Cabecera: 16 bits

Suma de Control de la cabecera solamente. Dado que algunos campos de la cabecera cambian (p. ej. el tiempo de vida), esta suma es recalculada y verificada en cada punto donde la cabecera internet es procesada.

El algoritmo de la suma de control es:

> El campo suma de control es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera. A la hora de calcular la suma de control, el valor inicial de este campo es cero.

Esta es una suma de control fácil de calcular y la evidencia experimental indica que es adecuada, pero es provisional y puede ser reemplazada por un procedimiento CRC, dependiendo de la experiencia ulterior.

Dirección de Origen: 32 bits

La dirección de origen. Ver sección 3.2.

Dirección de Destino: 32 bits

La dirección de destino. Ver sección 3.2.

Opciones: variable

Las opciones pueden o no aparecer en los datagramas. Deben ser implementadas por todos los módulos IP (host y pasarelas). Lo que es opcional es su transmisión en cualquier datagrama en particular, no su implementación.

En algunos entornos la opción de seguridad puede ser requerida en todos los datagramas.

El campo opción es de longitud variable. Pueden existir cero o más opciones. Existen dos casos para el formato de una opción:

Caso 1: Un sólo octeto de tipo-opción.

Caso 2: Un octeto tipo-opción, un octeto longitud-opción y los octetos correspondientes a los datos de opción. El octeto longitud-opción es la cuenta del octeto tipo-opción y el octeto longitud-opción así como los octetos de datos de opción.

El octeto tipo-opción tiene 3 campos

1 bit indicador de copiado,
2 bits clase de opción,
5 bits número de opción.

El indicador de copiado indica si esta opción es copiada en todos los fragmentos al fragmentar.

0 = no copiada
1 = copiada

Las clases de opción son:

0 = control
1 = reservado para uso futuro
2 = depuración y medida
3 = reservado para uso futuro

Se definen las siguientes opciones de internet:

CLASE	NUMERA	LONGITUD	DESCRIPCIÓN
0	0	-	Fin de la lista de opciones. Esta opción ocupa un sólo octeto. No tiene octeto de longitud.
0	1	-	Sin Operación. Esta opción ocupa un sólo octeto. No tiene octeto de longitud.
0	2	11	Seguridad. Usado para Seguridad, Compartimentación, Grupo de Usuario (TCC), y Códigos de Manejo de Restricciones compatibles con los requerimientos del Departamento de Defensa.
0	3	var.	Encaminamiento de Origen No Estricto (Loose Source Routing). Usado para encaminar el datagrama internet en base a la información suministrada por el origen.
0	9	var.	Encaminamiento de Origen Fijo (Strict Source Routing). Usado para encaminar el datagrama internet en base a la información suministrada por el origen.
0	7	var.	Registrar Ruta (Record Route). Usado para rastrear la ruta que sigue un datagrama internet.
0	8	4	Identificador de Flujo (Stream ID). Usado para transportar el identificador de flujo.
2	4	var.	Marca de Tiempo Internet (Internet Timestamp)

Definiciones de Opciones Específicas

Fin De La Lista de Opciones

Esta opción indica el final de la lista de opciones. Esta puede no coincidir con el final de la cabecera internet según expresa la longitud de cabecera internet. Se usa al final de todas las opciones, no al final de cada opción, y sólo es necesario usarla si, caso de no hacerlo, el final de las opciones no coincidiera con el final de la cabecera internet.

Puede ser copiado, introducido o borrado en la fragmentación, o por cualquier otra razón.

Sin Operación

Esta opción puede usarse entre opciones para, por

ejemplo, ajustar el comienzo de una opción siguiente a una posición múltiplo de 32 bits.

Puede ser copiado, introducido o borrado en la fragmentación, o por cualquier otra razón.

Seguridad

Esta opción proporciona a los hosts una forma de enviar parámetros de seguridad, compartimentación, manejo de restricciones y TCC (grupo de usuarios cerrado). El formato de esta opción es el siguiente:

```
+--------+--------+---//---+---//---+---//---+---//---+
|10000010|00001011|SSS  SSS|CCC  CCC|HHH  HHH|  TCC   |
+--------+--------+---//---+---//---+---//---+---//---+
Tipo=130 Longitud=11
```

Seguridad (campo S):16 bits

Especifica uno de entre 16 niveles de seguridad (ocho de los cuales están reservados para uso futuro)

```
00000000 00000000 - No Clasificado
11110001 00110101 - Confidencial
01111000 10011010 - EFTO
10111100 01001101 - MMMM
01011110 00100110 - PROG
10101111 00010011 - Restringido
11010111 10001000 - Secreto
01101011 11000101 - Alto Secreto
00110101 11100010 - (Reservado para uso futuro)
10011010 11110001 - (Reservado para uso futuro)
01001101 01111000 - (Reservado para uso futuro)
00100100 10111101 - (Reservado para uso futuro)
00010011 01011110 - (Reservado para uso futuro)
10001001 10101111 - (Reservado para uso futuro)
11000100 11010110 - (Reservado para uso futuro)
11100010 01101011 - (Reservado para uso futuro)
```

Compartimentos (Campo C):16 bits

Cuando la información no está compartimentada se usa un valor de todo ceros. Otros valores para el campo Compartimentos pueden obtenerse de la Agencia de Inteligencia de Defensa.

Manejo de Restricciones (campo H): 16 bits

Los valores para los marcadores de control y liberación son

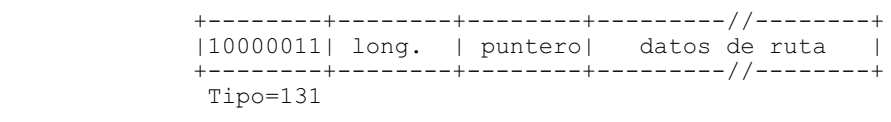
digrafos alfanuméricos y están definidos en el Manual DIAM 65-19 "Marcadores de Seguridad Estándar", de la Agencia de Inteligencia de Defensa.

Código de Control de Transmisión (Campo TCC): 24 bits

Proporciona un mecanismo para segregar el tráfico y definir comunidades de interés controladas entre diversos suscriptores. Los valores TCC son trigrafos, se pueden encontrar en "HQ DCA Code 530".

Debe copiarse al fragmentar. Esta opción aparece como máximo una vez en un datagrama.

Ruta de Origen No Estricta y Registrar Ruta



La opción de Encaminamiento de Origen No Estricto y Registrar Ruta ("loose source and record route (LSRR)") proporciona un mecanismo mediante el cual el origen de un datagrama internet puede suministrar información de encaminamiento que será usada por las pasarelas para transmitir el datagrama a su destino, y para almacenar la información de la ruta.

La opción comienza con el código de tipo de opción. El segundo octeto es la longitud de la opción que incluye al código de tipo de opción, al propio octeto de longitud, al octeto con el puntero y a los (long. - 3) octetos de datos de la ruta. El tercer octeto es el puntero a los datos de ruta que indica el octeto en el cual comienza la siguiente dirección de origen a procesar. El puntero es relativo a esta opción, y el valor legal más pequeño para el puntero es 4.

Los datos de ruta se componen de una serie de direcciones internet. Cada dirección internet son 32 bits o 4 octetos. Si el puntero es mayor que la longitud, la ruta de origen está vacía (y la ruta registrada llena) y el encaminamiento debe basarse en el campo de la dirección de destino.

Si se ha alcanzado la dirección indicada en el campo dirección de destino y el puntero no es mayor que la longitud, la

siguiente dirección en la ruta de origen sustituye a la dirección del campo de dirección de destino, y la dirección de ruta registrada sustituye a la dirección de origen recién usada, y se suma cuatro al puntero.

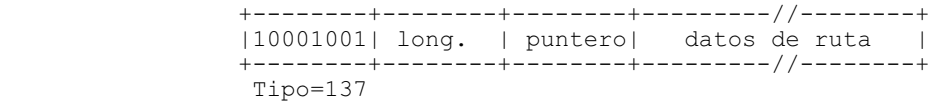
La dirección de ruta registrada es la propia dirección internet que tiene el módulo internet en el entorno en el cual el datagrama está siendo reenviado.

Este procedimiento de sustituir la ruta de origen con la ruta registrada (si bien está en orden inverso del necesario para ser usada como ruta de origen) significa que la opción (y la cabecera IP en su totalidad) sigue teniendo una longitud constante a medida que el datagrama progresa a través de internet.

Esta opción es una ruta de origen no estricta porque la pasarela o IP del host puede utilizar cualquier ruta con cualquier número de pasarelas intermedias para alcanzar la siguiente dirección en la ruta.

Debe copiarse al fragmentar. Aparece como máximo una vez en un datagrama.

Ruta de Origen Estricta y Registrar Ruta



La opción de Ruta de Origen Estricta y Registrar Ruta (strict source and record route (SSRR)) proporciona al origen de un datagrama internet un medio de suministrar información de encaminamiento a ser usada por las pasarelas al reenviar el datagrama al destino y para registrar la información de la ruta. La opción comienza con el código de tipo de opción. El segundo octeto es la longitud de la opción que incluye al código de tipo de opción, al propio octeto de longitud, al octeto con el puntero y a los (long. - 3) octetos de datos de la ruta. El tercer octeto es el puntero a los datos de ruta que indica el octeto en el cual comienza la siguiente dirección de origen a procesar. El puntero es relativo a esta opción, y su menor valor legal es 4.

Los datos de ruta se componen de una serie de direcciones internet. Cada dirección internet son 32 bits o 4 octetos. Si el puntero es mayor que la longitud, la ruta de origen está vacía (y

la ruta registrada llena) y el encaminamiento debe basarse en el campo de la dirección de destino.

Si se ha alcanzado la dirección indicada en el campo de dirección de destino y el puntero no es mayor que la longitud, la siguiente dirección en la ruta de origen sustituye a la dirección del campo de dirección de destino, y la dirección de ruta registrada sustituye a la dirección de origen recién usada, y se suma cuatro al puntero.

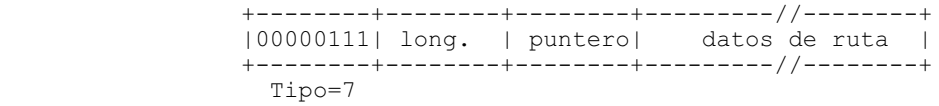
La dirección de ruta registrada es la propia dirección internet del módulo internet tal y como es en el entorno en el cual el datagrama está siendo reexpedido.

Este procedimiento de sustituir la ruta de origen con la ruta registrada (si bien está en orden inverso del necesario para ser usada como ruta de origen) significa que la opción (y la cabecera IP en su totalidad) sigue teniendo una longitud constante a medida que el datagrama progresa a través de internet.

Esta opción es una ruta de origen estricta porque la pasarela o el IP del host deben enviar el datagrama directamente a la siguiente dirección en la ruta de origen sólamente a través de la red conectada directamente indicada en la siguiente dirección, para alcanzar la siguiente pasarela o host especificado en la ruta.

Debe copiarse al fragmentar. Aparece como máximo una vez en un datagrama.

Registrar Ruta



La opción de Registrar Ruta proporciona un mecanismo para registrar la ruta de un datagrama internet.

La opción comienza con el código de tipo de opción. El segundo octeto es la longitud de la opción que incluye al código de tipo de opción, al propio octeto de longitud, al octeto con el puntero y a los (long. - 3) octetos de datos de la ruta. El tercer octeto es el puntero a los datos de ruta que indica el octeto en

el cual comienza la siguiente área para almacenar una dirección de ruta. El puntero es relativo a esta opción, y su menor valor legal es 4.

Una ruta registrada se compone de una serie de direcciones internet. Cada dirección internet son 32 bits o 4 octetos. Si el puntero es mayor que la longitud, el área de datos de la ruta registrada esta llena. El host de origen debe construir esta opción con una área de datos de ruta con suficiente espacio para contener todas las direcciones esperadas. El tamaño de la opción no cambia al agregar direcciones. El contenido inicial del área de datos de ruta debe ser cero.

Cuando un módulo internet encamina un datagrama, comprueba si la opción Registrar ruta está presente. Si lo está, inserta su propia dirección internet, tal y como se la conoce en el entorno en el cual el datagrama está siendo transmitido, en la ruta registrada, comenzando en el octeto indicado por el puntero, e incrementa el puntero en cuatro.

Si el área de datos de ruta ya está llena (el puntero sobrepasa a longitud) el datagrama es transmitido sin insertar la dirección en la ruta registrada. Si hay algo de espacio pero no el suficiente para insertar una dirección completa, el datagrama original es considerado erróneo y descartado. En ambos casos se puede enviar un mensaje de problema de parámetros ICMP al host de origen [3].

No se copia al fragmentar, va sólo en el primer fragmento. Aparece como máximo una vez en un datagrama.

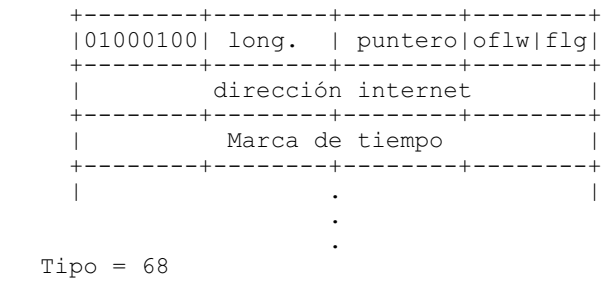
Identificador de Flujo

```
+--------+--------+--------+--------+
|10001000|00000010|   ID de Flujo   |
+--------+--------+--------+--------+
 Tipo=136 Longitud=4
```

Esta opción proporciona una forma de transportar el identificador de flujo SATNET de 16 bits a través de redes que no soportan el concepto de flujo.

Debe copiarse al fragmentar. Aparece como máximo una vez en un datagrama.

Marca de tiempo Internet



La Longitud de opción es el número de octetos en la opción contando los octetos de tipo, longitud, puntero y desbordamiento/ indicadores (oflw/flg, overflow/flags) (máxima longitud: 40)

El puntero es el número de octetos desde el principio de esta opción hasta el final de las marcas de tiempo mas uno (es decir, apunta al octeto de comienzo del espacio para la siguiente marca de tiempo). El valor legal mínimo es 5. El área de marca de tiempo está llena cuando el puntero es mayor que la longitud. El Desbordamiento (oflw) (4 bits) es el número de módulos IP que no han podido registrar marcas de tiempo debido a falta de espacio.

Los valores de los Indicadores (4 bits) son

0 Sólo marcas de tiempo, almacenadas en palabras de 32 bits consecutivas,

1 cada marca de tiempo es precedida con la dirección internet de la entidad registradora,

3 Los campos de dirección internet están preespecificados. Un módulo IP registra su marca de tiempo sólo si sudirección concuerda con la siguiente dirección internet especificada.

La Marca de Tiempo es una marca temporal de 32 bits, justificada a la derecha, en milisegundos desde la medianoche UT. Si el tiempo no está disponible en milisegundos o no puede darse con respecto a la medianoche UT, entonces se puede insertar cualquier tiempo como marca de tiempo, siempre que el bit de mayor orden sea puesto a uno para indicar el uso de un valor no estándar.

El host de origen debe construir está opción con un área de

datos para la marca de tiempo lo suficientemente grande para que quepa toda la información de marcas temporales esperada. El tamaño de la opción no cambia al añadir marcas de tiempo. El contenido inicial del área de datos de marcas de tiempo debe ser cero o pares dirección internet/cero.

Si el área de datos de marca de tiempo ya está llena (el puntero sobrepasa a la longitud) el datagrama es transmitido sin insertar marca de tiempo, pero el contador de desbordamiento es incrementado en uno.

Si hay algo de espacio pero no el suficiente para insertar una marca de tiempo completa, o el contador de desbordamiento el mismo se desborda, el datagrama original es considerado erróneo y descartado. En ambos casos se puede enviar un mensaje de problema de parámetros ICMP al host de origen [3]. La opción de marca de tiempo no se copia al fragmentar. Es transportada sólo en el primer fragmento. Aparece como máximo una vez en un datagrama.

Valor de Relleno: variable

El Valor de Relleno se usa para asegurar que la cabecera internet ocupa un múltiplo de 32 bits. El valor de relleno es cero.

3.2.Discusión

La implementación de un protocolo debe ser robusta. Cada implementación debe estar preparada para interoperar con otras creadas por diferentes individuos. Aunque el objeto de esta especificación es ser explícita acerca del protocolo, existe la posibilidad de que se produzcan interpretaciones discrepantes. En general, una implementación debe ser conservadora en su comportamiento al enviar, y tolerante al recibir. Es decir, debe tener cuidado de enviar datagramas bien formados, pero debe aceptar cualquier datagrama que pueda interpretar (p. ej. no poner pegas a errores técnicos cuando a pesar de ello el significado sea claro).

El servicio básico internet está orientado a datagramas y posibilita la fragmentación de datagramas en las pasarelas, teniendo lugar el reensamblaje en el módulo internet de destino en el host de destino. Naturalmente, la fragmentación y el

reensamblaje de datagramas en una red o mediante acuerdo privado entre las pasarelas de una red está permitido también, dado que esto es transparente a los protocolos internet y a protocolos de nivel superior. Este tipo de fragmentación y reensamblaje transparente se denomina fragmentación "dependiente de la red" (o intranet) y no se tratará más sobre ello aquí.

En las direcciones internet se distingue entre orígenes (fuentes) y destinos a nivel de host y se proporciona también un campo en el protocolo para ellas. Se ha asumido que cada protocolo proporcionará los medios para cualquier multiplexado que sea necesario en un host.

Direccionamiento

Para proporcionar flexibilidad en la asignación de direcciones a redes y tener en cuenta un gran número de redes de pequeño a medio tamaño, la interpretación del campo dirección está codificada para especificar un pequeño número de redes con un gran número de hosts, un número moderado de redes con un número moderado de hosts, y un gran número de redes con un pequeño número de hosts. Además existe un código de escape para un modo de direccionamiento extendido.

Formatos de dirección:

Bits de mayor orden	Formato	Class
0	7 bits de red, 24 bits de host	a
10	14 bits de red, 16 bits de host	b
110	21 bits de red, 8 bits de host	c
111	cód. escape para modo extendido	

Un valor cero en el campo de red indica la red actual. Sólo se usa en ciertos mensajes ICMP. El modo de direccionamiento extendido no está definido. Ambas características están reservadas para uso futuro.

Los valores efectivos asignados para direcciones de red se dan en "Números asignados" [9].

La dirección local, asignada por la red local, debe tener en cuenta que un sólo host puede actuar como varios hosts internet distintos. Es decir, debe existir una correspondencia, entre direcciones de host internet e interfaces de red/host que permitan que a una interface le correspondan varias direcciones

internet. Se debe tener en cuenta que un host puede tener varios interfaces físicos y deba tratar los datagramas de varios de ellos como si fueran destinados al mismo host.

La correspondencia entre direcciones internet y direcciones para ARPANET, SATNET, PRNET y otras redes se describen en "Correspondencias entre direcciones" [5].

Fragmentación y Reensamblaje.

El campo de identificación internet (ID) se usa junto con las direcciones de origen y destino, y los campos de protocolo, para identificar fragmentos de datagrama para su reensamblaje.

El bit indicador Más Fragmentos ("More Fragments") (MF) está a uno si el datagrama no es el último fragmento. El campo Posición de Fragmento ("Fragment Offset") identifica la posición del fragmento, relativa al comienzo del datagrama original no fragmentado. Los fragmentos se miden en unidades de 8 octetos. La estrategia de fragmentación está diseñada de forma que un datagrama no fragmentado tiene toda su información de fragmentación a cero (MF = 0, posición de fragmento = 0). Si un datagrama internet es fragmentado, su parte de datos debe ser dividida en múltiplos de 8 octetos.

Este formato permite 2**13 = 8192 fragmentos de 8 octetos cada uno, para un total de 65,536 octetos. Nótese que esto es consistente con el campo de longitud total del datagrama (por supuesto, la cabecera se cuenta en la longitud total y no en los fragmentos).

Cuando tiene lugar la fragmentación, algunas opciones se copian, pero otras permanecen sólo en el primer fragmento.

Cada módulo internet debe ser capaz de transmitir un datagrama de 68 octetos sin necesidad de más fragmentación. Esto es porque una cabecera internet puede ser de hasta 60 octetos, y el fragmento mínimo son 8 octetos.

Todo destino internet debe ser capaz de recibir un datagrama de 576 octetos en una sola pieza o en fragmentos a reensamblar.

Los campos que pueden verse afectados por la fragmentación incluye a:

(1) el campo opciones
(2) el indicador Más Fragmentos
(3) la posición del fragmento
(4) el campo longitud de la cabecera internet
(5) el campo longitud total
(6) la suma de control de la cabecera

Si el indicador "Don't Fragment" (No Fragmentar) (DF) está a uno, entonces NO está permitida la fragmentación de este datagrama, si bien puede ser descartado. Esto puede utilizarse para prohibir la fragmentación en casos donde el host receptor no dispone de suficientes recursos para reensamblar los fragmentos internet.

Un ejemplo del uso de la característica "Don't Fragment" es aliviar la carga de un pequeño host. Un pequeño host podría tener un programa de arranque que acepte un datagrama, lo almacene en memoria y lo ejecute.

Los procedimientos de fragmentación y reensamblaje se describen de forma mucho más sencilla mediante ejemplos. Los siguientes procedimientos son implementaciones de ejemplo.

Notación general en los pseudo-programas siguientes: "=<" significa "menor o igual que", "#" significa "no igual", "=" significa "igual", "<-" significa "se le asigna". Además, "x to y" incluye 'x' y excluye 'y'; por ejemplo, "4 to 7" incluiría 4, 5 y 6 (pero no 7).

Un Ejemplo de Procedimiento de Fragmentación

Al datagrama de tamaño máximo que se puede transmitir a través de la siguiente red se le llama la unidad de transmisión máxima (maximun transmission unit (MTU)).

Si la longitud total es menor o igual la unidad de transmisión máxima entonces pasar este datagrama al siguiente paso en el procesamiento de datagrama; sino partir el datagrama en dos fragmentos, siendo el primero de tamaño máximo, y el segundo el resto del datagrama. El primer fragmento es pasado al siguiente paso en el procesamiento de datagramas, mientras que el segundo fragmento se pasa otra vez a este procedimiento en caso de ser todavía demasiado grande.

Notación:

FO Posición de Fragmento ("Fragment Offset")
IHL Long. de la cabecera internet (Internet Header Length)
DF Indicador No Fragmentar ("Don't Fragment")
MF indicador Más Fragmentos ("More Fragment")
TL Longitud total (Total Length)
OFO FO Previo ("Old Fragment Offset")
OIHL IHL Previo
OMF Indicador MF Previo (Old More Fragments)
OTL Longitud total previa (Old Total Length)
NFB Núm. de bloques de fragmento (Number of Fragment Blocks)
MTU Unidad de transmisión máxima (Maximum Transmission Unit)

Procedimiento:

SI TL =< MTU ENTONCES
Pasar este datagrama al siguiente paso en el procesamiento de datagramas.

SINO SI DF = 1 ENTONCES
descartar el datagrama

SINO
Para producir el primer fragmento:
(1)Copiar la cabecera internet original;
(2)OIHL <- IHL; OTL <- TL; OFO <- FO; OMF <- MF;
(3)NFB <- (MTU-IHL*4)/8;
(4)Adjuntar los primeros NFB*8 octetos de datos;
(5)Corregir la cabecera:
MF <- 1;TL <- (IHL*4)+(NFB*8);
Recalcular la Suma de Control;
(6)Pasar este datagrama al siguiente paso en el procesamiento de datagramas;

Para producir el segundo fragmento:
(7)Copiar de forma selectiva la cabecera internet (algunas opciones no se copian, ver definiciones de opción);
(8) Añadir los datos restantes;
(9)Corregir la cabecera:
IHL <- (((OIHL*4)-(long. de opciones no copiadas))+3)/4;

```
                TL <- OTL - NFB*8 - (OIHL-IHL)*4);
                FO <- OFO + NFB;  MF <- OMF;
                Recalcular Suma de Control;
        (10) Pasar este fragmento al test de fragmentación;
FIN.
```

En el procedimiento anterior cada fragmento (excepto el último) fue construido con el tamaño máximo permitido. Otra alternativa podría producir datagramas menores que el tamaño máximo. Por ejemplo, uno podría implementar un procedimiento de fragmentación que dividiera repetidamente los datagramas grandes en dos hasta que los fragmentos resultantes fueran de menor tamaño que el tamaño de la unidad de transmisión máxima.

Un ejemplo de Procedimiento de Reensamblaje

Para cada datagrama el identificador de buffer se calcula como la concatenación de los campos de origen, destino, protocolo e identificación. Si se trata de un datagrama completo (es decir, los campos Posición de Fragmento y Más Fragmentos son cero), entonces todos los recursos de reensamblaje asociados con este identificador de buffer son liberados y el datagrama se pasa al siguiente paso en el procesamiento de datagramas.

Si no hay a mano otros fragmentos con este identificador de buffer entonces se asignan recursos de reensamblaje. Los recursos de reensamblaje consisten en un buffer de datos, un buffer de cabecera, una tabla de bits de bloque de fragmento, un campo de longitud total de datos, y un temporizador. Los datos contenidos en el fragmento se guardan en el buffer de datos de acuerdo con su posición de fragmento y longitud y se modifican bits en la tabla de bits de bloques de fragmento correspondientes a los bloques de fragmento recibidos.

Si este es el primer fragmento (es decir, la posición del fragmento es cero) esta cabecera se guarda en el buffer de cabecera. Si este es el último fragmento (es decir, su campo Más Fragmentos es cero) se calcula la longitud total de los datos. Si este fragmento completa el datagrama (lo cual se comprueba mirando los bits puestos a uno en la tabla de bloques de fragmento), entonces el datagrama es enviado al siguiente paso en el procesamiento de datagramas; si no se le asigna al

temporizador el máximo de su valor actual y el valor del campo Tiempo de Vida de este fragmento; y por último la rutina de reensamblaje devuelve el control.

Si el temporizador se agota, se liberan todos los recursos de reensamblaje para este identificador de buffer. El valor inicial del temporizador es un límite inferior del tiempo de espera de reensamblaje. Esto es así porque el tiempo de espera será incrementado si el Tiempo de Vida del fragmento recién llegado es mayor que el valor actual del temporizador, pero no será decrementado si es menor. El valor máximo que este temporizador puede alcanzar es el tiempo de vida máximo (aproximadamente 4.25 minutos). La recomendación actual para el valor inicial del temporizador es 15 segundos. Este puede variar conforme la experiencia con este protocolo se acumule. Nótese que la elección del valor de este parámetro está relacionada con la capacidad de buffer disponible y la velocidad de datos del medio de transmisión; es decir, la velocidad de datos por el valor del temporizador es igual al tamaño del buffer (p. ej., 10Kb/s X 15s = 150 Kb).

Notación:

FO	Posición del Fragmento ("Fragment Offset")
IHL	Long. de la cabecera internet (Internet Header Length)
MF	indicador Más Fragmentos ("More Fragments")
TTL	Tiempo de Vida
NFB	Núm. de bloques de fragmento (Number of Fragment Blocks)
TL	Longitud total (Total Length)
TDL	Longitud total de Datos (Total Data Length)
BUFID	Identificador de buffer (Buffer Identifier)
RCVBT	Tabla de Bits de Fragmentos Recibidos (Fragment Received Bit Table)
TLB	Límite Inferior del Temporizador (Timer Lower Bound)

Procedimiento:

(1) BUFID <- origen|destino|protocolo|identificación;
(2) SI FO = 0 Y MF = 0
(3) ENTONCES SI está reservado el buffer con BUFID
(4) ENTONCES liberar todo el reensamblaje para este BUFID;
(5) Pasar el datagrama al siguiente paso;

FIN.
(6) SINO
SI no está reservado el buffer con BUFID
(7) ENTONCES Asignar recursos de reensamblaje con BUFID;
TEMPORIZADOR <- TLB; TDL <- 0;
(8) guardar los datos del fragmento en el buffer de datos con BUFID desde el octeto FO*8 al octeto (TL-(IHL*4))+FO*8;
(9) poner a uno los bits RCVBT desde FO a FO+((TL-(IHL*4)+7)/8);
(10) SI MF = 0 ENTONCES TDL <- TL-(IHL*4)+(FO*8)
(11) SI FO = 0 ENTONCES guardar cabecera en buffer de cabecera
(12) SI TDL # 0
(13) Y todos los bits RCVBT desde 0 a (TDL+7)/8 son uno
(14) ENTONCES TL <- TDL+(IHL*4)
(15) Pasar el datagrama al siguiente paso;
(16) liberar todos los recursos de reensamblaje para este BUFID;
FIN.

(17) TEMPORIZADOR <- MAX(TEMPORIZADOR, TTL);
(18) ceder control hasta siguiente fragmento o hasta que el temporizador se agote;
(19) El temporizador se ha agotado: vaciar todo el reensamblaje con este BUFID;
FIN.

En el caso en que dos o más fragmentos contengan los mismos datos tanto idénticamente como por solapamiento parcial, este procedimiento usará la copia llegada más recientemente en el buffer de datos y en el datagrama entregado.

Identificación

La elección del Indentificador de un datagrama está basada en la necesidad de proporcionar una forma de identificar de forma única un datagrama en particular. El módulo del protocolo que reensambla fragmentos los evalúa como pertenecientes al mismo datagrama si tienen el mismo origen, destino, protocolo e Identificador. De este modo el remitente debe elegir un Identificador que sea único para ese par origen-destino y ese protocolo, y durante el tiempo que el datagrama (o cualquier

fragmento suyo) pueda perdurar en internet.

Parece entonces que un módulo de protocolo que actúe de remitente necesite mantener una tabla de Identificadores, con una entrada por cada destino con el que haya comunicado en el último periodo de máxima duración de un paquete en internet.

Sin embargo, dado que el campo Identificador permite 65,536 valores diferentes, algún host puede ser capaz de usar simplemente identificadores únicos independientemente del destino.

Para algunos protocolos de nivel superior resulta adecuado elegir el identificador. Por ejemplo, los módulos de protocolo TCP pueden retransmitir un segmento TCP idéntico, y la probabilidad de una recepción correcta se vería ampliada si la retransmisión lleva el mismo identificador que la transmisión original, ya que se podrían usar fragmentos de cualquiera de los dos datagramas para construir un segmento TCP correcto.

Tipo de Servicio

El Tipo de servicio (Type Of Service (TOS)) se utiliza para la selección de la calidad del servicio internet. El tipo de servicio se especifica a través de los parámetros abstractos de precedencia, demora, rendimiento, y fiabilidad. Estos parámetros abstractos se hacen corresponder con parámetros de servicio efectivos de las redes particulares que el datagrama atraviesa.

Precedencia.	Medida independiente de la importancia de este datagrama.
Demora.	La entrega inmediata es importante para datagramas con esta indicación.
Rendimiento.	Una alta velocidad de datos es importante para datagramas con esta indicación.
Fiabilidad.	Para datagramas con esta indicación es importante un mayor nivel de esfuerzo para asegurar la entrega.

Por ejemplo, La ARPANET tiene un bit de prioridad, y la posibilidad de elegir entre mensajes "estándar" (tipo 0) y "no controlados" (tipo 3) (La elección entre mensajes de paquete

único o multipaquete puede considerarse también un parámetro de servicio). Los mensajes no controlados tienden a ser entregados con menos fiabilidad y a sufrir menos demora. Supóngase que un datagrama internet se va a enviar a través de ARPANET. Sea el tipo de servicio internet dado como:

Precendencia: 5
Demora: 0
Rendimiento:1
Fiabilidad: 1

En este ejemplo, la correspondencia de estos parámetros con aquellos disponibles para ARPANET sería poner a uno el bit de prioridad de ARPANET, ya que la precedencia Internet se sitúa en la mitad superior de su escala, seleccionar mensajes estándar ya que se han indicado los requerimientos de rendimiento y fiabilidad y no de demora. Se dan más detalles sobre correspondencias de servicio en "Correspondencias de Servicio ("Service Mappings") [8].

Tiempo de Vida

El tiempo de vida es establecido por el remitente al tiempo máximo que un datagrama puede estar en el sistema internet. Si el datagrama permanece en el sistema internet durante más tiempo que su tiempo de vida, entonces el datagrama debe ser destruido.

Este campo debe ser decrementado en cada punto donde se procese para reflejar el tiempo invertido en procesar el datagrama. Incluso si no existe información local acerca del tiempo realmente empleado, el campo debe decrementarse en 1. El tiempo es medido en unidades de segundo (es decir, el valor 1 significa un segundo). Por tanto, el tiempo de vida máximo son 255 segundos o 4.25 minutos. Como todo módulo que procese un datagrama debe decrementar el TTL por lo menos en uno incluso si lo procesa en menos de un segundo, debe pensarse en el TTL sólo como un límite superior del tiempo que un datagrama puede existir. La intención es hacer que los datagramas que no se pueden entregar sean descartados, y limitar el máximo periodo de vida del datagrama.

Algunos protocolos de conexión fiables de alto nivel se basan en la presunción de que los datagramas duplicados más viejos no llegarán después de pasar un cierto tiempo. El TTL es

para tales protocolos un medio de tener la seguridad de que su suposición se cumple.
Opciones

Las opciones son opcionales en cada datagrama, pero obligatorias en las implementaciones. Es decir, la presencia o ausencia de una opción es elección del remitente, pero cada módulo internet debe ser capaz de analizar cualquier opción. Puede haber varias opciones presentes en el campo opción.

Las opciones pueden no ocupar exactamente un espacio múltiplo de 32 bits. La cabecera debe ser completada con octetos de ceros. El primero de estos será interpretado como la opción fin-de-opciones, y el resto como el relleno de la cabecera internet.

Todo módulo internet debe ser capaz de actuar en cada opción. La Opción de Seguridad es obligatoria si se debe atravesar un tráfico clasificado, restringido o compartimentado.

Suma de Control

La suma de control de la cabecera internet es recalculada si la cabecera es modificada. Por ejemplo, una reducción del tiempo de vida, las adiciones o cambios en las opciones internet, o debido a la fragmentación. Esta suma de control a nivel internet está pensada para proteger de errores de transmisión los campos de la cabecera internet.

Existen algunas aplicaciones donde son aceptables unos pocos errores en los bits de datos, mientras que los retrasos por retransmisión no lo son. Si el protocolo internet impusiera la exactitud de datos tales aplicaciones no podrían ser soportadas.

Errores

Los errores en el protocolo internet pueden indicarse mediante los mensajes ICMP [3].

3.3.Interfaces

La descripción funcional de interfaces de usuario para IP

es, en el mejor de los casos, ficticia, ya que cada sistema operativo dispondrá de distintos recursos. En consecuencia, debemos advertir a los lectores que diferentes implementaciones IP pueden tener diferentes interfaces de usuario. Sin embargo, todas deben proporcionar un cierto conjunto mínimo de servicios para garantizar que todas las implementaciones IP pueden soportar la misma jerarquía de protocolos. Esta sección especifica las interfaces funcionales obligatorias para todas las implementaciones IP.

El protocolo internet interactúa con la red local por un lado y con un protocolo de nivel superior o una aplicación por el otro. En lo que sigue, el protocolo de nivel superior o aplicación (o incluso un programa pasarela) será llamado el "usuario", dado que es lo que usa el módulo internet. Como el protocolo internet es un protocolo de datagramas, se mantiene un mínimo de memoria o estado entre transmisiones de datagramas, y cada llamada del usuario al módulo del protocolo internet proporciona toda la información necesaria para que el IP ejecute el servicio pedido.

Interface Ejemplo de nivel superior

Las siguientes dos llamadas de ejemplo satisfacen los requisitos de la comunicación entre el usuario y el módulo del protocolo internet ("=>" significa 'devuelve'):

SEND (src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt => result) donde:

src = dirección de origen
dst = dirección de destino
prot = protocolo
TOS = tipo de servicio
TTL = tiempo de vida
BufPTR = puntero a buffer
len = longitud del buffer
Id= Identificador
DF = No Fragmentar
opt = datos de opción
result = respuesta
OK = datagrama enviado correctamente
Error = error en los argumentos o error en la red local
Nótese que la precedencia se incluye en el TOS y la seguridad/compartimiento se pasa como opción.

RECV (BufPTR, prot, => result, src, dst, TOS, len, opt)

donde:

BufPTR = puntero a buffer
prot = protocolo
result = respuesta
OK = datagrama recibido correctamente
Error = error en los argumentos
len = longitud del buffer
src = dirección de origen
dst = dirección de destino
TOS = tipo de servicio
opt = datos de opción

Cuando el usuario envía un datagrama, ejecuta la llamada SEND suministrando todos los argumentos. El módulo del protocolo internet, al recibir esta llamada, comprueba los argumentos y prepara y envía el mensaje. Si los argumentos son válidos y el datagrama es aceptado por la red local, la llamada retorna con éxito. Si los argumentos no son correctos, o bien el datagrama no es aceptado por la red local, la llamada retorna sin éxito. En retornos de llamada sin éxito, se debe construir un informe razonable sobre la causa del problema, pero los detalles de tales informes corresponden a las distintas implementaciones.

Cuando un datagrama llega al módulo del protocolo internet desde la red local, o hay una llamada RECV pendiente del usuario al que va dirigido o no la hay. En el primer caso, la llamada pendiente es satisfecha pasando la información desde el datagrama al usuario. En el segundo caso, se notifica al usuario de destino que tiene un datagrama pendiente. Si el usuario de destino no existe, se devuelve un mensaje de error ICMP al remitente, y los datos son desechados.

La notificación de un usuario puede ser a través de una pseudo interrupción o un mecanismo similar, correspondiente al entorno particular del sistema operativo de la implementación.

Una llamada RECV de usuario puede ser inmediatamente satisfecha por un datagrama pendiente, o bien quedar pendiente hasta que llegue un datagrama.

La dirección de origen se incluye en la llamada SEND en el caso de que el host remitente tenga varias direcciones (múltiples conexiones físicas o direcciones lógicas). El módulo internet debe

comprobar que la dirección de origen es una de las direcciones legales de este host.

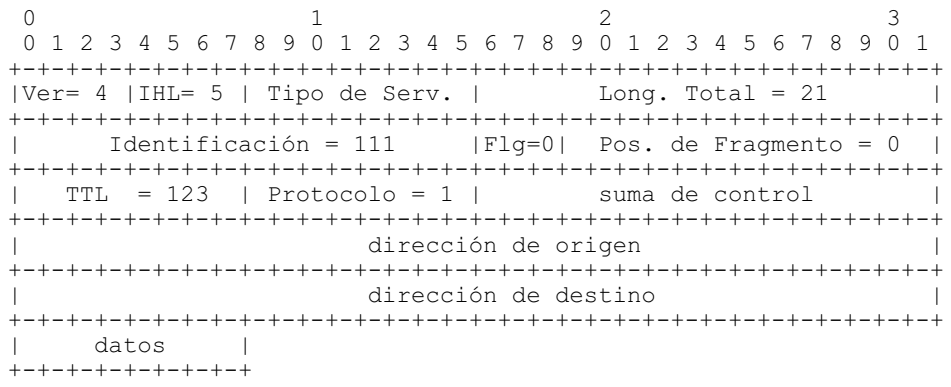
Una implementación también puede permitir o exigir una llamada al módulo internet para indicar interés en o reservar para uso exclusivo una clase de datagramas (p. ej., todos aquellos con un cierto valor en el campo protocolo).

Esta sección caracteriza funcionalmente una interfaz USUARIO/IP. La notación utilizada es similar a la mayoría de llamadas de función de lenguajes de alto nivel, pero este uso no pretende excluir las llamadas de servicio tipo 'trap' (p. ej., SVCs, UUOs, EMTs), o cualquier otra forma de comunicación entre procesos.

Apéndice A: Ejemplos y Escenarios

Ejemplo 1

Este es un ejemplo de un datagrama internet con un mínimo de datos:



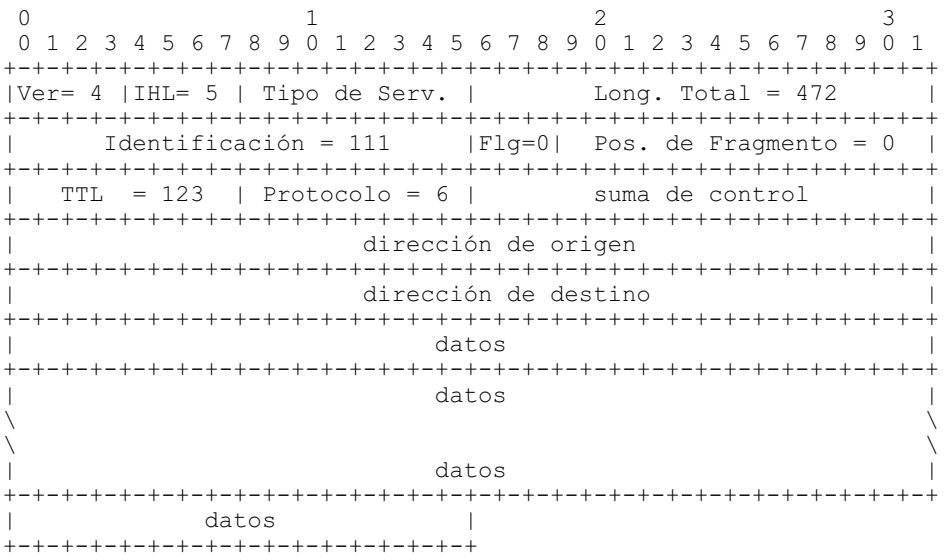
Ejemplo de Datagrama Internet

Figura 5.

Nótese que cada división representa una posición de un bit. Este es un datagrama internet en la versión 4 del protocolo internet; la cabecera internet consta de 5 palabras de 32 bits, y la longitud total del datagrama es de 21 octetos. Este datagrama es un datagrama completo (no un fragmento).

Ejemplo 2

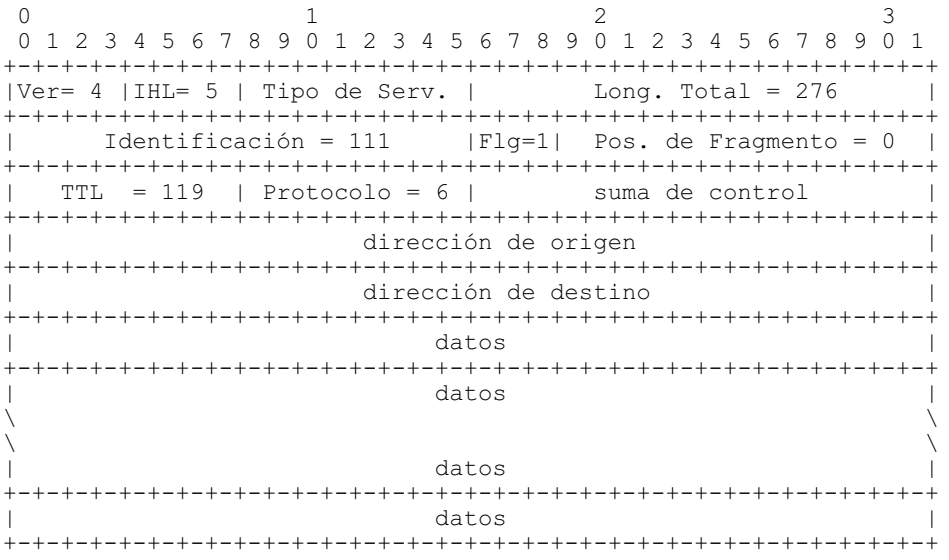
En este ejemplo, se muestra primero un datagrama internet de tamaño medio (452 octetos de datos), y después dos fragmentos internet que podrían ser resultado de la fragmentación de este datagrama si la transmisión de máximo tamaño permitida fuese de 280 octetos.



Ejemplo de Datagrama Internet

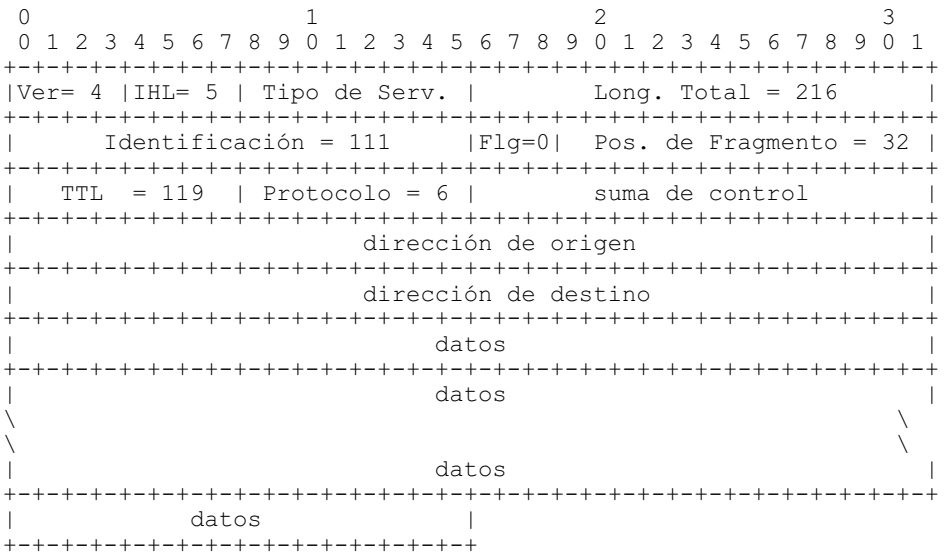
Figura 6.

Ahora el primer fragmento resultado de dividir el datagrama a los 256 octetos.



Ejemplo de Fragmento Internet
Figura 7.

Y el segundo fragmento.

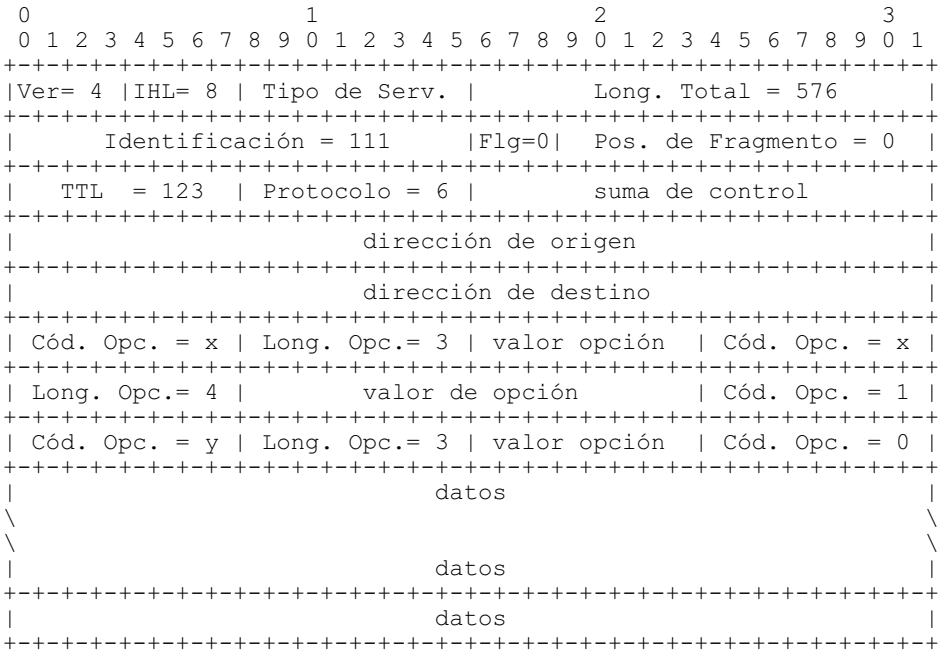


Ejemplo de Fragmento Internet

Figura 8.

Ejemplo 3:

Aqui se muestra un ejemplo de un datagrama que contiene opciones:

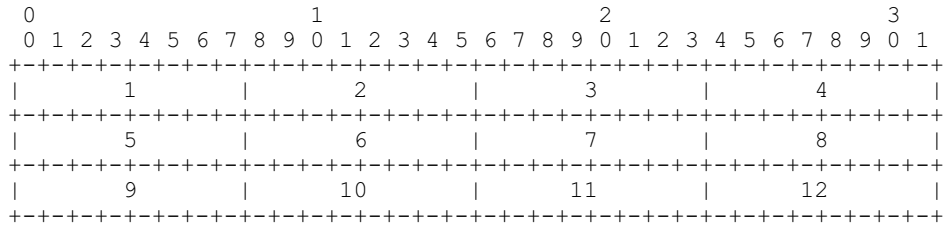


Ejemplo de Datagrama Internet

Figura 9.

Apéndice B: Orden de Transmisión de Datos

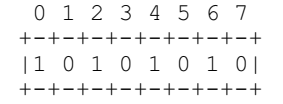
El orden de transmisión de la cabeceras y los datos descritos en este documento se resuelve a nivel de octeto. Cuando un diagrama muestra un grupo de octetos, el orden de transmisión de éstos es el orden normal en el cual se leen en Inglés. Por ejemplo, en el siguiente diagrama los octetos son transmitidos en el orden en el que van numerados.



Orden de Transmisión de Bytes

Figura 10.

Cuando un octeto representa una cantidad numérica el bit más a la izquierda en el diagrama es el bit de mayor orden o bit más significativo. Es decir, El bit etiquetado como 0 es el bit más significativo. Por ejemplo, el diagrama siguiente representa el valor 170 (decimal).



Significado de Bits

Figura 11.

De forma similar, cuando un campo multiocteto representa una cantidad numérica el bit más a la izquierda de todo el campo es el bit más significativo. Cuando una cantidad multi-octeto es transmitida se transmite primero el octeto más significativo.

Glosario

1822
: BBN Report 1822, "Especificación de la Interconexión de un Host y un IMP". La especificación de la interfaz entre un host y ARPANET.

ARPANET leader
: La información de control en un mensaje ARPANET en la interfaz host-IMP.

mensaje ARPANET
: La unidad de transmisión entre un host y un IMP en ARPANET. El tamaño máximo es aproximadamente 1012 octetos (8096 bits).

paquete ARPANET
: Una unidad de transmisión usada internamente entre IMPs en ARPANET. El tamaño máximo es aprox. 126 octetos (1008 bits).

Destino
: La dirección de destino, un campo de la cabecera internet.

DF
: El bit 'Don't Fragment' (No Fragmentar) del campo de indicadores.

Indicadores (Flags)
: Un campo de la cabecera internet con varios indicadores de control.

Posición del Fragmento
: Este campo de la cabecera internet indica a que lugar del datagrama internet pertenece un fragmento.

GGP
: Gateway to Gateway Protocol (Protocolo Pasarela a Pasarela), el protocolo usado principalmente entre pasarelas para controlar el encaminamiento y otras funciones de pasarela.

cabecera
: Información de control al principio de un mensaje, segmento, datagrama, paquete o bloque de datos.

ICMP
: Internet Control Message Protocol (Protocolo de Mensajes de Control de Internet), implementado en el módulo internet, el ICMP es usado de pasarelas a hosts y entre hosts para informar de errores y hacer sugerencias de encaminamiento.

Identificación
: Un campo de la cabecera internet que almacena el valor identificador asignado por el remitente como ayuda para ensamblar los fragmentos de un datagrama.

IHL
: El campo de la cabecera internet 'Internet Header Length' (Longitud de la cabecera internet) es la longitud de la cabecera internet medida en palabras de 32 bits.

IMP
: Interface Message Processor (Procesador de Mensajes de Interfaz), el intercambiador de paquetes de ARPANET.

Dirección Internet
: Una dirección de origen o destino de 4 octetos (32 bits) formada por un campo de Red y un campo de Dirección Local.

Datagrama internet
: La unidad de datos intercambiada entre un par de módulos internet (incluye la cabecera internet).

Fragmento internet
: Una parte de los datos de un datagrama internet con una cabecera internet.

Dirección Local
: La dirección de un host en una red. La relación real entre una dirección local internet con las direcciones de un host en una red es muy general, permitiéndose relaciones de muchos a uno.

MF
: El indicador 'More-Fragments' (Más Fragmentos) presente en el campo indicadores de la cabecera internet.

módulo
: Una implementación, normalmente en software, de un

protocolo u otro procedimiento.

indicador 'more-fragments'
: Un indicador que dice si este datagrama internet contiene el final de un datagrama internet, presente en el campo Indicadores de la cabecera internet.

NFB
: Number of Fragment Blocks (Número de Bloques de Fragmento) en la parte de datos de un fragmento internet. Es decir, la longitud de una parte de datos medida en unidades de 8 octetos.

octeto
: Un byte de 8 bits.

Opciones
: El campo Opciones de la cabecera internet puede contener varias opciones, y cada una de ellas puede ser de varios octetos de longitud.

Valor de Relleno (Padding)
: El campo Valor de Relleno de la cabecera internet se utiliza para asegurar que el campo de datos comienza en una dirección múltiplo de 32 bits. El relleno es cero.

Protocolo
: En este documento, un campo de la cabecera internet, el identificador del protocolo del siguiente nivel superior.

Resto
: La parte de dirección local de una dirección internet.

Origen
: La dirección de origen, un campo de la cabecera internet.

TCP
: Transmission Control Protocol (Protocolo de Control de Transmisión): Un protocolo host-a-host para comunicación fiable en entornos internet.

Segmento TCP
: La unidad de datos intercambiada entre dos módulos TCP (incluyendo la cabecera TCP).

TFTP

Trivial File Transfer Protocol (Protocolo de Transferencia de Archivos Trivial): Un sencillo protocolo de transferencia de archivos construido sobre UDP).

Tiempo de Vida
: Un campo de la cabecera internet que indica el límite superior de cuánto tiempo puede existir el datagrama.

TOS
: Type of Service (Tipo de Servicio)

Longitud Total
: El campo de la cabecera Internet 'Longitud Total' es la longitud del datagrama en octetos, incluyendo cabecera y datos.

TTL
: Time to Live (Tiempo de Vida)

Tipo de Servicio
: Un campo de la cabecera internet que indica el tipo (o calidad) de servicio para este datagrama.

UDP
: User Datagram Protocol (Protocolo de Datagrama de Usuario): Un protocolo a nivel de usuario para aplicaciones orientadas a transacciones.

Usuario
: El usuario del protocolo internet. Este puede ser un módulo de protocolo de nivel superior, una aplicación, o un programa pasarela.

Versión
: El campo Versión indica el formato de una cabecera internet.

Referencias

[1] Cerf, V., "The Catenet Model for Internetworking", Information Processing Techniques Office, Defense Advanced Research Projects Agency, IEN 48, Julio 1978.

[2] Bolt Beranek and Newman, "Specification for the Interconnection of a Host and an IMP", BBN Technical Report 1822, Revisado Mayo 1978.

[3] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification",RFC 792, USC/Information Sciences Institute, Septiembre 1981.

[4] Shoch, J., "InterNetwork Naming, Addressing, and Routing", COMPCON, IEEE Computer Society, Otoño 1978.

[5] Postel, J., "Address Mappings", RFC 796, USC/Information Sciences Institute, Septiembre 1981.

[6] Shoch, J., "Packet Fragmentation in InterNetwork Protocols", Computer Networks, v. 3, n. 1, Febrero 1979.

[7] Strazisar, V., "How to Build a Gateway", IEN 109, Bolt Beranek andNewman, Agosto 1979.

[8] Postel, J., "Service Mappings", RFC 795, USC/Information Sciences Institute, Septiembre 1981.

[9] Postel, J., "Assigned Numbers", RFC 790, USC/Information Sciences Institute, Septiembre 1981.

Introducción

El Protocolo Internet (IP) [1] se utiliza para el servicio de datagramas de "host" a "host" en un sistema de redes interconectadas denominado Catenet [2]. Los dispositivos de conexión de redes se denominan Pasarelas (Gateways). Estas pasarelas se comunican entre ellas con propósito de control mediante el Protocolo Pasarela a Pasarela (Gateway to Gateway Protocol (GGP)) [3,4]. Ocasionalmente, una pasarela o un "host" de destino se comunicará con un "host" de origen para, por ejemplo, informar de un error en el procesamiento de datagramas. El Protocolo de Mensajes de Control Internet (ICMP) se usa para este propósito. ICMP utiliza el soporte básico de IP como si se tratara de un protocolo de nivel superior. Sin embargo, ICMP es realmente una parte integrante de IP, y debe ser implementado por todo módulo IP.

Los mensajes ICMP son enviados en varias situaciones: por ejemplo, cuando un datagrama no puede alcanzar su destino, cuando una pasarela no dispone de capacidad de almacenamiento temporal para reenviar el datagrama, y cuando la pasarela puede dirigir al "host" para enviar el tráfico por una ruta más corta.

El Protocolo Internet no está diseñado para ser absolutamente fiable. El propósito de estos mensajes de control no es hacer a IP fiable, sino suministrar información sobre los problemas en el entorno de comunicación. Sigue sin garantizarse que un datagrama sea entregado o que se devuelva un mensaje de control. Existe la posibilidad de que algunos datagramas no sean entregados, sin ningún informe sobre su pérdida. Los protocolos de nivel superior que usen IP deben implementar sus propios procedimientos de fiabilidad en caso de que requieran comunicación fiable.

Típicamente, los mensajes ICMP informan de errores en el procesamiento de datagramas. Para evitar la generación sin fin de mensajes acerca de mensajes, etc..., no se envían mensajes ICMP acerca de mensajes ICMP. Además sólo se envían mensajes ICMP acerca de errores en el procesamiento del

fragmento cero de un datagrama fragmentado. (el fragmento cero es el que tiene el campo posición ("offset") de fragmento igual a cero).

Formatos de Mensaje

Los mensajes ICMP se envían usando la cabecera IP básica. El primer octeto de la parte de datos del datagrama es el campo de tipo ICMP; el valor de este campo determina el formato del resto de los datos.

Los campos etiquetados como "no usado" están reservados para posteriores extensiones y deben ser cero al ser enviados, y los receptores no deberán usar estos campos (excepto para incluirlos en la suma de control). Exceptuando las descripciones de formato individuales en las que se indique lo contrario, los valores de los campos de la cabecera internet son como sigue:

Version

4

IHL ("Internet Header Length")

Longitud de la cabecera internet en palabras de 32 bits.

Tipo de Servicio

0

Longitud Total

Longitud de la cabecera y los datos en octetos.

Identificación, Indicadores ("flags") y Posición De Fragmento

Usados en fragmentación, ver [1].

Tiempo de Vida (TTL, "Time To Live")

Tiempo de vida en segundos; como este valor se decrementa en cada máquina en la cual el datagrama es procesado, debe ser al menos igual o mayor que el número de pasarelas que atravesará.

Protocolo

ICMP = 1

Suma de Control de Cabecera

El complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

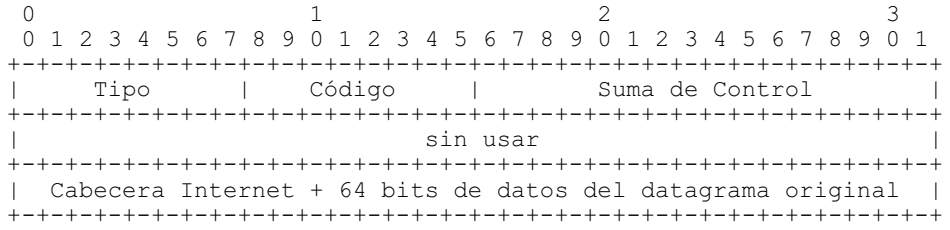
Dirección de Origen

La dirección de la pasarela o "host" que crea el mensaje ICMP. Si no se indica lo contrario, puede ser cualquiera de las direcciones de una pasarela.

Dirección de Destino

La dirección de la pasarela o "host" al cual se debe enviar el mensaje.

Mensaje de Destino Inaccesible



Campos IP:

Dirección de Destino

La red y dirección del origen extraídos de los datos del datagrama original.

Campos ICMP:

Tipo

3

Código

0 = red inaccesible;

1 = "host" inaccesible;

2 = protocolo inaccesible;

3 = puerto inaccesible;

4 = se necesitaba fragmentación pero DF estaba activado;

5 = fallo en la ruta de origen.

Suma de Control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Cabecera internet + 64 bits de Datagrama de Datos

La cabecera Internet más los primeros 64 bits de datos del datagrama original. Este dato lo utiliza el "host" para asociar el mensaje al proceso apropiado. Si un protocolo de nivel superior utiliza números de puerto, se asume que están en los primeros 64 bits de datos del datagrama original.

Descripción

Si, de acuerdo con la información existente en las tablas de enrutamiento de la pasarela, la red especificada en el campo de destino internet de un datagrama es inaccesible, p. ej., si la distancia a la red es infinita, la pasarela puede de enviar un mensaje de destino inaccesible al "host" de origen del

datagrama.

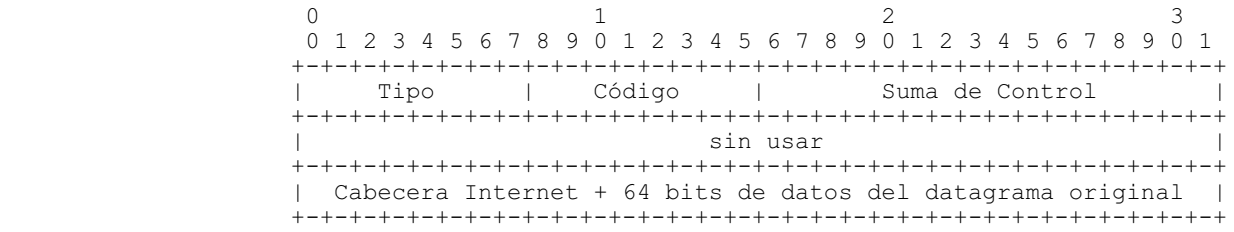
Además, en algunas redes, la pasarela puede ser capaz de determinar si el "host" de destino en internet es inalcanzable. Las pasarelas de estas redes pueden enviar al "host" de origen mensajes de destino inaccesible cuando el "host" de destino sea inaccesible.

Si en el "host" de destino el módulo IP no puede enviar el datagrama debido a que el módulo de protocolo o el puerto del proceso indicado no estén activos, puede enviar un mensaje de destino inaccesible al "host" de origen.

Otro caso se presenta cuando un datagrama debe ser fragmentado para poder ser enviado a través de una pasarela aún cuando el indicador "Don't Fragment" (No Fragmentar) esté activado. En este caso la pasarela debe desechar el datagrama y puede devolver un mensaje de destino inaccesible.

Los códigos 0, 1, 4 y 5 pueden ser recibidos desde una pasarela. Los códigos 2 y 3 pueden ser recibidos desde un "host".

Mensaje de Tiempo Superado



Campos IP:

Dirección de Destino

La red y dirección del origen extraídos de los datos del datagrama original.

Campos ICMP:

Tipo

11

Código

0 = tiempo de vida superado en tránsito;

1 = tiempo de reensamblaje de fragmentos superado.

Suma de Control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Cabecera internet + 64 bits de Datagrama de Datos

La cabecera Internet más los primeros 64 bits de datos del datagrama original. Este dato lo utiliza el "host" para asociar el mensaje al proceso apropiado. Si un protocolo de nivel superior utiliza números de puerto, se asume que están en los primeros 64 bits de datos del datagrama original.

Descripción

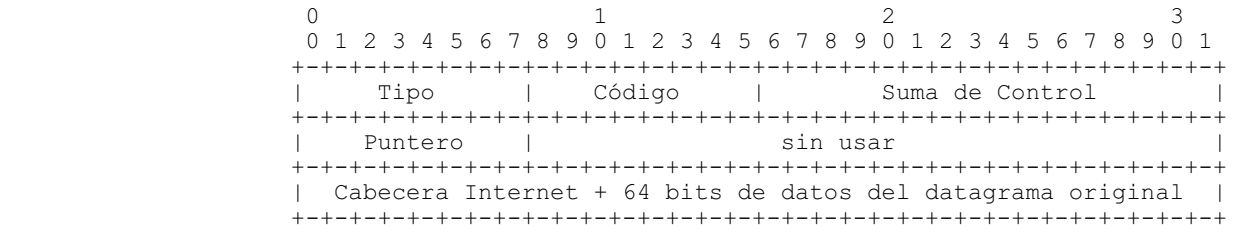
Si la pasarela que está procesando el datagrama detecta que el campo tiempo de vida es cero debe desechar el datagrama. La pasarela puede también notificar el suceso al "host" de origen mediante el mensaje de tiempo de vida superado.

Si un "host" que trata de reensamblar un datagrama fragmentado no puede hacerlo en el tiempo límite debido a fragmentos perdidos, descartará el datagrama y puede enviar un mensaje de tiempo de reensamblaje superado.

Si el fragmento cero no está disponible no es necesario enviar ningún mensaje de tiempo superado.

El código 0 puede ser recibido desde una pasarela. El código 1 desde un "host".

Mensaje de Problema de Parámetros



Campos IP:

Dirección de Destino

La red y dirección del origen extraídos de los datos del datagrama original.

Campos ICMP:

Tipo

12

Código

0 = el puntero indica el error.

Suma de Control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Puntero

Si código = 0, identifica el octeto donde se detectó el error.

Cabecera internet + 64 bits de Datagrama de Datos

La cabecera Internet más los primeros 64 bits de datos del datagrama original. Este dato lo utiliza el "host" para asociar el

mensaje al proceso apropiado. Si un protocolo de nivel superior utiliza números de puerto, se asume que están en los primeros 64 bits de datos del datagrama original.

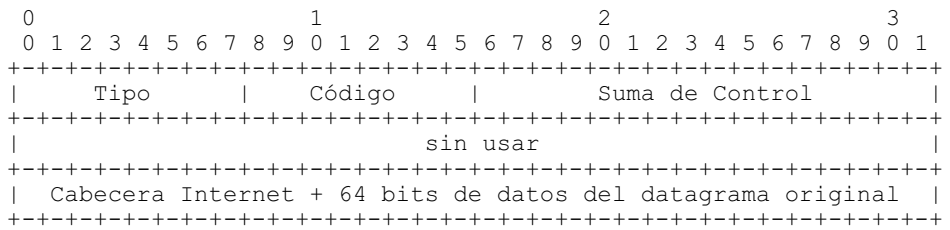
Descripción

Si la pasarela o "host" que procesa el datagrama encuentra un problema con los parámetros de cabecera, de modo que no puede completar el procesamiento del datagrama, debe desecharlo. Una potencial fuente de este tipo de problema son los argumentos incorrectos en una opción. La pasarela o "host" puede también notificarlo al "host" de origen mediante el mensaje de Problema de Parámetros. Este mensaje sólo se envía si el error provocó que el datagrama fuera desechado.

El puntero identifica el octeto de la cabecera del datagrama original donde fue detectado el error (puede estar en medio de una opción). Por ejemplo, 1 indica que algo va mal con el Tipo de Servicio y (si hay opciones presentes) 20 indica un error en el código de tipo de la primera opción.

El código 0 puede ser recibido desde una pasarela o un "host".

Mensaje de Disminución del Tráfico desde el Origen



Campos IP:

Dirección de Destino

La red y dirección del origen extraídos de los datos del datagrama original.

Campos ICMP:

Tipo

4

Código

0

Suma de Control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Cabecera internet + 64 bits de Datagrama de Datos

La cabecera Internet más los primeros 64 bits de datos del datagrama original. Este dato lo utiliza el "host" para asociar el mensaje al proceso apropiado. Si un protocolo de nivel superior utiliza números de puerto, se asume que están en los primeros 64 bits de datos del datagrama original.

Descripción

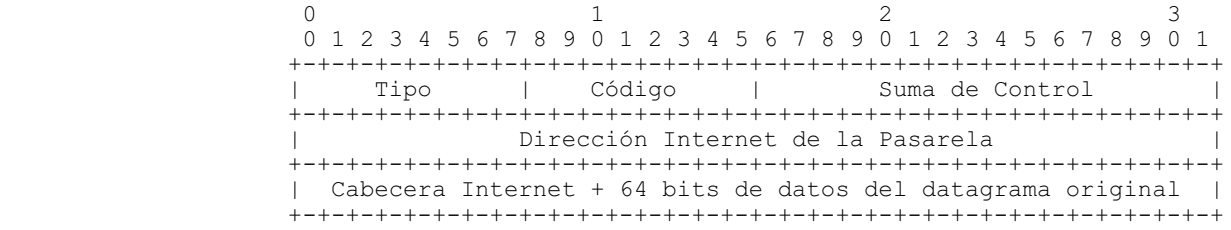
Una pasarela puede descartar datagramas de internet si no dispone del espacio de búfer suficiente para ponerlos en la cola de salida hacia la próxima red de la ruta a la red de destino. Si una pasarela descarta un datagrama por este motivo, puede enviar un mensaje de Disminución de Tráfico desde el Origen (DTO) al "host" de origen del datagrama. Un "host" de destino puede también enviar un DTO si los datagramas llegan demasiado rápido para ser procesados. El DTO es una petición al "host" para que reduzca el ritmo al que envía tráfico al "host" de destino. Una pasarela puede enviar un DTO por cada mensaje que descarta. Al recibir un DTO, el "host" de origen debe disminuir el ritmo de generación de tráfico al destino especificado hasta que deje de recibir DTOs de la pasarela. Después, el "host" de origen puede aumentar gradualmente la frecuencia de mensajes al destino hasta que vuelva a recibir DTOs.

La pasarela o "host" puede enviar el DTO cuando se está acercando al límite de su capacidad, antes que esperar a que

ésta se sobrepase. Esto significa que el datagrama de datos que provocó el DTO puede que sea enviado.

El Código 0 puede ser recibido desde un "host" o una pasarela.

Mensaje de Redirección



Campos IP:

Dirección de Destino

La red y dirección del origen extraídos de los datos del datagrama original.

Campos ICMP:

Tipo

5

Código

0 = Redirigir datagramas debido a la Red.

1 = Redirigir datagramas debido al "host".

2 = Redirigir datagramas debido al Tipo de Servicio y la Red.

3 = Redirigir datagramas debido al Tipo de Servicio y el "host".

Suma de Control

El complemento a uno de 16 bits de la suma de los

complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Dirección Internet de la Pasarela

Dirección de la Pasarela a la cual se debe dirigir el tráfico destinado a la red especificada en el campo 'red de destino internet' de los datos del datagrama original.

Cabecera internet + 64 bits de Datagrama de Datos

La cabecera Internet más los primeros 64 bits de datos del datagrama original. Este dato lo utiliza el "host" para asociar el mensaje al proceso apropiado. Si un protocolo de nivel superior utiliza números de puerto, se asume que están en los primeros 64 bits de datos del datagrama original.

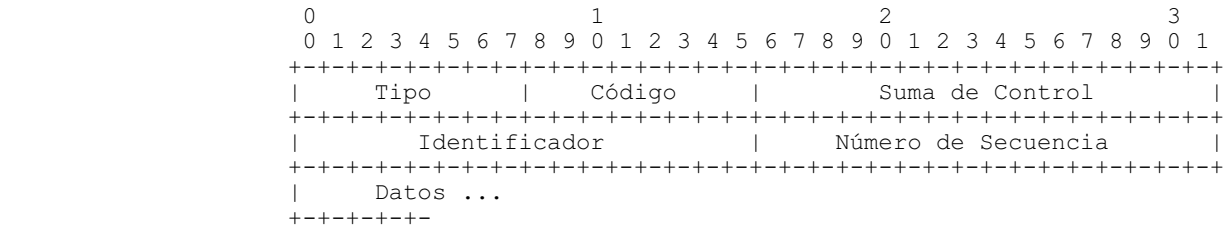
Descripción

La pasarela envía un mensaje de redirección a un "host" en la siguiente situación: Una pasarela, G1, recibe un datagrama internet de un "host" en una red a la cual la pasarela está conectada. G1 comprueba su tabla de encaminamiento y obtiene la dirección de la siguiente pasarela, G2, en la ruta hacia la red X, destino del datagrama en internet. Si G2 y el "host" identificado por la dirección internet de origen del datagrama están en la misma red, se envía un mensaje de redirección al "host". Un mensaje de redirección recomienda al "host" que dirija el tráfico destinado a la red X directamente a la pasarela G2, ya que se trata de un camino más corto hacia el destino. La pasarela reenvía el datagrama original a su destino en internet.

No se envía ningún mensaje de redirección para aquellos datagramas con opciones IP de 'ruta de origen' y la dirección de la pasarela en el campo dirección de destino, incluso si existe una ruta mejor al destino final que la que pasa por la siguiente dirección en la ruta de origen.

Los códigos 0, 1, 2 y 3 pueden ser recibidos desde una pasarela.

Mensaje de Eco o de Respuesta de Eco



Campos IP:

Direcciones

La dirección del origen en un mensaje de eco será la del destino del mensaje de respuesta de eco. Para componer un mensaje de respuesta de eco, simplemente se invierten las direcciones de origen y destino, el código de tipo se cambia a 0 y se vuelve a calcular la suma de control.

Campos ICMP:

Tipo

8 para mensaje de eco;

0 para mensaje de respuesta de eco.

Código

0

Suma de Control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Si la longitud total es impar, los datos recibidos son completados con un octeto de ceros para calcular la suma de control. Esta suma de control puede ser sustituida en el futuro.

Identificador

Si código = 0, un identificador como referencia para emparejar ecos y respuestas, que puede ser cero.

Número de Secuencia

Si código = 0, un número de secuencia como referencia para emparejar ecos y respuestas, que puede ser cero.

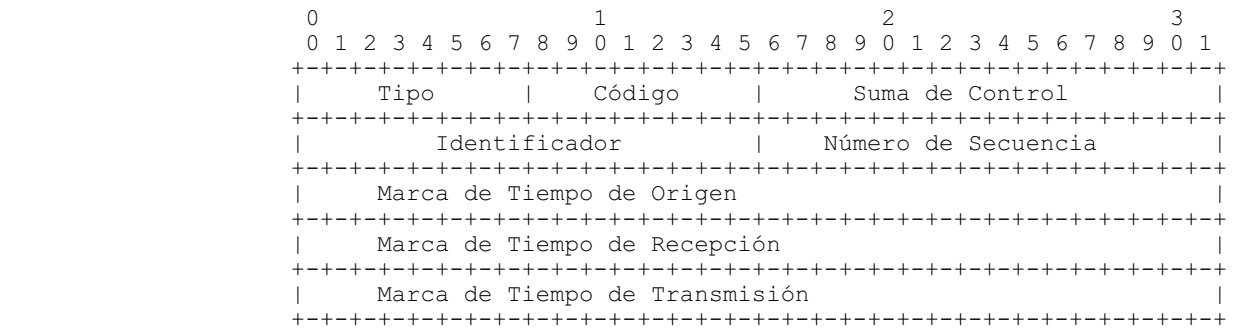
Descripción

Los datos recibidos en el mensaje de eco deben ser devueltos en el mensaje de respuesta de eco.

El identificador y número de secuencia pueden ser usados por el emisor del eco como referencia para emparejar las respuestas con las peticiones de eco. Por ejemplo, el identificador podría usarse como un puerto en TCP o UDP para identificar una sesión, y el número de secuencia se iría incrementando con cada nueva petición de eco enviada. El "host" que hace eco devuelve estos mismos valores en la respuesta de eco.

El código 0 puede ser recibido desde un "host" o una pasarela.

Mensaje de Solicitud de Marca de Tiempo o de Respuesta de Marca de Tiempo



Campos IP:

Direcciones

La dirección del origen en un mensaje de marca de tiempo será la del destinatario del mensaje de respuesta. Para formar un mensaje de respuesta de marca de tiempo, simplemente se intercambian las direcciones de origen y destino, se cambia el código de tipo a 14 y se vuelve a calcular la suma de control.

Campos ICMP:

Tipo

13 para el mensaje de solicitud de marca de tiempo;

14 para el mensaje de respuesta.

Código

0

Suma de Control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Identificador

Si Código = 0, entonces es un identificador, que puede ser cero, que se usa para hacer corresponder mensajes de marca de tiempo con sus respectivas respuestas.

Número de Secuencia

Si Código = 0, entonces es número de secuencia, que puede ser cero, que se usa para hacer corresponder mensajes de marca de tiempo con sus respectivas respuestas

Descripción

Los datos recibidos (una marca de tiempo) en el mensaje

son devueltos en la respuesta junto con marcas de tiempo adicionales. La marca de tiempo es un entero de 32 bits que indica los milisegundos transcurridos desde la medianoche UT. Un posible uso de estas marcas de tiempo se describe en Mills [5].

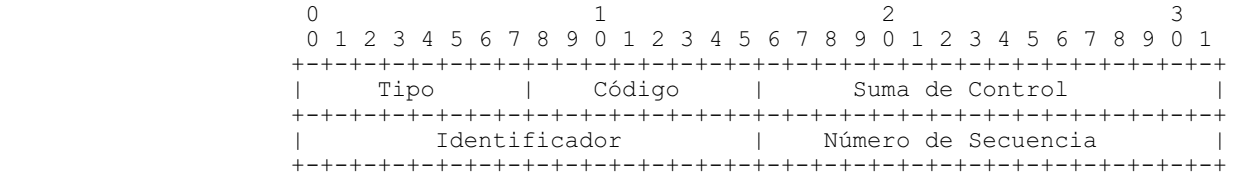
La Marca de Tiempo de Origen es el instante en el cual el mensaje fue manipulado por última vez por el emisor antes de enviarlo. La Marca de Tiempo de Recepción es el instante en el cual el destinatario recibe el mensaje. Por último, la Marca de Tiempo de Transmisión es el momento en el cual el destinatario manipula el mensaje por última vez antes de enviarlo.

Si la medida del tiempo no está disponible en milisegundos, o bien no puede ser indicada respecto a la medianoche UT, entonces se puede insertar cualquier valor de tiempo en la marca de tiempo, siempre y cuando el bit más significativo de la marca de tiempo sea puesto a uno para indicar que se trata de un valor no estándar.

El Identificador y Número de Secuencia pueden ser usados por el emisor del eco como ayuda para relacionar las respuestas con sus respectivas peticiones. Por ejemplo, el identificador puede usarse como un puerto en TCP o UDP para identificar una sesión, y el número de secuencia podría ser incrementado con cada petición enviada. El destinatario devuelve estos mismos valores en la respuesta.

El Código 0 puede ser recibido desde una pasarela o un "host".

Mensaje de Solicitud de Información o de Respuesta de Información



Campos IP:

Direcciones

La dirección del origen en un mensaje de solicitud de información será la dirección del destinatario del mensaje de respuesta. Para formar un mensaje de respuesta, simplemente se intercambian las direcciones de origen y destino, se cambia el código de tipo a 16 y se vuelve a calcular la suma de control.

Campos ICMP:

Tipo

15 para mensaje de solicitud de información;

16 para mensaje de respuesta;

Código

0

Suma de control

El complemento a uno de 16 bits de la suma de los complementos a uno del mensaje ICMP, comenzando por el Tipo ICMP. A la hora de calcular la suma de control, el valor inicial de este campo es cero. Esta suma de control puede ser sustituida en el futuro.

Identificador

Si Código = 0, entonces es un identificador, que puede ser cero, y se usa para hacer corresponder mensajes de respuesta con sus respectivas solicitudes.

Número de Secuencia

Si Código = 0, entonces es número de secuencia, que puede ser cero, y se usa para hacer corresponder mensajes de respuesta con sus respectivas solicitudes.

Descripción

Este mensaje puede ser enviado indicando en el campo dirección de origen de la cabecera IP la dirección de red de origen y los campos de dirección de destino puestos a cero (lo cual indica "esta" red).

Este mensaje puede ser enviado con la parte de red de la dirección de origen de la cabecera IP tomando un valor cero (lo que significa "esta red").

El módulo IP que ha de responder debería enviar la respuesta con las direcciones completamente especificadas. Este es un mensaje mediante el cual un "host" puede saber el número de la red en la que se encuentra.

El Identificador y Número de Secuencia pueden ser usado por el emisor del eco como ayuda para relacionar las respuestas con sus respectivas peticiones. Por ejemplo, el identificador puede usarse como un puerto en TCP o UDP para identificar una sesión, y el número de secuencia podría ser incrementado con cada petición enviada. El destinatario devuelve estos mismos valores en la respuesta.

El Código 0 puede ser recibido desde una pasarela o un "host".

Resumen de los Tipos de Mensajes

0	Eco Respuesta ("Echo Reply")
3	Destino Inaccesible ("Destination Unreachable")
4	Disminución del tráfico desde el origen ("Source Quench")
5	Redirección ("Redirect")
8	Eco ("Echo")
11	Tiempo Superado ("Time Exceeded")
12	Problema de Parámetros ("Parameter Problem")
13	Marca de Tiempo ("Timestamp")
14	Respuesta de Marca de Tiempo ("Timestamp Reply")
15	Solicitud de Información ("Information Request")
16	Respuesta de Información ("Information Reply")

Referencias

[1] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute,Septiembre 1981.

[2] Cerf, V., "The Catenet Model for Internetworking", IEN 48, Information Processing Techniques Office, Defense Advanced Research Projects Agency, Julio 1978.

[3] Strazisar, V., "Gateway Routing: An Implementation Specification", IEN 30, Bolt Beranek and Newman, April 1979.

[4] Strazisar, V., "How to Build a Gateway", IEN 109, Bolt Beranek andNewman, Agosto 1979.

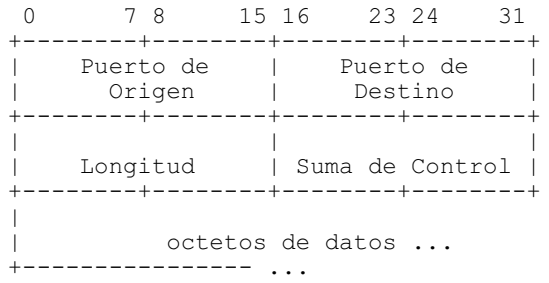
[5] Mills, D., "DCNET Internet Clock Service", RFC 778, COMSAT Laboratories, Abril 1981.

Introducción

Este Protocolo de Datagramas de Usuario (UDP: User Datagram Protocol) se define con la intención de hacer disponible un tipo de datagramas para la comunicación por intercambio de paquetes entre ordenadores en el entorno de un conjunto interconectado de redes de computadoras. Este protocolo asume que el Protocolo de Internet (IP: Internet Protocol) [1] se utiliza como protocolo subyacente.

Este protocolo aporta un procedimiento para que los programas de aplicación puedan enviar mensajes a otros programas con un mínimo de mecanismo de protocolo. El protocolo se orienta a transacciones, y tanto la entrega como la protección ante duplicados no se garantizan. Las aplicaciones que requieran de una entrega fiable y ordenada de secuencias de datos deberían utilizar el Protocolo de Control de Transmisión (TCP: Transmission Control Protocol). [2]

Formato



Formato de la Cabecera de un Datagrama de Usuario

Campos

El campo Puerto de Origen es opcional; cuando tiene sentido, indica el puerto del proceso emisor, y puede que se asuma que sea el puerto al cual la respuesta debería ser dirigida en ausencia de otra información. Si no se utiliza, se inserta un

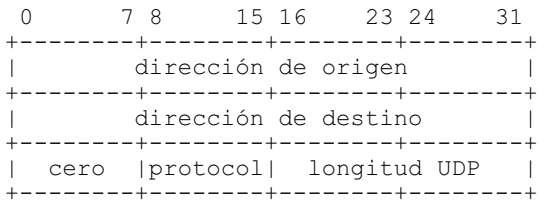
valor cero.

El campo Puerto de Destino tiene significado dentro del contexto de una dirección de destino en un entorno internet particular.

El campo Longitud representa la longitud en octetos de este datagrama de usuario, incluyendo la cabecera y los datos. (Esto implica que el valor mínimo del campo Longitud es ocho.)

El campo Suma de Control (Checksum) es el complemento a uno de 16 bits de la suma de los complementos a uno de las palabras de la combinación de una pseudo-cabecera construida con información de la cabecera IP, la cabecera UDP y los datos, y rellenada con octetos de valor cero en la parte final (si es necesario) hasta tener un múltiplo de dos octetos.

La pseudo-cabecera que imaginariamente antecede a la cabecera UDP contiene la dirección de origen, la dirección de destino, el protocolo y la longitud UDP. Esta información proporciona protección frente a datagramas mal encaminados. Este procedimiento de comprobación es el mismo que el utilizado en TCP.



Si la suma de control calculada es cero, se transmite como un campo de unos (el equivalente en la aritmética del complemento a uno). Un valor de la suma de control trasmitido como un campo de ceros significa que el emisor no genera la suma de control (para depuración o para protocolos de más alto nivel a los que este campo les sea indiferente).

Interfaz de Usuario

Una interfaz de usuario debería permitir: la creación de nuevos puertos de recepción, operaciones de recepción en los puertos de recepción que devuelvan los octetos de datos y una indicación del puerto de origen y de la dirección de origen, y una operación que permita enviar un datagrama, especificando los

datos y los puertos de origen y de destino y las direcciones a las que se debe enviar.

Interfaz IP

El módulo UDP debe ser capaz de determinar las direcciones de origen y destino en un entorno internet así como el campo de protocolo de la cabecera del protocolo internet. Una posible interfaz UDP/IP devolvería el datagrama de internet completo, incluyendo toda la cabecera, en respuesta a una operación de recepción. Una interfaz de este tipo permitiría también al módulo UDP pasar un datagrama de internet completo con cabecera al módulo IP para ser enviado. IP verificaría ciertos campos por consistencia y calcularía la suma de control de la cabecera del protocolo internet.

Aplicación del Protocolo

Los usos principales de este protocolo son el Servidor de Nombres de Internet [3] y la Transferencia Trivial de Ficheros (Trivial File Transfer) [4].

Número del protocolo

Este es el protocolo 17 (21 en octal) cuando se utilice en el Protocolo de Internet (IP). Se indican otros números de protocolo en [5].

Referencias

[1] Postel, J.,"Internet Protocol", RFC 760, USC/Information Sciences Institute, Enero de 1980.

[2] Postel, J., "Transmission Control Protocol", RFC 761, USC/Information Sciences Institute, Enero de 1980.

[3] Postel, J., "Internet Name Server", USC/Information Sciences Institute, IEN 116, Agosto de 1979.

[4] Sollins, K., "The TFTP Protocol", Massachusetts Institute of Technology, IEN 133, Enero de 1980.

[5] Postel, J., "Assigned Numbers", USC/Information Sciences Institute, RFC 762, Enero de 1980.

Prefacio

Este documento describe el protocolo, estándar del DoD, de control de la transmisión (TCP). Ha habido nueve ediciones previas de la especificación de TCP por ARPA sobre las que el presente texto se basa enormemente. Ha habido muchos participantes en este trabajo tanto en términos de conceptos como de texto. Esta edición clarifica varios detalles y elimina los ajustes de tamaño de búfer al "final de carta" y reescribe el mecanismo de carta como una función de entrega inmediata.

Jon Postel
Editor

1. Introducción

El "protocolo de control de transmisión" ('Transmission Control Protocol', TCP) está pensado para ser utilizado como un protocolo 'host' a 'host' muy fiable entre miembros de redes de comunicación de computadoras por intercambio de paquetes y en un sistema interconectado de tales redes.

Este documento describe las funciones que debe realizar el protocolo de control de transmisión, el programa que lo implementa, y su interfaz con los programas o usuarios que requieran de sus servicios.

1.1. Motivación

Los sistemas de comunicación entre computadoras están jugando un papel cada vez más importante en entornos militares, gubernamentales y civiles. Este documento centra principalmente su atención en los requisitos militares de comunicación entre computadoras, especialmente la robustez bajo comunicaciones no plenamente fiables y la disponibilidad ante congestiones, aunque muchos de estos problemas pueden encontrarse igualmente en los sectores civil y gubernamental.

A la par que las redes estratégicas y tácticas de comunicación entre computadoras están siendo desarrolladas y desplegadas, es esencial proporcionar medios de interconexión entre ellas y proporcionar protocolos estándares de comunicación entre procesos que puedan soportar un amplio rango de aplicaciones. Anticipando la necesidad de tales estándares, la subsecretaría de defensa del congreso de los diputados para la investigación e ingeniería ('the Deputy Undersecretary of Defense for Research and Engineering') ha declarado el protocolo de transmisión de control (TCP) descrito aquí como la base para la estandarización de los protocolos de comunicación entre procesos, dentro del ámbito de todo el Departamento de Defensa (DoD).

TCP es un protocolo orientado a la conexión, fiable y entre dos extremos, diseñado para encajar en una jerarquía en capas de protocolos que soportan aplicaciones sobre múltiples redes. TCP proporciona mecanismos para la comunicación fiable entre pares de procesos en computadoras 'host' ancladas en redes de comunicación de computadoras distintas, pero interconectadas.

Se hacen muy pocas suposiciones sobre la fiabilidad de los protocolos de comunicación por debajo de la capa de TCP. TCP sólo supone que puede acceder a un servicio de transmisión de datagramas simple, aunque en principio poco fiable, de los protocolos del nivel inferior. En principio, TCP debería ser capaz de operar encima de un amplio espectro de sistemas de comunicaciones que incluye desde conexiones por cables fijos ('hard-wired conections') hasta redes de intercambio de paquetes o redes de circuitos conmutados.

TCP se basa en los conceptos descritos primeramente por Cerf y Kahn en [1]. TCP encaja en una arquitectura de protocolos en capas justo por encima del protocolo de internet [2], protocolo básico que proporciona un medio para TCP de enviar y recibir segmentos de longitud variable de información envuelta en "sobres" de datagramas de internet. El datagrama de internet proporciona un medio de direccionar TCPs de origen y de destino situados en redes diferentes. El protocolo de internet también trata con la fragmentación y el reensamble de segmentos de TCP que sean necesarios para conseguir el transporte y la entrega sobre múltiples redes y las puertas de enlace que las interconectan. El protocolo de internet también lleva información sobre la prioridad, clasificación de seguridad y compartimentación de los segmentos de TCP, de tal forma que esta información pueda ser comunicada de extremo a extremo entre múltiples redes.

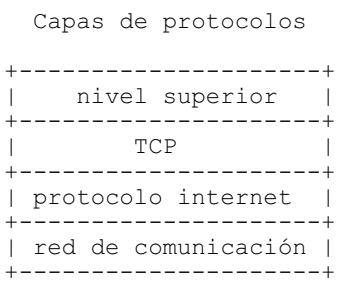


Figura 1

Gran parte de este documento se ha escrito dentro del contexto de las implementaciones de TCP que son corresidentes con protocolos de más alto nivel en la computadora anfitriona. Algunos sistemas de computadoras se conectarán a las redes vía computadoras intermediarias ('front-end computers') que alojan las capas de protocolos TCP e internet, a la vez que el software específico de redes. Esta especificación de TCP describe una interfaz con los protocolos de mayor nivel que resulta ser implementable incluso para el caso de computadoras

intermediarias, siempre y cuando se implemente un adecuado protocolo entre el 'host' y la computadora intermediaria.

1.2. Ámbito

TCP está pensado para proporcionar un servicio fiable de comunicación entre procesos en un entorno con múltiples redes. TCP está pensado para ser un protocolo de 'host' a 'host' de uso común en redes múltiples.

1.3. Sobre este documento

Este documento representa una especificación del comportamiento requerido a cualquier implementación de TCP, tanto en sus interacciones con protocolos de más alto nivel como con sus interacciones con otros TCP. El resto de esta sección ofrece una muy breve visión de las interfaces y operaciones del protocolo. La sección 2 resume los presupuestos de corte filosófica para el diseño de TCP. La sección 3 ofrece tanto una descripción detallada de las acciones que se le exigen a TCP cuando varios eventos acontecen (llegada de nuevos segmentos, llamadas de usuario, errores, etc.) así como los detalles de los formatos de los segmentos TCP.

1.4. Interfaces

TCP presenta interfaz por un lado con el usuario o los procesos de aplicación y por el otro con un protocolo de más bajo nivel como es el protocolo de internet (IP).

La interfaz entre un proceso de aplicación y TCP se ilustrará con un detalle razonable. Esta interfaz consiste en un conjunto de llamadas, de forma muy similar a las llamadas que un sistema operativo proporciona a los procesos de aplicación para manipular ficheros. Por ejemplo, hay llamadas para abrir y cerrar conexiones y para enviar y recibir datos por las conexiones establecidas. Se exige también que TCP pueda comunicarse asíncronamente con los programas de aplicación.

Aunque se deja considerable libertad a los fabricantes de implementaciones de TCP a la hora de diseñar las interfaces que

sean apropiadas para el entorno de un sistema operativo particular, se exige un mínimo de funcionalidad en la interfaz TCP/usuario de cualquier implementación válida.

La interfaz entre TCP y el protocolo de nivel inferior queda esencialmente sin especificar, exceptuando el hecho de que se asume que hay un mecanismo por el cual los dos niveles pueden pasar información asíncronamente el uno al otro. Típicamente, se espera que el protocolo de nivel inferior especifique esta interfaz. Se ha diseñado TCP para trabajar en un entorno muy genérico de redes interconectadas. El nivel inferior que se asumirá a lo largo de este documento es el protocolo de internet [2].

1.5.Operación

Como se ha hecho notar más arriba, el propósito principal de TCP consiste en proporcionar un servicio de conexión o circuito lógico fiable y seguro entre pares de procesos. Para proporcionar este servicio encima de un entorno de internet menos fiable, el sistema de comunicación requiere de mecanismos relacionados con las siguientes áreas:

Transferencia básica de datos
Fiabilidad
Control de flujo
Multiplexamiento
Conexiones
Prioridad y seguridad

La operación básica de TCP en cada uno de estas áreas se describe en los siguiente párrafos:

Transferencia básica de datos

TCP es capaz de transferir un flujo continuo de octetos en cada sentido entre sus usuarios empaquetando un cierto número de octetos en segmentos para su transmisión a través del sistema de internet. En general, los módulos de TCP deciden cuándo bloquear y enviar datos según su propia conveniencia.

Algunas veces los usuarios necesitan estar seguros de que todos los datos que habían entregado al módulo de TCP han sido

transmitidos. Para este propósito se define una función 'push' ("enviar inmediatamente"). Para asegurar que los datos entregados al módulo de TCP son realmente transmitidos, el usuario emisor debe indicarlo mediante la función 'push'. Un 'push' en un cierto instante causa que los módulos de TCP envíen y entreguen inmediatamente al usuario receptor los datos almacenados hasta ese instante. El instante exacto en que se ejecuta la función 'push' podría no ser visible para el usuario receptor. Tampoco la función 'push' proporciona una marca de límite de registros.

Fiabilidad

El módulo de TCP debe poder recuperar los datos que se corrompan, pierdan, dupliquen o se entreguen desordenados por el sistema de comunicación del entorno de internet. Esto se consigue asignando un número de secuencia a cada octeto transmitido, y exigiendo un acuse de recibo del módulo de TCP receptor. Si no se recibe un ACK dentro de un cierto plazo de expiración prefijado, los datos se retransmiten. En el receptor, se utilizan los números de secuencia para ordenar correctamente los segmentos que puedan haber llegado desordenados y para eliminar los duplicados. La corrupción de datos se trata añadiendo un campo de suma de control ('checksum') a cada segmento transmitido, comprobándose en el receptor y descartando los segmentos dañados.

En tanto en cuanto los módulos de TCP continúen funcionando adecuadamente y el sistema de internet no llegue a quedar particionado de forma completa, los errores de transmisión no afectarán la correcta entrega de datos. TCP se recupera de los errores del sistema de comunicación de internet.

Flujo de control

TCP proporciona al receptor un medio para controlar la cantidad de datos enviados por el emisor. Esto se consigue devolviendo una "ventana" con cada ACK, indicando el rango de números de secuencia aceptables más allá del último segmento recibido con éxito. La ventana indica el número de octetos que se permite que el emisor transmita antes de que reciba el siguiente permiso.

Multiplexamiento

Para permitir que muchos procesos dentro de un único 'host' utilicen simultáneamente las posibilidades de comunicación de TCP, el módulo de TCP proporciona una serie de direcciones o puertos dentro de cada 'host'. Concatenadas con las direcciones de red y de 'host' de la capa de comunicación internet conforman lo que se denomina una dirección de conector ('socket'). Un par de direcciones de conector identifica de forma única la conexión. Es decir, un conector puede utilizarse simultáneamente en múltiples conexiones.

La asignación de puertos a los procesos se gestiona de forma independiente en cada 'host'. Sin embargo, resulta de la máxima utilidad asignar a los procesos más utilizados frecuentemente (i.e., un gestor de registros ('logger') o un servicio compartido) conectores fijos que se hacen conocer de forma pública. Estos servicios pueden entonces ser accedidos a través de direcciones conocidas públicamente. El establecimiento y aprendizaje de las direcciones de los puertos de otros procesos puede involucrar otros mecanismos más dinámicos.

Conexiones

La fiabilidad y los mecanismos de control de flujo descritos más arriba exigen que los módulos de TCP inicialicen y mantengan una información de estado para cada flujo de datos. La combinación de esta información, incluyendo las direcciones de los conectores, los números de secuencia y los tamaños de las ventanas, se denomina una conexión. Cada conexión queda especificada de forma única por un par de conectores que corresponden con sus dos extremos.

Cuando dos procesos desean comunicarse, sus módulos de TCP deben establecer primero una conexión (inicializar la información de estado en cada lado). Cuando la comunicación se ha completado, la conexión se termina o cierra con la intención de liberar recursos para otros usos.

Como las conexiones tienen que establecerse entre 'hosts' no fiables y sobre un sistema de comunicación internet no fiable, se utiliza un mecanismo de acuerdo que usa números de secuencia basados en tiempos de reloj para evitar una inicialización errónea de las conexiones.

Prioridad y seguridad

Los usuarios de TCP pueden indicar el nivel de seguridad y prioridad de su comunicación. Se emplean valores por defecto cuando estas características no se necesiten.

2. Filosofía

2.1. Elementos del sistema de internet.

El entorno de internet consiste en una serie de 'hosts' conectados a varias redes que a su vez están interconectadas vía pasarelas ('gateras'). Aquí se supone que las redes pueden ser tanto redes locales (i.e. de tipo de ETHERNET) o grandes redes (i.e. ARPANET), pero en cualquier caso basadas en tecnología de intercambio de paquetes. Los agentes activos que producen y consumen los mensajes son los procesos. Varios niveles de protocolos en las redes, las pasarelas y 'hosts' dan soporte a un sistema de comunicación entre procesos que proporciona un flujo de datos bidireccional sobre conexiones lógicas entre los puertos de los procesos.

El término paquete o trama se utiliza generalmente aquí para significar el conjunto de datos de una transacción entre un 'host' y su red. El formato de bloques de datos intercambiados dentro de una red, en general, no nos importará aquí.

Los 'hosts' son computadoras unidas a una red, y, desde el punto de vista de la comunicación en la red, constituyen los orígenes y destinos de los paquetes. Los procesos han de verse como los elementos activos en los computadores anfitriones (de acuerdo con la definición bastante común de un proceso como un programa en ejecución). Incluso los terminales y ficheros u otros dispositivos de E/S son vistos como comunicándose unos con otros a través del uso de procesos. Así, toda comunicación puede verse como una comunicación entre procesos.

Ya que un proceso puede necesitar distinguir entre varios flujos de comunicación entre él mismo y otro proceso (o procesos), se supone que un proceso puede tener un cierto número de puertos a través de los cuales se comunica con los puertos de otros procesos.

2.2.Modelo de operación

Los procesos transmiten datos llamando al módulo de TCP y pasando búferes de datos como argumentos de la llamada. El módulo de TCP empaqueta en segmentos los datos provenientes de estos búferes y efectúa una llamada al módulo de internet para que transmita cada segmento al módulo de TCP de destino. El TCP receptor coloca los datos de un segmento en el búfer de recepción del usuario y lo notifica al usuario receptor. Los módulos de TCP incluyen información de control en los segmentos que puede ser utilizada para asegurar una transmisión fiable y ordenada de datos.

El modelo de comunicación del protocolo de internet es de tal forma que hay un módulo del protocolo internet asociado con cada módulo de TCP y que, a su vez, proporciona una interfaz con la red local. Este módulo de internet empaqueta los segmentos de TCP dentro de los datagramas de internet y encamina estos datagramas hacia un módulo de internet de destino por una pasarela intermedia. Para poder transmitir el datagrama a través de la red local, se encapsula dentro de un paquete de red local.

Los conmutadores de paquetes en la red pueden realizar ulteriores empaquetamientos, fragmentaciones o cualquier otra operación necesaria para conseguir la entrega del paquete local al módulo de internet de destino.

En una pasarela entre redes, el datagrama de internet es "desenvuelto" a partir del paquete de red local y examinado para determinar a través de qué red debería el datagrama viajar a continuación. El datagrama de internet es entonces "envuelto" otra vez por un paquete de red apropiado, enviado a la red siguiente y encaminado hacia la siguiente pasarela, o hacia el destino final.

Se permite que una pasarela divida un datagrama de internet en fragmentos de datagrama más pequeños, si esto es necesario para la transmisión a través de la siguiente red. Para hacer esto, la pasarela produce un conjunto de datagramas de internet; cada uno de ellos transportando un fragmento. Los fragmentos pueden ser subdivididos a su vez en ulteriores pasarelas. El formato de fragmento de datagrama de internet se ha diseñado de tal forma que el módulo de internet de destino

puede reensamblar los fragmentos en datagramas de internet.

El módulo de internet de destino desenvuelve el segmento del datagrama (después de haber reensamblado el datagrama, si así era necesario) y lo pasa al módulo de TCP de destino.

Este modelo simple de operación descrito disimula muchos detalles. Una característica importante es el tipo de servicio. Éste proporciona información a la pasarela (o a un módulo de internet) para guiarla en la selección de los parámetros de servicio a utilizar para atravesar la siguiente red. Entre la información del tipo de servicio se encuentra el grado de prioridad del datagrama. Los datagramas pueden también transportar información de seguridad que permitan a los 'hosts' y a las pasarelas que operen en entornos de seguridad multinivel separar adecuadamente datagramas por razones de seguridad.

2.3. El entorno del 'host'

Se supone que el módulo de TCP lo es del sistema operativo. Los usuarios accederán a dicho módulo de una forma muy similar a como acceden al sistema de archivos. El módulo de TCP, a su vez, puede llamar a otras funciones del sistema operativo, por ejemplo, para gestionar las estructuras de datos. Se supone que la interfaz real final con la red se controla mediante un módulo manejador del dispositivo ('device driver') de red. El módulo de TCP no llama directamente al manejador, sino que efectúa llamadas al módulo del protocolo de datagramas de internet que en su lugar llama al manejador del dispositivo de red.

Los mecanismos de TCP no impiden la implementación de TCP en un procesador intermediario ('front-end'). Sin embargo, en una implementación de este tipo, un protocolo de 'host' a 'front-end' debe proporcionar la funcionalidad necesaria para soportar el tipo de interfaz TCP-usuario descrita en este documento.

2.4. Interfaces

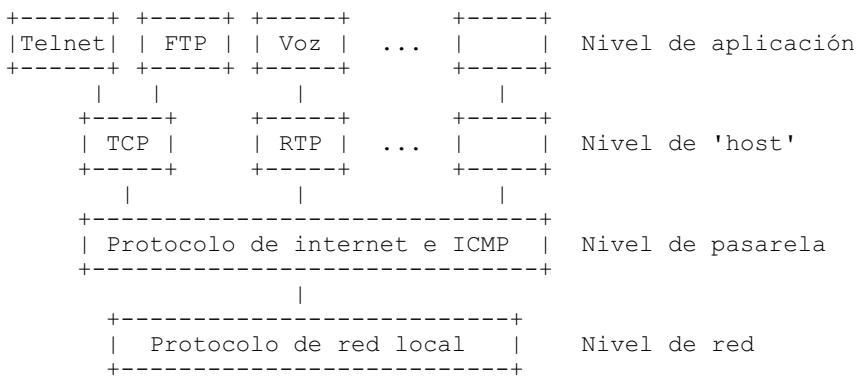
La interfaz TCP/usuario proporciona al usuario funciones de llamada al módulo de TCP para abrir (OPEN) o cerrar (CLOSE) una conexión, para enviar (SEND) o recibir (RECEIVE) datos, o

para obtener información de estado (STATUS) sobre una conexión. Estas llamadas son del mismo tipo que otras llamadas al sistema operativo realizadas desde programas de usuario como, por ejemplo, las llamadas para abrir, leer y cerrar un fichero.

La interfaz TCP/internet proporciona llamadas para enviar y recibir datagramas direccionados a los módulos TCP en cualquier 'host' del sistema de internet. Estas llamadas tienen parámetros para pasar la dirección, el tipo de servicio, la prioridad y otra información de control.

2.5. Relación con otros protocolos

El siguiente diagrama ilustra el lugar de TCP en la jerarquía de protocolos:



Relación entre protocolos

Figura 2.

Se espera que TCP sea capaz de soportar protocolos de nivel superior eficientemente. Debería ser fácil implementar la interfaz de TCP con los protocolos de nivel superior como Telnet de ARPANET o AUTODIN II THP.

2.6. Comunicación fiable

Un flujo de datos enviado sobre una conexión de TCP se entrega de forma fiable y ordenada al destino.

La transmisión es fiable gracias al uso de números de secuencia y de acuses de recibo. Básicamente, se le asigna un

número de secuencia a cada octeto de datos. El número de secuencia del primer octeto de datos en un segmento se transmite con ese segmento y se le denomina el número de secuencia del segmento. Los segmentos también llevan un número de acuse de recibo que es el número de secuencia del siguiente octeto de datos esperado en la transmisión en el sentido inverso.

Cuando el módulo de TCP transmite un segmento conteniendo datos, pone una copia en una cola de retransmisión e inicia un contador de tiempo; si llega el acuse de recibo para esos de datos, el segmento se borra de la cola. Si no se recibe el acuse de recibo dentro de un plazo de expiración, el segmento se retransmite.

La llegada del acuse de recibo no garantiza que los datos ya hayan sido entregados al usuario final, sino únicamente que el TCP receptor ha asumido la responsabilidad de hacerlo.

Para controlar el flujo de datos entre los módulos de TCP, se utiliza un mecanismo de flujo de control. El TCP receptor devuelve una "ventana" al TCP emisor. Esta ventana especifica el número de octetos, a contar a partir del número del acuse de recibo, que el TCP receptor está en ese momento preparado para recibir.

2.7. Establecimiento y finalización de la conexión

Para identificar los distintos flujos de datos que un módulo TCP puede manejar simultáneamente, TCP proporciona un identificador de puerto. Como los identificadores de puertos son seleccionados de forma independiente por cada TCP, puede que no sean únicos. Para disponer de direcciones únicas dentro de cada TCP, se concatena la dirección de internet que identifica al módulo de TCP con el identificador de puerto para así conformar una dirección de conector ('socket') que será única a largo de todo el conjunto de redes interconectadas.

Una conexión queda completamente especificada por el par de conectores de sus extremos. Un conector local puede participar en muchas conexiones con diferentes conectores foráneos. Una conexión puede ser utilizada para transportar datos en los dos sentidos, es decir, es bidireccional ('full duplex').

Los módulos de TCP pueden asociar libremente los puertos a procesos. Sin embargo, algunos conceptos básicos son necesarios en cualquier implementación. Debe haber un conjunto de conectores públicos que el módulo de TCP asocie sólo con los procesos "apropiados" por algún medio. Se supone la posibilidad de que los procesos puedan "poseer" puertos, y que los procesos puedan iniciar conexiones sólo con los puertos que ellos posean. (Qué medios son necesarios para implementar estas posesiones es una cuestión local, pero se supone la existencia de un comando de usuario de petición de puerto ('Request Port'), o un método de asignar de forma única un grupo de puertos a un proceso dado, por ejemplo, la asociación de los bits más significativos de un numero de puerto con un proceso dado).

Una conexión se especifica en la llamada de apertura OPEN con los argumentos de un puerto local y una dirección de conector remoto. Como respuesta, el módulo de TCP proporciona un nombre local (corto) a la conexión con la que el usuario puede referirse a la conexión en subsiguientes llamadas. Hay varias cosas que deben recordarse acerca de una conexión. Para guardar esta información podemos imaginar que existe una estructura llamada "Bloque de control de transmisión" (TCB, 'Transmission Control Block'). Una estrategia para su implementación tendría el nombre local de la conexión como un puntero al TCB de esta conexión. La llamada OPEN también especifica si se persigue de forma activa el establecimiento de la conexión o si se espera de forma pasiva.

Una petición pasiva OPEN significa que el proceso desea aceptar peticiones de conexión entrantes en vez de intentar iniciar directamente una conexión. A menudo, el proceso solicitante de un OPEN pasivo aceptará una petición posterior de conexión proveniente de cualquiera. En este caso, la dirección de conector remoto igual a todo ceros se utiliza para denotar un conector sin especificar. Sólo se permiten llamadas OPEN pasivas a conectores remotos de este tipo.

Un proceso de servicio que deseara proporcionar servicios a otros procesos desconocidos lanzaría una petición de OPEN pasivo contra un conector remoto sin especificar. Entonces, se podría establecer una conexión con cualquier proceso que haga una petición de conexión al conector local. Sería de gran ayuda si es público el conector local al que está asociado este servicio.

El uso de conectores públicos ('well-known') constituye un mecanismo conveniente para la asociación a priori de direcciones

de conectores con un servicio estándar. Por ejemplo, el proceso "Servidor de telnet" está permanentemente asignado a un conector particular, y del mismo modo se reservan otros conectores para los procesos "Transferencia de ficheros" ('File Transfer'), "Entrada de trabajos remotos" ('Remote Job Entry'), "Generador de texto" ('Text Generator'), "Generador de eco" ('Echoer') y "Sumidero" ('Sink') (el propósito de los últimos tres servicios es la realización de pruebas). Una dirección de conector podría ser reservada para el acceso a un servicio de "guía" que podría devolver la dirección de conector específica en la que se proporcione un servicio nuevo. El concepto de un conector público es parte de la especificación de TCP, pero la asignación de conectores con servicios queda fuera de esta especificación. (Ver [4]).

Los procesos pueden ejecutar OPEN pasivos y esperar a otros OPEN activos que concuerden y provengan de otros procesos, siendo entonces informados por el módulo de TCP cuando las conexiones se hayan finalmente establecido. Dos procesos que se dirigen OPEN activos entre sí al mismo tiempo serán conectados correctamente. Esta flexibilidad es crítica para dar soporte a la computación distribuida en la que los componentes pueden actuar asíncronamente unos con respecto a otros.

Hay dos casos principales para la concordancia de conectores entre OPEN pasivos locales y OPEN activos remotos. En el primer caso, un OPEN pasivo local ha especificado completamente el conector remoto. En este caso, la concordancia debe ser exacta. En el segundo caso, los OPEN pasivos locales han dejado el conector remoto sin especificar. En este caso, cualquier conector remoto es aceptable en tanto en cuanto los conectores locales concuerden. Otras posibilidades incluirían concordancias parcialmente restringidas.

Si hay varios OPEN pasivos pendientes (registrados en varios TCB) con el mismo conector local, un OPEN activo remoto concordará con el TCB que contenga el conector remoto específico en el campo OPEN activo remoto, si existe un TCB así, antes de seleccionar un TCB con un conector remoto sin especificar.

Los procedimientos para establecer conexiones utilizan el indicador de control de sincronización (SYN) e involucran un intercambio de tres mensajes. Se ha denominado a este intercambio como el acuerdo en tres pasos ('three-way

handshake') [3].

Una conexión se inicia ante la coincidencia de un segmento entrante que contiene un SYN y una entrada de TCB en espera previa, habiendo sido ambos creados por un comando de usuario OPEN. La concordancia de conectores locales y remotos determina cuándo una conexión ha sido iniciada. La conexión queda "establecida" cuando los números de secuencia quedan sincronizados en ambos sentidos.

La finalización de una conexión también involucra el intercambio de segmentos, en este caso transportando un indicador de control FIN.

2.8. Comunicación de datos

Puede pensarse en los datos que fluyen en una conexión como en un flujo de octetos. El usuario emisor indica en cada llamada SEND mediante la puesta a uno del indicador PUSH si los datos de esa llamada (y de cualquier llamada precedente) deben ser entregados inmediatamente al usuario receptor.

Se permite a un TCP emisor recolectar datos del usuario emisor y enviar esos datos en segmentos según propia conveniencia, siempre y cuando no se invoque la función 'push', en ese caso, el TCP emisor debe enviar todos los datos pendientes. Cuando el TCP receptor ve el indicador PUSH, no debe esperar más datos del TCP receptor antes de pasar los datos al proceso receptor.

No hay necesariamente una relación entre el uso de funciones 'push' y el tamaño de los segmentos. Los datos de cualquier segmento en particular pueden ser el resultado, total o parcial, de una llamada SEND única, o de múltiples llamadas SEND.

El propósito de la función 'push' y del indicador PUSH consiste exclusivamente en transportar inmediatamente los datos desde el usuario emisor al usuario receptor. No proporciona en ningún caso un servicio de registro.

Sí existe una estrecha relación entre el uso de la función 'push' y el uso de los búferes de datos empleados en la interfaz TCP/usuario. Cada vez que un indicador PUSH se asocia con

datos colocados en el búfer del usuario receptor, su contenido se devuelve al usuario para ser procesado incluso si el búfer no se ha llenado. Si los datos que llegan llenan el búfer de usuario antes de que se lea un indicador PUSH, los datos se pasan al usuario en las unidades de tamaño del búfer.

TCP también proporciona un mecanismo para comunicar al receptor de los datos que en algún punto más adelante en el flujo de datos que está actualmente leyendo habrá datos urgentes. TCP no intenta definir lo que el usuario debe hacer en concreto cuando se le notifica la existencia de datos urgentes pendientes, sólo da la noción general de que el proceso receptor tendrá que tomar medidas para procesar los datos urgentes de forma rápida.

2.9. Prioridad y seguridad

TCP hace uso del campo de tipo de servicio del protocolo de internet y de la opción de seguridad para proporcionar prioridad y seguridad a los usuarios de TCP, tomando como base cada conexión. No todos los módulos de TCP trabajarán necesariamente en un entorno de seguridad multinivel; algunos puede que estén limitados a usos reservados solamente, y otros puede que operen en un único nivel de seguridad y compartimentación. Consecuentemente, algunas implementaciones y servicios de TCP para usuarios puede que estén limitados a un subconjunto del caso con seguridad multinivel.

Los módulos de TCP que operen en un entorno con seguridad multinivel deben marcan apropiadamente los segmentos salientes con los niveles de seguridad, compartimentación y prioridad. Estos módulos TCP deben también proporcionar a sus usuarios o a los protocolos de más alto nivel como Telnet o THP una interfaz que les permita especificar el grado de seguridad, compartimentación y prioridad de las conexiones.

2.10. Principio de robustez

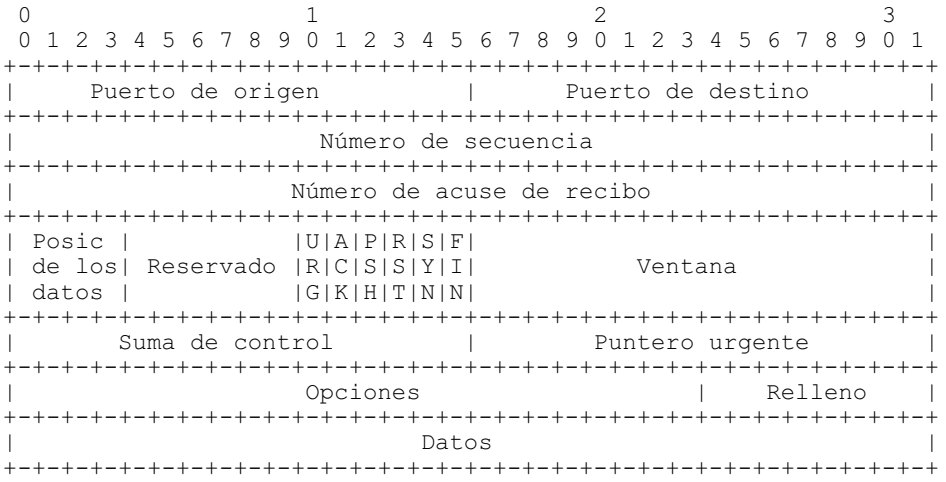
Las implementaciones de TCP seguirán un principio general de robustez: sé conservador en lo que hagas, sé liberal en lo que aceptes de los demás.

3. Especificación Funcional

3.1. Formato de la cabecera

Los segmentos de TCP se envían como datagramas de internet. La cabecera del protocolo de internet transporta varios campos de información, entre los que se incluyen las direcciones de los 'host' de origen y de destino [2]. Una cabecera de TCP sigue a la cabecera de internet, aportando información específica del protocolo de TCP. Esta división permite la existencia de otros protocolos de la capa de 'host' distintos de TCP.

Formato de la cabecera de TCP



Formato de la cabecera de TCP

Nótese que cada marca horizontal representa un bit.

Figura 3.

Puerto de origen: 16 bits

El número del puerto de origen.

Puerto de destino:16 bits

El número del puerto de destino.

Número de secuencia: 32 bits

El número de secuencia del primer octeto de datos de este

segmento (excepto cuando el indicador SYN esté puesto a uno). Si SYN está puesto a uno es el número de secuencia original (ISN: 'initial sequence number') y, entonces, el primer octeto de datos es ISN+1.

Número de acuse de recibo: 32 bits

Si el bit de control ACK está puesto a uno, este campo contiene el valor del siguiente número de secuencia que el emisor del segmento espera recibir. Una vez que una conexión queda establecida, este número se envía siempre.

Posición de los datos: 4 bits

El número de palabras de 32 bits que ocupa la cabecera de TCP. Este número indica dónde comienzan los datos. La cabecera de TCP (incluso una que lleve opciones) es siempre un número entero de palabras de 32 bits.

Reservado: 6 bits

Reservado para uso futuro. Debe valer 0.

Bits de control: 6 bits (de izquierda a derecha):

URG:	Hace significativo el campo "Puntero urgente"
ACK:	Hace significativo el campo "Número de acuse de recibo"
PSH:	Función de "Entregar datos inmediatamente" ('push')
RST:	Reiniciar ('Reset') la conexión
SYN:	Sincronizar ('Synchronize') los números de secuencia
FIN:	Últimos datos del emisor

Ventana: 16 bits

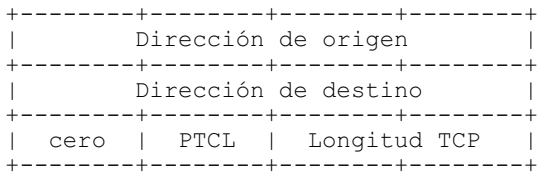
El número de octetos de datos, a contar a partir del número indicado en el campo de "Número de acuse de recibo", que el emisor de este segmento está dispuesto a aceptar.

Suma de control: 16 bits

El campo "Suma de control" es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera y del texto. Si un segmento contiene un número impar de octetos de cabecera y texto, el

último octeto se rellena con ceros a la derecha para formar una palabra de 16 bits con el propósito de calcular la suma de control. En el cálculo de la suma de control, el propio campo suma de control se considera formado por ceros.

La suma de control también incluye una pseudocabecera de 96 bits prefijada imaginariamente a la cabecera TCP. Esta pseudocabecera contiene la dirección de origen, la dirección de destino, el protocolo, y la longitud del segmento de TCP. Esto proporciona una protección ante segmentos mal encaminados. Esta información es transportada por el protocolo de internet y es transferida a través de la interfaz TCP/Red en los argumentos o en los resultados de las llamadas de TCP a IP.



La "longitud TCP" consiste en la suma de la longitud de la cabecera de TCP más la de los datos en octetos (esto no es una cantidad transmitida explícitamente, sino que ha de calcularse), y no incluye los 12 octetos de la pseudo cabecera.

Puntero urgente: 16 bits.

Este campo indica el valor actual del puntero urgente como un desplazamiento positivo desde el número de secuencia de este segmento. El puntero urgente apunta al número de secuencia del octeto al que seguirán los datos urgentes. Este campo es interpretado únicamente si el bit de control URG está establecido a uno.

Opciones: variable

Los campos de opciones pueden ocupar un cierto espacio al final de la cabecera de TCP, pero siempre de una longitud múltiplo de 8 bits.

En el cálculo de la suma de control, se incluyen todas las opciones. Una opción puede empezar en cualquier posición múltiplo de ocho.

Existen dos posibilidades para el formato de una opción:

Caso 1: Un octeto único con el tipo de opción.

Caso 2: Un octeto con el tipo de opción, un octeto con la longitud de la opción, y los octetos con los datos propiamente dichos de la opción.

La longitud de la opción tiene en cuenta tanto el octeto con el tipo de opción como el propio octeto de longitud así como los octetos con los datos de la opción.

Nótese que la lista de opciones puede ser más corta que lo que el campo "Posición de los datos" podría implicar. El contenido de la cabecera más allá de la opción "Fin de la lista de opciones" debe ser un relleno de cabecera (es decir, ceros).

Un módulo de TCP debe implementar todas las opciones.

Las opciones definidas en la actualidad incluyen (donde el tipo se indica en octal):

Tipo	Longitud	Significado
0	-	Fin de la lista de opciones.
1	-	Sin operación.
2	4	Tamaño máximo de segmento.

Definiciones de opciones específicas

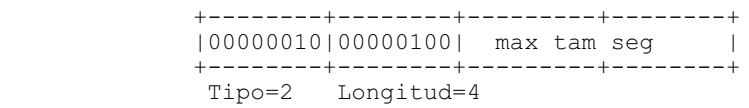
Fin de la lista de opciones

Este código de opción indica el final de la lista de opciones. Éste podría no coincidir con el final de la cabecera de TCP deducida a partir del campo "Posición de los datos". Esta opción se utiliza al final de todas las opciones, no al final de cada opción, y sólo es necesario utilizarla si el final de las opciones restantes no coincide con el final de la cabecera de TCP.

Sin operación

Este código de opción puede ser utilizado entre opciones, por ejemplo, para alinear el comienzo de una opción subsiguiente con el comienzo de una palabra. No se garantiza que los emisores vayan a utilizar esta opción, por lo que los receptores deben estar preparados para procesar todas las opciones, incluso si no comienzan al principio de una palabra.

Máximo tamaño de segmento



Datos de la opción "Máximo tamaño de segmento": 16 bits

Si esta opción está presente, entonces indica el tamaño máximo de segmento que puede recibir el módulo de TCP que envía este segmento. Este campo debe enviarse únicamente en la petición inicial de conexión (i.e., en los segmentos con el bit de control SYN puesto a uno). Si no se utiliza esta opción, se permite cualquier tamaño de segmento.

Relleno: variable

El relleno de la cabecera de TCP se utiliza para asegurar que la cabecera de TCP finaliza, y que los datos comienzan, en una posición múltiplo de 32 bits. El relleno está compuesto de ceros.

3.2. Terminología

Antes de empezar a discutir cualquier cosa sobre el modo de operación de TCP, es necesario introducir con cierto detalle algo de terminología. El mantenimiento de una conexión de TCP requiere el almacenamiento y seguimiento de varias variables. Estas variables han de imaginarse almacenadas en un registro de conexión denominado "Bloque de control de la transmisión"o TCB ('Transmission Control Block'). Entre las variables almacenadas en el TCB se hallan las direcciones de los conectores local y remoto, los valores de seguridad y prioridad de la conexión, los punteros a los búferes de envío y recepción del usuario, los

punteros a la cola de retransmisión y al segmento actual. También se almacenan en el TCB algunas variables relacionadas con los números de secuencia de envío y recepción.

Variables de la secuencia de envío

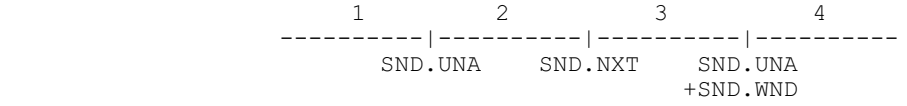
SND.UNA	envío sin acuse de recibo recibido ('unacknowledged')
SND.NXT	envío siguiente ('next')
SND.WND	ventana ('window') de envío
SND.UP	puntero urgente ('urgent pointer') de envío
SND.WL1	número de secuencia del segmento utilizado en la última ('last') actualización de la ventana
SND.WL2	número de acuse de recibo del segmento utilizado en la última actualización de la ventana
ISS	número de secuencia de envío inicial ('initial send sequence')

Variables de la secuencia de recepción

RCV.NXT	siguiente recepción
RCV.WND	ventana de recepción
RCV.UP	puntero urgente de recepción
IRS	número de secuencia de recepción inicial ('initial receive sequence')

Los siguientes diagramas pueden ayudar a relacionar alguna de estas variables con el espacio de secuencias.

Espacio de secuencias de envío



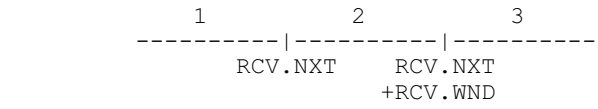
1 - anteriores números de secuencia de los que ya se ha recibido acuse de recibo
2 - número de secuencia de datos sin acuse de recibo recibido
3 - número de secuencia permitido en la siguiente transmisión de datos
4 - futuros números de secuencia no permitidos todavía en la siguiente transmisión

Espacio de secuencias de envío

Figura 4.

La ventana de envío es la porción del espacio de números de secuencia etiquetada como 3 en la figura 4.

Espacio de secuencias de recepción



1 - anteriores números de secuencia de los que ya se han envíado el acuse de recibo
2 - números de secuencia permitidos para una nueva recepción
3 - futuros números de secuencia que todavía no están permitidos

Espacio de secuencias de Recepción

Figura 5.

La ventana de recepción es la porción del espacio de secuencias etiquetada como 2 en la figura 5.

Además hay algunas variables que se utilizarán con frecuencia en la exposición que sigue más adelante y que toman sus valores de los campos del segmento actualmente en proceso.

Variables del segmento actual

SEG.SEQ	número de secuencia del segmento
SEG.ACK	número de acuse de recibo del segmento
SEG.LEN	longitud ('length') del segmento
SEG.WND	ventana del segmento
SEG.UP	puntero urgente del segmento
SEG.PRC	valor de prioridad del segmento

Una conexión progresa de acuerdo con una serie de estados durante su tiempo de vida. Los estados son: 'LISTEN' (en escucha), 'SYN-SENT' (SYN enviado), 'SYN-RECEIVED' (SYN recibido), 'ESTABLISHED' (establecida), 'FIN-WAIT-1' ("en espera de fin-1"), 'FIN-WAIT-2' ("en espera de fin-2"), 'CLOSE-WAIT' (en espera de cierre), 'CLOSING' (cerrándose), 'LAST-ACK' (último acuse de recibo), 'TIME-WAIT' (en espera), y el estado ficticio 'CLOSED' (cerrada).'CLOSED' es un estado ficticio porque representa el estado en el que no existe TCB, y por lo tanto, no hay conexión. De forma breve, los significados de los estados son:

LISTEN representa la espera de una solicitud de conexión

proveniente de cualquier TCP y puerto remotos.

SYN-SENT representa la espera de una solicitud de conexión concordante tras haber enviado previamente una solicitud de conexión.

SYN-RECEIVED representa la espera del acuse de recibo confirmando la solicitud de conexión tras haber recibido tanto como enviado una solicitud de conexión.

ESTABLISHED representa una conexión abierta, los datos recibidos pueden ser entregados al usuario. El estado normal para la fase de transferencia de una conexión.

FIN-WAIT-1 representa la espera de una solicitud de finalización de la conexión proveniente del TCP remoto, o del acuse de recibo de la solicitud de finalización previamente enviada.

FIN-WAIT-2 representa la espera de una solicitud de finalización del TCP remoto.

CLOSE-WAIT representa la espera de una solicitud de finalización de la conexión proveniente del usuario local.

CLOSING representa la espera del paquete, proveniente del TCP remoto, con el acuse de recibo de la solicitud de finalización.

LAST-ACK representa la espera del acuse de recibo de la solicitud de finalización de la conexión previamente enviada al TCP remoto (lo que incluye el haber enviado el acuse de recibo de la solicitud remota de finalización de la conexión).

TIME-WAIT representa la espera durante suficiente tiempo para asegurar que el TCP remoto recibió el acuse de recibo de su solicitud de finalización de la conexión.

CLOSED representa un estado sin conexión en absoluto

Una conexión de TCP progresa de un estado a otro en respuesta a eventos. Los eventos son: las llamadas de usuario,

OPEN ("abrir"), SEND ("enviar"), RECEIVE ("recibir"), CLOSE ("cerrar"), ABORT ("interrumpir"), and STATUS ("mostrar el estado"); la llegada de segmentos, particularmente aquellos que contengan alguno de los indicadores SYN, ACK, RST y FIN, y la expiración de plazos de tiempos.

El diagrama de estados de la figura 6 ilustra sólo los cambios de estados, junto con los eventos causantes y las acciones resultantes, pero no aborda ni las condiciones de error ni las acciones que no estén relacionadas con cambios de estado. En una sección posterior, se ofrecerá más detalle sobre todo lo relacionado con la reacción de TCP ante eventos.

NOTA BENE: este diagrama únicamente es un resumen y no debe ser tomado como la especificación total

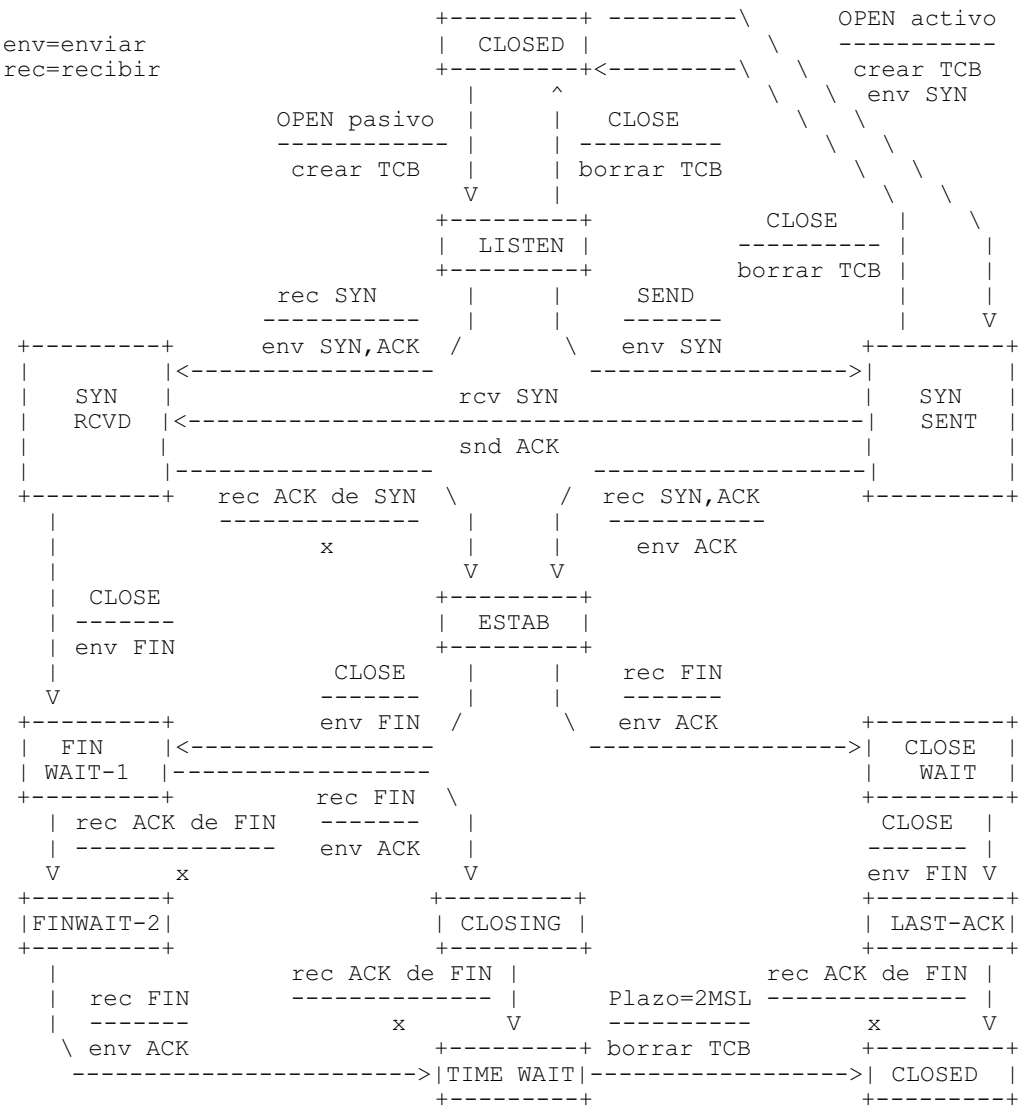


Diagrama de estados de una conexión de TCP
Figura 6.

3.3. Números de secuencia

El hecho de que todo octeto de datos enviado por una conexión de TCP tenga asociado un número de secuencia constituye una noción fundamental en el diseño de TCP. Ya que cada octeto se secuencia, puede realizarse un acuse de recibo de cada uno de ellos. El mecanismo de acuse de recibo empleado es acumulativo, de tal forma que el acuse de recibo de un número de secuencia X indica que todos los octetos hasta, pero no

incluyendo, X, han sido recibidos. Este mecanismo permite la detección fácil y directa de duplicados generados por retransmisiones. La numeración de octetos dentro de un segmento es de la siguiente forma: el primer octeto de datos inmediatamente tras la cabecera es el de menor número y los octetos siguientes son numerados consecutivamente.

Es esencial tener presente que el espacio real de números de secuencia es finito, aunque muy grande. Este espacio abarca desde el 0 hasta 2**32 -1. Como el espacio es finito, toda la aritmética con números de secuencia debe ser desarrollada módulo 2**32. Esta aritmética sin signo preserva la relación entre números de secuencia incluso cuando se rota desde 2**32 -1 a 0 de nuevo. Hay algunas sutilezas en los cálculos de la aritmética de módulos, por tanto se ha de tener un especial cuidado a la hora de programar las funciones de comparación de valores. El símbolo "=<" significa "menor o igual" (modulo 2**32).

Las típicas comparaciones de números de secuencia que TCP debe realizar incluyen:

(a) Determinar si un acuse de recibo se refiere a algún número de secuencia enviado pero del que no se ha recibido todavía su acuse de recibo

(b) Determinar si se ha recibido el acuse de recibo de todos los números de secuencia ocupados por un segmento (por ejemplo, para eliminar un segmento de la cola de retransmisión).

© Determinar si un segmento entrante contiene números de secuencia esperados (es decir, si el segmento cabrá en la ventana de recepción).

Como respuesta a sus envíos de datos, el módulo de TCP recibirá acuses de recibo. Es necesario realizar las siguientes comparaciones para procesar los acuses de recibo.

SND.UNA el menor número de secuencia sin acuse de recibo recibido

SND.NXT número de secuencia del próximo envío

SEG.ACK acuse de recibo procedente del TCP receptor (próximo número de secuencia esperado por el TCP

receptor)

SEG.SEQ primer número de secuencia de un segmento

SEG.LEN el número de octetos ocupados por los datos en el segmento (incluyendo SYN y FIN)

SEG.SEQ+SEG.LEN-1 último número de secuencia de un segmento

Un nuevo acuse de recibo (denominado un "acuse de recibo aceptable"), es aquél para el cual la siguiente desigualdad es cierta:

SND.UNA < SEG.ACK =< SND.NXT

Un segmento ubicado en la cola de retransmisión queda completamente confirmado por un acuse de recibo si la suma de su número de secuencia inicial y su longitud es menor o igual que el valor del acuse de recibo indicado en el segmento entrante.

Es necesario realizar las siguientes comparaciones cuando se reciben datos.

RCV.NXT siguiente número de secuencia esperado de los segmentos entrantes, lo que define el borde izquierdo o inferior de la ventana de recepción

RCV.NXT+RCV.WND-1 último número de secuencia esperado en un segmento entrante, lo que define el borde derecho o superior de la ventana de recepción

SEG.SEQ primer número de secuencia ocupado por el segmento entrante

SEG.SEQ+SEG.LEN-1 último número de secuencia ocupado por el segmento entrante

Se considera que un segmento ocupa una porción válida del espacio de secuencias de recepción si

RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND

o si

RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

La primera parte de esta comprobación mira a ver si el comienzo del segmento cae dentro de la ventana, la segunda parte comprueba si el final del segmento cae dentro de la ventana; si el segmento pasa cualquiera de las dos partes de la comprobación, entonces es que contiene datos dentro de la ventana.

La realidad es algo más compleja que todo esto. Debido a la existencia de ventanas y segmentos de tamaño cero, tenemos cuatro posibilidades a considerar a la hora de aceptar un segmento entrante:

Longitud segmento	Ventana recep.	Comprobación
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	no aceptable
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND o RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Nótese que cuando el tamaño de la ventana de recepción es cero, no se debería aceptar ningún segmento excepto los segmentos ACK. Así, es perfectamente posible para un TCP mantener una ventana de recepción de tamaño nulo mientras esté transmitiendo datos y recibiendo segmentos ACK. Sin embargo, incluso cuando la ventana de recepción es cero, un módulo de TCP debe procesar los campos RST y URG de todos los segmentos entrantes.

Se ha aprovechado el esquema de numeración para proteger también cierta información de control. Esto se consigue incluyendo implícitamente algunos indicadores de control en el espacio de secuencias de tal forma que puedan ser retransmitidos y confirmados sin posibilidad de confusión (i.e., se considerará una y sólo una copia de la información de control). La información de control no se transporta físicamente en el espacio de datos del segmento. Consecuentemente, se deben adoptar reglas implícitas de asignación de números de secuencia para los indicadores de control. SYN y FIN son los únicos indicadores de control que requieren esta protección, y estos indicadores se utilizan únicamente para la apertura y cierre de la conexión. Para propósitos de numeración de secuencias, se considera que el SYN ocurre antes que el primer octeto de datos

reales del segmento en el que es transportado, mientras que el FIN se considera que ocurre después del último octeto de datos del segmento que lleva el FIN. La longitud del segmento (SEG.LEN) incluyen tanto los datos como el espacio de secuencias ocupado por los indicadores de control. Cuando un SYN está presente, entonces SEG.SEQ es el número de secuencia del SYN.

Selección del número de secuencia inicial

Este protocolo no pone ninguna restricción sobre el hecho de reutilizar una y otra vez la misma conexión. Una conexión se define por el par de conectores. Cualquier nueva instancia de una conexión será referida como una encarnación de la conexión. El problema que aparece aquí es: "¿cómo identifica TCP los segmentos duplicados de encarnaciones previas de la conexión?" Este problema resulta evidente si la conexión se abre y se cierra en una sucesión muy rápida, o si la conexión se corta acompañada de pérdidas de memoria y después se reestablece.

Para evitar una confusión, se debe evitar que los segmentos de una encarnación de una conexión utilicen los números de secuencia que todavía posiblemente estén siendo utilizados en la red por otra encarnación anterior. Se desea asegurar esto, incluso si un TCP se cuelga y pierde todo conocimiento de los números de secuencia que estaba empleando. Cuando se crean nuevas conexiones se utiliza un generador de números iniciales de secuencia (ISN, 'initial sequence number') que selecciona un nuevo ISN de 32 bits. El generador estará asociado a un reloj (posiblemente ficticio) de 32 bits, cuyo bit menos significativo se incrementa aproximadamente cada 4 microsegundos. Así, el ISN rota aproximadamente cada 4.55 horas. Como queda asumido que los segmentos no permanecerán más tiempo en la red que la "vida máxima del segmento" (MSL, 'Maximum Segment Lifetime') y que el MSL es menor que 4.55 horas, podemos por tanto razonablemente presuponer que los ISN serán únicos.

Para cada conexión existe un número de secuencia de envío y un número de secuencia de recepción. El número de secuencia de envío inicial (ISS, 'initial send sequence') lo elige el TCP emisor, y el número de secuencia de recepción inicial (IRS, 'initial receive sequence') se asume durante el procedimiento de establecimiento de la conexión.

Para que una conexión quede establecida o inicializada, los dos TCP deben sincronizarse entre sí mediante sus números de secuencia iniciales. Esto se consigue durante el establecimiento de la conexión mediante el intercambio de segmentos que transportan un bit de control denominado "SYN" (acrónimo de 'syncronize' o sincronizar en inglés) y los números de secuencia iniciales. A modo de abreviatura, los segmentos que transportan el bit SYN se denominan también "SYN". Por tanto, la solución requiere de un apropiado mecanismo para elegir un número de secuencia inicial y una forma de acuerdo ligeramente complicada para intercambiar los ISN.

La sincronización requiere que cada parte envíe su propio número de secuencia inicial y que reciba una confirmación de su llegada en la forma de un acuse de recibo de la otra parte. Cada parte debe también recibir el número de secuencia inicial de la otra parte y enviar un acuse de recibo como confirmación.

1) A --> B SYN mi número de secuencia es X
2) A <-- B ACK tu número de secuencia es X
3) A <-- B SYN mi número de secuencia es Y
4) A --> B ACK tu número de secuencia es Y

Como los pasos 2 y 3 pueden combinarse en un único mensaje, este procedimiento se denomina acuerdo en tres pasos (o en tres mensajes).

Es necesario un acuerdo en tres pasos porque los números de secuencia no están sujetos a un reloj global de la red, y los TCP puede que tengan diferentes mecanismos para elegir los ISN. El receptor del primer SYN no tiene forma de saber si el segmento era uno anterior retrasado o no, a menos que recuerde el último número de secuencia utilizado en la conexión (lo que no siempre es posible), y por tanto debe pedir al emisor que verifique este SYN. El acuerdo en tres pasos y las ventajas de un esquema basado en relojes se discuten en [3]. Saber cuándo permanecer en silencio

Para estar seguro de que un TCP no cree un segmento que transporte un número de secuencia que podría estar duplicado por la existencia de otro segmento anterior que todavía permanezca en la red, TCP debe permanecer en silencio durante el tiempo de vida máximo de un segmento (MSL, 'maximum segment lifetime') antes de que asigne cualquier número de secuencia tras arrancar o recuperarse de una caída en la que se perdieron los números de secuencia en uso. En esta

especificación, el MSL se toma como de 2 minutos. Esta es la elección de un ingeniero, y puede ser cambiada si la experiencia indica que es conveniente hacerlo. Nótese que si un TCP se reinicializa de alguna forma en la que todavía retenga en memoria los números de secuencia en uso, entonces no es necesaria ninguna espera en absoluto; en ese caso, basta con utilizar números de secuencia mayores que los utilizados con anterioridad.

El concepto de "tiempo en silencio" en TCP

Esta especificación establece que los 'hosts' que se "cuelguen" sin haber retenido ningún conocimiento de los últimos números de secuencia transmitidos en cada conexión activa (i.e., sin cerrar) deberían retrasar la emisión de cualquier segmento de TCP durante al menos el "tiempo de vida máximo de segmento" (MSL) acordado. Más abajo, se da una explicación para esta especificación. Los fabricantes de TCP pueden violar esta restricción del "tiempo en silencio", pero sólo asumiendo el riesgo de causar que algunos datos viejos sean aceptados como nuevos o que los nuevos datos sean rechazados como duplicados de otros anteriores por algunos receptores en el entorno de internet.

Los TCP van consumiendo el espacio de números de secuencia según van formando segmentos e introduciéndolos en la cola de salida de red en un 'host' de origen. La detección de duplicados y el algoritmo de secuenciación del protocolo de TCP se apoya en la asociación única de datos del segmento a un espacio de secuencias hasta el punto de que los números de secuencia no pasarán por todos los 2**32 valores antes de que los datos del segmento asociados a aquellos números de secuencia hayan sido entregados y confirmados en su entrada por el receptor y que todas las copias duplicadas hayan sido "drenadas" del entorno internet. Si no se asumiera esto, dos segmentos TCP distintos muy bien podrían tener asignados los mismos, o coincidentes en parte, números de secuencia, causando confusión al receptor sobre cuáles son los datos viejos y cuáles los nuevos.

Recuérdese que cada segmento se asocia con tantos números de secuencia como octetos tenga el segmento.

Bajo condiciones normales, los TCP llevan la cuenta del siguiente número de secuencia para emitir y del último acuse de

recibo que se espera para evitar utilizar otra vez de forma equivocada un número de secuencia antes de que la entrega de su primer uso haya sido confirmada. Esto por sí solo no garantiza que los datos duplicados antiguos desaparezcan de internet, por ello el espacio de secuencia se ha definido muy grande para reducir la probabilidad de que un duplicado errante pueda causar problemas en su llegada. A 2 megabits/seg. lleva 4.5 horas el utilizar todos los 2**32 octetos del espacio de secuencias. Como la vida máxima de un segmento en la red probablemente no supere unas pocas decenas de segundos, se piensa que esto es suficiente protección para futuras redes, incluso si las transferencias de datos escalan hasta varias decenas de megabits/seg. A 100 megabits/seg, el tiempo de una vuelta es de 5,4 minutos, lo que puede ser un poco corto, pero todavía dentro de lo razonable.

El algoritmo básico de detección de duplicados y de secuenciación de TCP puede confundirse, sin embargo, si un TCP de origen no mantiene ninguna memoria de los números de secuencia que utilizó en su última conexión. Por ejemplo, si el módulo TCP comenzara todas las conexiones con el número de secuencia 0, entonces después de una caída y un reinicio, un TCP podría volver a formar una conexión anterior (posiblemente después de una resolución de la conexión medio abierta) y emitir paquetes con números de secuencia idénticos o coincidentes en parte con los paquetes que todavía circulan por la red y que fueron emitidos en la anterior encarnación de la misma conexión. En ausencia de cualquier conocimiento de los números de secuencia empleados en una conexión concreta, la especificación de TCP recomienda que el origen retrase la emisión de cualquier segmento en la conexión durante MSL segundos, para dejar tiempo suficiente a los segmentos de la anterior encarnación de la conexión a que desaparezcan del sistema.

Incluso los 'hosts' que puedan recordar la hora del día y utilizarla para seleccionar los valores de los números de secuencia no son inmunes a este problema (es decir, incluso si la hora del día se utilizara para seleccionar un número de secuencia inicial en cada nueva encarnación de la conexión).

Supóngase, por ejemplo, que se abre una conexión con un número de secuencia S. Supóngase que no se utiliza mucho esta conexión y que eventualmente la función que da el número de secuencia inicial a partir de la hora (ISN(t)) toma un valor igual al número de secuencia, por ejemplo S1, del último segmento enviado por este TCP en una conexión concreta. Ahora

supóngase que, en este instante, el 'host' se cuelga, se recupera y establece una nueva encarnación de la conexión. El número de secuencia inicial elegido es S1 = ISN(t) -- ¡ el último número de secuencia utilizado en la anterior encarnación de la conexión ! Si la recuperación sucede de forma suficientemente rápida, cualquier anterior duplicado en la red que transportan números de secuencia próximos a S1 pueden llegar y ser tratados como paquetes nuevos por el receptor de esta nueva encarnación de la conexión.

El problema consiste en que el 'host' que se recupera puede que no sepa por cuánto tiempo estuvo caído y si todavía quedan viejos duplicados en el sistema pertenecientes a anteriores encarnaciones de la conexión.

Una forma de tratar este problema consiste en retrasar de forma deliberada la emisión de segmentos durante un MSL tras recuperarse de una caída - ésta es la especificación del "tiempo en silencio". Los 'hosts' que prefieran evitar esperar están exponiéndose al riesgo de una posible confusión entre viejos y nuevos paquetes. Los fabricantes pueden proporcionar a los usuarios de TCP la posibilidad de seleccionar conexión por conexión si se debe esperar tras una caída o no, o puede que implementen informalmente el "tiempo en silencio" para todas las conexiones. Obviamente, incluso si el usuario elige esperar, esto no es necesario si el 'host' ha estado funcionando durante al menos MSL segundos.

Resumiendo: cada segmento emitido ocupa uno o más números de secuencia del espacio de secuencias, los números utilizados por un segmento están "ocupados" o "en uso" hasta que hayan pasado MSL segundos, tras una caída un cierto bloque de espacio-tiempo es ocupado por los octetos del último segmento emitido, si una nueva conexión empieza demasiado pronto y utiliza cualquiera de los números de secuencia de la huella dejada en el espacio-tiempo por el último segmento de la encarnación previa de la conexión, hay un peligro potencial de que haya una coincidencia en una cierta área de número, lo que podría causar confusión al receptor.

3.4. Establecemiento de una conexión

El procedimiento de acuerdo en tres pasos se utiliza para establecer una conexión. Normalmente, este procedimiento se

inicia por un TCP y responde otro TCP distinto. El procedimiento también funciona si dos TCP simultáneamente inician el procedimiento. En el caso de intentos simultáneos, cada TCP recibe un segmento 'SYN', que no lleva acuse de recibo, tras haber enviado su 'SYN'. Por supuesto, la llegada de un segmento 'SYN' anterior duplicado puede, potencialmente, hacer creer al receptor que está en progreso una iniciación simultánea de conexión. Un uso adecuado de los segmentos de tipo 'reset' puede eliminar la ambigüedad de estos casos.

A continuación, se dan algunos ejemplos de iniciación de conexión. Aunque estos ejemplos no muestren la sincronización de la conexión con segmentos que transporten datos, esto último es algo perfectamente legítimo, siempre y cuando el TCP receptor no entregue los datos al usuario hasta que no esté claro que los datos son válidos (es decir, los datos deben permanecer en los búferes del receptor hasta que la conexión alcance el estado ESTABLISHED o de conexión establecida). El acuerdo en tres pasos reduce la posibilidad de una conexión falsa. Es la implementación de la negociación entre la memoria y los mensajes quien debe proporcionar la información para esta comprobación.

El acuerdo en tres pasos más simple posible se muestra en la figura 7 de más abajo. Las figuras deben ser interpretadas de la siguiente manera. Cada línea se numera para referencias futuras. Las flechas hacia la derecha (-->) indican la salida de un segmento TCP desde el TCP A dirigido hacia el TCP B, o la llegada de un segmento en B proveniente de A. Las flechas hacia la izquierda (<--), indican lo contrario. Los puntos suspensivos (...) indican un segmento que todavía está en la red (retrasado). "XXX" indica un segmento que se perdió o fue rechazado. Los comentarios aparecen en paréntesis. Los estados de TCP representan el estado DESPUÉS de la salida o llegada del segmento (del que se muestra su contenido en el centro de cada línea). El contenido del segmento se muestra de forma abreviada, con el número de secuencia, los indicadores de control y el valor del campo de número de acuse de recibo (ACK).Otros campos tales como la ventana, direcciones, longitudes y el texto han sido omitidos en aras de la claridad.

```
    TCP A                                                TCP B

1.  CLOSED                                               LISTEN

2.  SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED

3.  ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED

4.  ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>       --> ESTABLISHED

5.  ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATOS>--> ESTABLISHED
```

Acuerdo en 3 pasos básico de sincronización de la conexión

Figura 7.

En la línea 2 de la figura 7, el TCP A comienza enviando un segmento SYN que además indica que va a utilizar números de secuencia comenzando por el número 100.En la línea 3, el TCP B envía un SYN confirmando la recepción del SYN que le envió el TCP A. Nótese que el campo de acuse de recibo indica que el TCP B está esperando recibir el 101 de la secuencia, confirmado la recepción del SYN que ocupó el lugar 100 de la secuencia.

En la línea 4, el TCP A responde con un segmento vacío conteniendo un ACK para el SYN del TCP B; y en la línea 5, el TCP A envía algunos datos. Nótese que el número de secuencia del segmento en la línea 5 es el mismo que el de la línea 4 porque el ACK no consume un número del espacio de secuencias (si se hiciera, ¡se estaría abocado a confirmar segmentos ACK!)

La iniciación simultánea es sólo un poco más compleja, como se muestra en la figura 8. Cada TCP pasa de CLOSED a SYN-SENT, luego a SYN-RECEIVED y por último a ESTABLISHED.

```
    TCP A                                                TCP B

1.  CLOSED                                               CLOSED

2.  SYN-SENT     --> <SEQ=100><CTL=SYN>              ...

3.  SYN-RECEIVED <-- <SEQ=300><CTL=SYN>              <-- SYN-SENT

4.               ... <SEQ=100><CTL=SYN>              --> SYN-RECEIVED

5.  SYN-RECEIVED --> <SEQ=100><ACK=301><CTL=SYN,ACK> ...

6.  ESTABLISHED  <-- <SEQ=300><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED

7.               ... <SEQ=101><ACK=301><CTL=ACK>     --> ESTABLISHED
```

Sincronización simultánea de la conexión

Figura 8.

El principal motivo del uso del acuerdo de tres pasos es

evitar que las iniciaciones de conexiones duplicadas anteriores causen confusión. Para tratar esto, se ha previsto un mensaje de control especial, el de reinicio o 'reset'. Si el TCP receptor está en un estado no sincronizado (i.e., SYN-SENT, SYN-RECEIVED), vuelve al estado LISTEN tras la recepción de un "reset" aceptable. Si el TCP está en uno de los estados sincronizados (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE- WAIT, CLOSING. LAST-ACK, TIME-WAIT), corta abruptamente la conexión e informa de ello a su usuario. Se discutirá este último caso en la sección dedicada a las conexiones "medio abiertas" ('half-open') de más adelante.

```
    TCP A                                                TCP B

1.  CLOSED                                               LISTEN

2.  SYN-SENT    --> <SEQ=100><CTL=SYN>               ...

3.  (duplicado) ... <SEQ=90><CTL=SYN>                --> SYN-RECEIVED

4.  SYN-SENT    <-- <SEQ=300><ACK=91><CTL=SYN,ACK>   <-- SYN-RECEIVED

5.  SYN-SENT    --> <SEQ=91><CTL=RST>                --> LISTEN

6.              ... <SEQ=100><CTL=SYN>               --> SYN-RECEIVED

7.  SYN-SENT    <-- <SEQ=400><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED

8.  ESTABLISHED --> <SEQ=101><ACK=401><CTL=ACK>      --> ESTABLISHED
```

Recuperación ante un SYN duplicado anterior

Figura 9.

Como ejemplo simple de una recuperación ante la presencia de duplicados anteriores, considérese la figura 9. En la línea 3, un SYN duplicado anterior llega al TCP B. El TCP B no puede discernir si se trata de un duplicado anterior, y por tanto responde de forma normal (línea 4). El TCP A detecta que el campo ACK es incorrecto y devuelve un RST ("reset") con su campo SEQ elegido de tal forma que el segmento sea creíble. El TCP B, al recibir el RST, vuelve al estado LISTEN. Cuando el SYN original (juego de palabras intencionado, 'original SYN' se pronuncia igual en inglés que "pecado original") llega finalmente al otro extremo en la línea 6, la sincronización procede de forma normal. Si el SYN de la línea 5 hubiera llegado antes que el RST, podría haberse dado un intercambio más complejo con RST enviados en ambas direcciones.

Conexiones "medio abiertas" y otras anomalías

Una conexión establecida se dice que está "medio abierta" ('half-open') si uno de los TCP ha cerrado o interrumpido la conexión en su extremo sin avisar a la otra parte, o si los dos extremos de la conexión han quedado desincronizados por culpa de una caída que causó pérdidas de memoria. Tales conexiones se reiniciarán automáticamente (mediante un "reset") si hay un intento de enviar datos en cualquier sentido. Sin embargo, es de esperar que las conexiones medio abiertas se den con poca frecuencia, y además el procedimiento de recuperación no es excesivamente complicado.

Si en el extremo A la conexión deja de existir, entonces un intento por parte del usuario en el extremo B de enviar datos sobre dicha conexión causará que el TCP receptor del extremo B reciba un mensaje de control de reinicio (un 'reset'). Este tipo de mensaje indica al TCP de B que algo va mal, y que se espera de él que interrumpa la conexión.

Asúmase que dos procesos de usuario A y B están comunicándose entre sí cuando una caída ocurre causando pérdidas de memoria en el TCP de A. Dependiendo del sistema operativo que dé soporte al TCP de A, es probable que exista algún mecanismo de recuperación ante errores. Cuando el TCP esté funcionando otra vez, es probable que A empiece de nuevo desde el comienzo o desde un punto de recuperación. Como resultado, A intentará probablemente abrir la conexión de nuevo mediante una llamada OPEN o intentará llamadas SEND sobre la conexión que A considera abierta. En el último caso, recibirá el mensaje de error "conexión no abierta" proveniente del TCP local (de A). En un intento de establecer la conexión, el TCP de A enviará un segmento conteniendo un SYN. Este escenario conduce al ejemplo mostrado en la figura 10. Después de que el TCP de A se cuelga, el usuario intenta reabrir la conexión. El TCP B, mientras tanto, piensa que la conexión sigue abierta.

```
     TCP de A                                              TCP de B

  1.  (CAÍDA)                              (enviar el 300, recibir el 100)

  2.  CLOSED                                               ESTABLISHED

  3.  SYN-SENT --> <SEQ=400><CTL=SYN>                 --> (??)

  4.  (!!)     <-- <SEQ=300><ACK=100><CTL=ACK>        <-- ESTABLISHED

  5.  SYN-SENT --> <SEQ=100><CTL=RST>                --> (¡¡Interrumpir!!)

  6.  SYN-SENT                                             CLOSED

  7.  SYN-SENT --> <SEQ=400><CTL=SYN>                 -->
```

Descubrimiento de una conexión "medio abierta"

Figura 10.

Cuando, en la línea 3, el SYN llega, el TCP de B, estando en un estado sincronizado y el segmento entrante siendo mayor que la ventana, responderá con un acuse de recibo indicando qué número de secuencia espera recibir a continuación (ACK 100). El TCP de A aprecia que este segmento no confirma nada enviado y, estando desincronizado, envía un 'reset' (RST) porque ha detectado que la conexión está medio-abierta. En la línea 5, el TCP de B interrumpe la conexión. EL TCP de A continuará intentando reestablecer la conexión; el problema queda reducido ahora al acuerdo en tres pasos básico descrito en la figura 7.

Un caso alternativo interesante se da si el TCP de A se cuelga y el TCP de B intenta enviar datos sobre lo que él piensa es una conexión sincronizada. Esto se ilustra en la figura 11.En este caso, los datos llegando al TCP de A provenientes del TCP de B (línea 2) no son aceptables porque no existe tal conexión, por tanto el TCP de A envía un RST. El RST es aceptable, por tanto el TCP B lo procesa e interrumpe la conexión.

```
       TCP de A                                                  TCP de B

  1.  (CAIDA)                                (enviar el 300,recibir el 100)

  2.  (??)    <-- <SEQ=300><ACK=100><DATA=10><CTL=ACK> <-- ESTABLISHED

   3.             --> <SEQ=100><CTL=RST>                              -->
(¡¡INTERRUMPIR!!)
```

Un extremo activo causa el descubrimiento de una conexión "medio abierta"

Figura 11.

En la figura 12, encontramos los TCP de A y de B ambos con conexiones pasivas esperando un SYN. Un duplicado anterior llega al TCP de B (línea 2) y provoca que B entre en acción. Le devuelve un SYN-ACK (línea 3) lo que causa que el TCP de A

genere un RST (pues el ACK de la línea 3 no es aceptable). El TCP de B acepta el "reset" y vuelve a su estado pasivo LISTEN.

```
     TCP A                                                TCP B

  1. LISTEN                                               LISTEN

  2.       ... <SEQ=Z><CTL=SYN>                  -->      SYN-RECEIVED

  3.  (??) <-- <SEQ=X><ACK=Z+1><CTL=SYN,ACK>     <--      SYN-RECEIVED

  4.       --> <SEQ=Z+1><CTL=RST>                -->      (-vuelve a LISTEN!)

  5.  LISTEN                                              LISTEN
```

Un SYN anterior duplicado inicia un 'reset' en dos conectores pasivos

Figura 12.

Otros muchos casos son posibles, estando todos ellos contemplados por las siguientes reglas de generación y proceso de segmentos RST.

Generación de reinicios ('resets')

Como regla general, se debe enviar un 'reset' (RST) siempre que llegue un segmento que aparentemente no esté destinado para la conexión actual. No debe enviarse un 'reset' si no está claro que éste sea el caso.

Hay tres grupos de estados:

1. Si la conexión no existe (estado CLOSED) entonces se envía un 'reset' como respuesta a cualquier segmento entrante excepto si es otro 'reset'. En particular, los SYN enviados a una conexión no existente serán rechazados de esta forma.

   Si en el segmento entrante tiene significado el campo ACK, el 'reset' escoge como su número de secuencia el valor del campo ACK del segmento, en otro caso el número de secuencia del 'reset' tiene valor cero y el campo ACK se establece a la suma del número de secuencia y la longitud del segmento entrante. La conexión permanece en el estado CLOSED.

2. Si la conexión está en cualquier estado no sincronizado (LISTEN, SYN-SENT, SYN-RECEIVED), y el segmento entrante confirma un número todavía no enviado (el segmento lleva un ACK inaceptable), o si un segmento entrante tiene un nivel de seguridad o

compartimentación que no concuerda exactamente con el solicitado para la conexión, entonces, se envía un 'reset'.

Si nuestro SYN no ha sido confirmado y el nivel de prioridad del segmento entrante es más alto que el nivel de prioridad solicitado entonces o se aumenta el nivel de prioridad local (si esto está permitido por el usuario y el sistema) o se envía un 'reset'; o si el nivel de prioridad del segmento entrante es menor que el nivel de prioridad solicitado entonces se continúa como si los niveles de prioridad hubieran concordado exactamente (si el TCP remoto no puede elevar el nivel de prioridad para concordar el nuestro sería detectado en el próximo segmento que enviara, y la conexión se finalizaría entonces). Si nuestro SYN ha sido confirmado (quizás en este mismo segmento entrante), el nivel de prioridad del segmento entrante debe concordar exactamente con el nivel de prioridad local, si no es así, se envía un 'reset'.

Si el segmento entrante tiene un campo ACK, el 'reset' escoge su número de secuencia del campo ACK del segmento, en otro caso el "reset" tiene como número de secuencia cero y el campo ACK se establece a la suma del número de secuencia y la longitud del segmento entrante. La conexión permanece en el mismo estado.

3. Si la conexión está en un estado sincronizado (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), cualquier segmento inaceptable (con un número de secuencia fuera de la ventana o con un número de acuse de recibo inaceptable) debe desencadenar únicamente el envío de un segmento de acuse de recibo vacío de datos, que llevaría el actual número de secuencia y un valor de acuse de recibo indicando el siguiente número de secuencia que se espera recibir; entonces la conexión permanece en el mismo estado.

   Si el segmento entrante tiene un nivel de seguridad, o compartimentación, o prioridad que no concuerda exactamente con los solicitados para la conexión, entonces se envía un 'reset' y la conexión pasa al estado CLOSED.El 'reset' escoge como su número de secuencia el valor del campo ACK del segmento entrante.

Procesamiento de 'resets'

En todos los estados exceptuando SYN-SENT, todos los segmentos de tipo 'reset' (RST) son validados comprobando sus campos SEQ. Un 'reset' es válido si su número de secuencia está dentro de la ventana. En el estado SYN-SENT (con un RST recibido en respuesta a un SYN inicial), el RST es aceptable si el campo ACK confirma el SYN.

El receptor de un RST primero lo comprueba, entonces cambia de estado. Si el receptor estaba en el estado LISTEN, lo ignora. Si el receptor estaba en el estado SYN-RECEIVED y estaba previamente en el estado LISTEN, entonces el receptor vuelve al estado LISTEN, en cualquier otro caso el receptor interrumpe la conexión y pasa al estado CLOSED. Si el receptor, estaba en cualquier otro estado, interrumple la conexión, avisa de ello al usuario y pasa al estado CLOSED.

3.5. Cierre de una conexión

CLOSE es una operación que significa "no tengo más datos que enviar". La noción de cerrar una conexión bidireccional tiene una interpretación ambigua, desde luego, dado que puede no resultar obvio como tratar a la parte receptora de la conexión. Se ha elegido tratar a CLOSE de una forma simple. El usuario que hace la llamada CLOSE puede continuar recibiendo mediante llamadas RECEIVE hasta que se le indique que el otro lado ha cerrado también la conexión mediante otro CLOSE. Por tanto, un programa puede iniciar varias llamadas SEND seguidas por una llamada CLOSE y continuar recibiendo (con RECEIVE) hasta que se le indique que falló una llamada RECEIVE debido a que su interlocutor cerró la conexión con un CLOSE. Se asume que TCP indicará al usuario que el otro lado de la conexión ha cerrado ésta incluso si no hay llamadas RECEIVE pendientes, de forma que pueda cerrar su conexión limpiamente. Una conexión TCP entregará de forma fiable todos los búferes enviados en llamadas SENT antes de que la conexión sea cerrada, de forma que un usuario que no espera datos de respuesta sólo necesite esperar a que la conexión sea cerrada con éxito para saber que todos sus datos fueron recibidos en el TCP de destino. Los usuarios deben mantenerse leyendo en las conexiones que cierren para envío hasta que TCP notifique que no hay más datos.

Existen fundamentalmente tres casos:

1) El usuario inicia el cierre de la conexión enviando la llamada CLOSE a TCP.

2) El TCP remoto inicia el cierre enviando una señal de control FIN.

3) Ambos usuarios realizan la llamada CLOSE simultáneamente.

Caso 1: el usuario local inicia el cierre

En este caso se puede construir un segmento FIN y ponerlo en la cola de salida de segmentos. TCP no aceptará más llamadas SEND y entrará en el estado FIN-WAIT-1. En este estado se permiten las llamadas RECEIVE. Todos los segmentos que preceden al FIN, incluido éste, serán retransmitidos hasta que se reciban los correspondientes acuses de recibo. Cuando el TCP remoto haya realizado el acuse de recibo del FIN y enviado su propio FIN, el TCP local puede realizar el acuse de recibo de este FIN. Nótese que un TCP que recibe un FIN enviará su ACK pero no enviará su propio FIN hasta que su usuario haya cerrado también la conexión con un CLOSE.

Caso 2: TCP recibe un FIN desde la red

Si llega un FIN no solicitado desde la red, el TCP receptor puede responder con un ACK y advertir al usuario de que la conexión se está cerrando. El usuario responderá con una llamada CLOSE, ante la cual TCP puede enviar un FIN al otro TCP tras enviar los datos restantes. Entonces TCP espera hasta el acuse de recibo de su propio FIN y elimina la conexión. Si el ACK no llega en el tiempo de espera del usuario, la conexión se aborta y así se notifica al usuario.

Caso 3: ambos usuarios cierran simultáneamente

El cierre simultáneo a ambos lados de la conexión causa el intercambio de segmentos FIN. Cuando todos los segmentos que preceden a los FIN se han procesado y confirmado con acuses de recibo, cada TCP puede responder con un ACK el FIN que ha recibido. Ambos, ante la recepción de los ACK, eliminarán la conexión.

```
    TCP A                                                TCP B

1.  ESTABLISHED                                          ESTABLISHED

2.  (Close)
    FIN-WAIT-1  --> <SEQ=100><ACK=300><CTL=FIN,ACK>  --> CLOSE-WAIT

3.  FIN-WAIT-2  <-- <SEQ=300><ACK=101><CTL=ACK>      <-- CLOSE-WAIT

4.                                                       (Close)
    TIME-WAIT   <-- <SEQ=300><ACK=101><CTL=FIN,ACK>  <-- LAST-ACK

5.  TIME-WAIT   --> <SEQ=101><ACK=301><CTL=ACK>      --> CLOSED

6.  (2 MSL)
    CLOSED
```

Secuencia de cierre normal

Figura 13.

```
    TCP A                                                TCP B

1.  ESTABLISHED                                          ESTABLISHED

2.  (Close)                                              (Close)
    FIN-WAIT-1  --> <SEQ=100><ACK=300><CTL=FIN,ACK>  ... FIN-WAIT-1
                <-- <SEQ=300><ACK=100><CTL=FIN,ACK>  <--
                ... <SEQ=100><ACK=300><CTL=FIN,ACK>  -->

3.  CLOSING     --> <SEQ=101><ACK=301><CTL=ACK>      ... CLOSING
                <-- <SEQ=301><ACK=101><CTL=ACK>      <--
                ... <SEQ=101><ACK=301><CTL=ACK>      -->

4.  TIME-WAIT                                            TIME-WAIT
    (2 MSL)                                              (2 MSL)
    CLOSED                                               CLOSED
```

Secuencia de cierre simultáneo

Figura 14.

3.6. Prioridad y seguridad

La intención es que la conexión sólo se permita entre puertos que operen exactamente con los mismos valores de seguridad y compartimentación y con un nivel de prioridad igual al mayor de los solicitados por ambos puertos.

Los parámetros de prioridad y seguridad usados en TCP son exactamente los definidos en el protocolo de internet (IP) [2]. A lo largo de esta especificación de TCP el término "seguridad/compartimentación" indica los parámetros de seguridad utilizados por IP que incluye los de seguridad, compartimentación, grupo de usuario y manejo de restricciones.

Un intento de conexión con valores de seguridad/compartimentación erróneos o con un nivel de

prioridad menor debe ser rechazado enviando un 'reset'. El rechazo de una conexión debido a un nivel de prioridad demasiado bajo ocurre solamente tras la recepción del acuse de recibo del SYN.

Nótese que los módulos de TCP que operan únicamente al nivel de prioridad por defecto deberán comprobar aún así la prioridad de los segmentos entrantes y posiblemente elevar el nivel de prioridad que usen en la conexión.

Los parámetros de seguridad pueden ser utilizados incluso en un entorno inseguro (los valores indicarían datos reservados secretos), por lo que las máquinas en entornos inseguros deben estar preparadas para recibir los parámetros de seguridad, aunque no es obligatorio que los envíen.

3.7. Comunicación de datos

Una vez establecida la conexión los datos se transmiten mediante el intercambio de segmentos. Dado que los segmentos pueden perderse debido a errores (fallo en la suma de comprobación) o congestión de la red, TCP utiliza la retransmisión (tras un tiempo de espera) para asegurar la entrega de cada segmento. Pueden llegar segmentos duplicados debido a la red o a la retransmisión de TCP. Tal y como se explica en la sección sobre números de secuencia, TCP realiza ciertas comprobaciones sobre los números de secuencia y de acuse de recibo en los segmentos para verificar su admisibilidad.

El emisor de los datos mantiene un registro del próximo número de secuencia a usar en la variable SND.NXT. El receptor de los datos mantiene un registro del siguiente número de secuencia esperado en la variable RCV.NXT. El emisor mantiene también un registro del número de secuencia sin acuse de recibo menor en la variable SND.UNA. Si el flujo de datos queda momentáneamente inactivo y de todos los datos enviados se tiene acuse de recibo entonces las tres variables serán idénticas.

Cuando el emisor crea un segmento y lo transmite, incrementa SND.NXT. Cuando el receptor acepta un segmento incrementa RCV.NXT y envía un acuse de recibo. Cuando el emisor de los datos recibe un acuse de recibo incrementa SND.UNA. La diferencia entre los valores de estas variables es una medida del retardo en la comunicación. El valor en el cual se

incrementan estas variables es el de la longitud de los datos del segmento. Nótese que, una vez en el estado ESTABLISHED todos los segmentos deben llevar la información del acuse de recibo actualizada.

La llamada de usuario CLOSE implica la función de entrega inmediata 'push', tal y como sucede con el indicador de control FIN en un segmento entrante.

Tiempo de espera de retransmisión

Debido a la variabilidad de las redes que componen un sistema de redes de internet y la gran cantidad de casos de conexiones TCP el tiempo de espera de retransmisión se debe determinar dinámicamente. Se ilustra a continuación un procedimiento para determinar un tiempo de espera de retransmisión.

Un ejemplo de procedimiento de tiempo de espera de retransmisión:

> Mídase el tiempo transcurrido entre el envío de un octeto de datos con un número de secuencia determinado y la recepción de un acuse de recibo que incluya ese número de secuencia (los segmentos enviados no tienen por qué concordar con los segmentos recibidos). Este tiempo medido es el "tiempo de ida y vuelta ('Round Trip Time' o RTT). Calcúlese después el "tiempo de ida y vuelta suavizado" ('Smoothed Round Trip Time' o SRTT) como:
>
> SRTT = (ALPHA * SRTT) + ((1-ALPHA) * RTT)
>
> y basándose en este, calcúlese el tiempo de espera de retransmisión (RTO) como:
>
> RTO = min[COTASUP,max[COTAINF,(BETA*SRTT)]]
>
> donde COTASUP es una cota superior del tiempo de espera (i.e., 1 minuto), COTAINF es una cota inferior del tiempo de espera (i.e., 1 segundo), ALPHA es un factor de suavizado (i.e., entre 0,8 y 0,9) y BETA es un factor de varianza del retardo (i.e., entre 1,3 y 2,0).

La comunicación de información urgente

El objetivo del mecanismo de urgencia de TCP consiste en permitir al usuario emisor la posibilidad de estimular al usuario receptor para que acepte ciertos datos urgentes y en permitir al TCP receptor que indique al usuario receptor cuándo se le han entregado todos los datos urgentes conocidos hasta el momento.

Este mecanismo permite elegir un punto en el flujo de datos como el final de la información urgente. Siempre que este punto esté más adelante del número de secuencia de recepción (RCV.NXT) en el TCP receptor, éste debe avisar al usuario para que cambie al "modo urgente"; cuando el número de secuencia de recepción alcance el puntero urgente, TCP debe avisar al usuario que cambie al "modo normal". Si el puntero urgente se actualiza mientras el usuario está en el "modo urgente", dicha actualización no será visible para el usuario.

El método emplea un campo urgente en todos los segmentos transmitidos. El indicador de control URG indica que el campo urgente tiene significado y debe ser añadido al número de secuencia de segmento para obtener el puntero urgente. La ausencia de este indicador denota la ausencia de datos urgentes.

Para enviar una indicación de urgencia el usuario debe enviar también al menos un octeto de datos. Si el usuario emisor indica también una llamada 'push', entonces se acelera la entrega de la información urgente al proceso de destino.

Manejo de la ventana

La ventana, que se envía en cada segmento, indica el rango de números de secuencia que el emisor de la ventana (el receptor de los datos) está preparado para aceptar en ese momento. Se supone que normalmente este rango está relacionado con el espacio de almacenamiento de datos disponible en ese momento para la conexión.

Indicar una ventana grande favorece las transmisiones. Si llegan más datos de los que pueden ser aceptados, serán descartados. Esto resultará en un exceso de retransmisiones, añadiendo una carga innecesaria a la red y a los módulos de TCP. Indicar una ventana pequeña puede restringir la transmisión de datos hasta el punto de introducir un retardo de ida y vuelta entre cada nuevo segmento transmitido.

Los mecanismos proporcionados permiten a un TCP anunciar una ventana grande y seguidamente anunciar una mucho menor sin tener que aceptar todos los datos. Esta práctica, conocida como "reducción la ventana", no se recomienda bajo ningún concepto. El principio de robustez dicta que cada módulo de TCP no reducirá la ventana por sí mismo, pero sí que deben estar preparados para esperar este comportamiento por parte de otros TCP.

El TCP emisor debe estar preparado para aceptar datos del usuario y enviar al menos un octeto de nuevos datos, incluso si el tamaño de la ventana de envío es cero. El TCP emisor debe retransmitir regularmente al receptor TCP incluso cuando el tamaño de la ventana es cero. Cuando el tamaño de la ventana es cero, se recomienda un intervalo de retransmisión de dos minutos. Esta retransmisión es esencial para garantizar que, siempre que uno de los TCP tenga una ventana de tamaño cero, la reapertura de la ventana sea notificada de forma fiable al otro.

Cuando el TCP receptor tiene una ventana de tamaño cero y llega un segmento debe todavía enviar un acuse de recibo con su próximo número de secuencia esperado y su tamaño de ventana actual (cero).

El TCP emisor empaqueta los datos para transmitir en segmentos que caben en la ventana actual y puede que reempaquete segmentos de la cola de retransmisión. Este reempaquetamiento no es obligatorio, pero puede resultar útil.

En una conexión con un flujo de datos de una sola dirección, la información de la ventana se transmite en segmentos de acuse de recibo, todos con el mismo número de secuencia, de manera que no hay forma de reordenarlos si llegan desordenados. Este no es un problema serio, pero permite que en ocasiones la información de ventana esté basada en información anticuada del receptor. Un refinamiento para evitar este problema consiste en escoger la información de ventana de los segmentos con el número de acuse de recibo más alto (es decir, los segmentos con número de acuse de recibo igual o mayor que el más alto recibido hasta el momento).

El procedimiento de gestión de ventana tiene una influencia significativa sobre el rendimiento de la comunicación. Los comentarios siguientes son sugerencias para los implementadores.

Sugerencias sobre la gestión de la ventana

Disponer de una ventana muy pequeña provoca que los datos sean transmitidos en muchos segmentos pequeños, cuando se puede conseguir un mejor rendimiento usando menos segmentos de mayor tamaño.

Una sugerencia para evitar las ventanas pequeñas es que el receptor retrase la actualización de una ventana hasta que el tamaño disponible sea al menos un X por ciento del tamaño máximo posible para la conexión (donde X podría estar entre 20 y 40).

Otra sugerencia consiste en que el emisor evite el envío de pequeños segmentos esperando hasta que la ventana sea lo bastante grande para enviar los datos. Si el usuario indica una función de entrega inmediata 'push' entonces los datos deberán ser enviados incluso si se trata de un segmento pequeño.

Nótese que los acuses de recibo no deben retrasarse o se producirán retransmisiones innecesarias. Una estrategia podría consistir en enviar un acuse de recibo cuando llega un segmento pequeño (sin actualizar la información de ventana), y luego enviar otro acuse de recibo con la nueva información de ventana cuando ésta sea mayor.

El segmento que se envía para sondear una ventana de tamaño cero puede también dar comienzo a un particionado de los datos transmitidos en segmentos más y más pequeños. Si un segmento que contiene un único octeto de datos enviado para sondear una ventana de tamaño cero se acepta, entonces se consume un octeto de la ventana ahora disponible. Si el TCP emisor simplemente envía todo lo que puede como cuando el tamaño de la ventana es distinto de cero, los datos transmitidos serán partidos en segmentos alternativamente grandes y pequeños. Conforme avance el tiempo, las pausas ocasionales en el receptor para permitir el ajuste de la ventana producirán una partición de cada segmento grande en uno pequeño y otro no tan grande como el original. Después de un rato la transmisión de datos se producirá en su mayor parte en segmentos pequeños.

La sugerencia aquí es que las implementaciones de TCP tienen que intentar, de forma activa, combinar las ventanas pequeñas en ventanas de mayor tamaño, dado que en las implementaciones simples, los mecanismos de gestión de

ventana tienden a producir muchas ventanas pequeñas.

3.8.Interfaces

Existen, por supuesto, dos interfaces de interés: la interfaz usuario/TCP y la interfaz TCP/nivel-inferior. Tenemos un modelo completamente elaborado de la interfaz usuario/TCP, pero dejamos sin especificar aquí la interfaz con el protocolo de nivel inferior, ya que estará especificado en detalle en la especificación del protocolo de nivel inferior. Para el caso en que el protocolo de nivel inferior sea IP, apuntamos algunos de los valores de los parámetros que los TCP pueden utilizar.

Interfaz usuario/TCP

La siguiente descripción funcional de las órdenes de usuario al módulo de TCP es, en el mejor de los casos, ficticia, dado que cada sistema operativo dispondrá de distintos recursos. En consecuencia, debemos advertir al lector que diferentes implementaciones de TCP pueden tener diferentes interfaces de usuario. Sin embargo, todos los TCP deben proporcionar un cierto conjunto mínimo de servicios para garantizar que todas las implementaciones TCP pueden soportar la misma jerarquía de protocolo. Esta sección especifica las interfaces funcionales obligatorias para todas las implementaciones de TCP.

Ordenes de usuario de TCP

Las siguientes secciones caracterizan de forma funcional una interfaz USUARIO/TCP. La notación utilizada es similar a la mayoría de las llamadas a procedimiento o función de los lenguajes de alto nivel, lo cual no significa que se excluyan las llamadas de servicio tipo 'trap' (i.e., SVC, UUO, EMT).

Las órdenes de usuario descritas más abajo especifican las funciones básicas que debe ejecutar TCP para soportar la comunicación entre procesos. Las implementaciones individuales deben definir su propio formato exacto, y pueden proporcionar combinaciones o subconjuntos de las funciones básicas en llamadas simples. En particular, algunas implementaciones pueden querer realizar la llamada de apertura OPEN de la conexión automáticamente en el primer SEND o RECEIVE

enviado por el usuario para una conexión dada.

Al proporcionar recursos para la comunicación entre procesos, el TCP no debe limitarse a aceptar órdenes, sino que también debe devolver información al proceso que sirve. Esta información consistirá en:

(a) información general acerca de una conexión (i.e., interrupciones, cierre remoto, enlace a un conector remoto sin especificar).

(b) respuestas a órdenes de usuario específicas indicando éxito o varios tipos de error.

Open

Formato: OPEN (puerto local, conector remoto, 'active'/'passive' [, tiempo de espera] [, prioridad] [, seguridad/compartimentación] [, opciones]) -> nombre de la conexión local

Se asume que el TCP local conoce la identidad de los procesos a los que sirve y que comprobará que el proceso tiene autoridad para utilizar la conexión especificada. Dependiendo de la implementación del TCP, los identificadores del TCP y de la red local serán proporcionados bien por el módulo de TCP o bien por el protocolo de nivel inferior (i.e. IP). Estas consideraciones derivan de cuestiones de seguridad, de forma que ningún TCP pueda hacerse pasar por otro, y así sucesivamente. De forma similar, ningún proceso puede hacerse pasar por otro sin la connivencia del módulo de TCP.

Si el indicador 'active'/'passive' está puesto a 'passive', entonces ésta será una llamada para escuchar (LISTEN) una conexión entrante. Un OPEN pasivo puede tener o bien un conector remoto completamente especificado para esperar una conexión particular o un conector remoto sin especificar para esperar cualquier llamada. Una llamada pasiva completamente especificada puede activarse mediante la ejecución subsecuente de un SEND.

Se crea un bloque de control de transmisión ('Transmission control block' o TCB) y se rellena parcialmente con los datos de los parámetros de la orden OPEN.

En una orden OPEN activa, el TCP iniciará el procedimiento para sincronizar (es decir, establecer) la conexión en una sola vez.

El tiempo de espera, si está presente, permite al llamador aplicar un tiempo de espera para todos los datos enviados por el TCP. Si los datos no se entregan con éxito en el destino dentro del tiempo de espera, TCP interrumpirá la conexión. El valor por defecto estándar en la actualidad es de 5 minutos.

TCP o algún componente del sistema operativo verificará la autorización de los usuarios para abrir una conexión con los valores de prioridad o seguridad/compartimentación especificados. La ausencia de valores de prioridad o seguridad/compartimentación en la llamada OPEN indica que se deben utilizar los valores por defecto.

TCP aceptará peticiones entrantes como válidas solamente si la información sobre seguridad/compartimentación es exactamente la misma y sólo si la prioridad es igual o mayor que la prioridad solicitada en la llamada OPEN.

La prioridad en la conexión es el mayor de los valores solicitados en la llamada OPEN y recibidos de la solicitud entrante, y fijado a este valor durante el resto de la conexión. Los implementadores pueden querer dar el control de la negociación de la prioridad al usuario. Por ejemplo, se podría permitir al usuario especificar que la prioridad sea exactamente la misma, o que cualquier intento de elevar la prioridad sea confirmado por el usuario.

TCP devolverá al usuario el nombre de la conexión local. Este nombre puede ser usado después como un atajo para denotar la conexión definida por el par <conector local, conector remoto>.

Send

Formato: SEND (nombre de la conexión local, dirección del búfer, contador de bytes, indicador PUSH, indicador URGENT [, tiempo de espera])

Esta llamada hace que se envíen mediante la conexión indicada los datos contenidos en el búfer de usuario indicado. Si la conexión no se ha abierto, el SEND se considera un error.

Puede que algunas implementaciones permitan a los usuarios enviar un SEND primero; en este caso, se efectuará un OPEN automático. Si el proceso llamador no está autorizado a usar esta conexión, se devuelve un error.

Si el indicador PUSH está activado, los datos deben ser transmitidos cuanto antes al receptor, y se activará el indicador PUSH en el último segmento TCP creado a partir del búfer. Si el indicador PUSH no está activado, los datos pueden ser combinados con datos de llamadas SEND subsiguientes en favor de la eficiencia de la transmisión.

Si el indicador URGENT está activado, los segmentos enviados al TCP de destino tendrán un valor en el campo de puntero urgente.

El TCP receptor señalará la condición de urgencia al proceso receptor si el puntero urgente indica que los datos que le preceden no han sido consumidos por el proceso receptor. El propósito de este indicador es estimular al receptor para que procese los datos urgentes e indicar al receptor cuándo se han recibido todos los datos urgentes actualmente conocidos. El número de veces que el TCP del usuario emisor señala una condición de urgencia no será necesariamente igual al número de veces que el usuario receptor será notificado de la presencia de datos urgentes.

Si no se ha especificado un conector remoto en la llamada OPEN, pero se establece la conexión (i.e., debido a que una conexión en estado de escucha LISTEN se ha hecho específica debido a la llegada de un segmento remoto al conector local), se envía el búfer indicado al conector remoto implícito. Los usuarios que hacen uso de la llamada OPEN sin especificar un conector remoto pueden usar la llamada SEND sin conocer explícitamente la dirección del conector remoto.

Sin embargo, si se intenta una llamada SEND antes de que el conector remoto se haya vuelto específico, se devolverá un error. Los usuarios pueden utilizar la llamada STATUS para determinar el estado de la conexión. En algunas implementaciones TCP puede notificar al usuario cuándo un conector sin especificar queda fijado.

Si se especifica un tiempo de espera, el tiempo de espera actual para esta conexión se cambia al nuevo valor.

En la implementación más simple, la llamada SEND no devolverá el control al usuario hasta que se complete la transmisión o se supere el tiempo de espera. Sin embargo, este método simple está sujeto a puntos muertos ('deadlocks') (por ejemplo, ambos lados de la conexión podrían intentar realizar operaciones SEND antes de hacer ninguna operación RECEIVE) y ofrece un rendimiento pobre, por lo cual no se recomienda. Una implementación más sofisticada devolverá el control inmediatamente para permitir al proceso ejecutarse concurrentemente con la E/S de red y, además, permitir que múltiples operaciones SEND tengan lugar a la vez. Las operaciones SEND múltiples son atendidas según van llegando, de forma que TCP pondrá en cola aquellas a las que no puede atender inmediatamente.

Hemos asumido implícitamente una interfaz de usuario asíncrona en la cual una llamada SEND provoca más tarde algún tipo de señal o pseudo-interrupción en el TCP servidor. Una posible alternativa es devolver una respuesta inmediatamente. Por ejemplo, las llamadas SEND podrían devolver un acuse de recibo local inmediato, incluso si el acuse de recibo del segmento enviado no ha sido aún recibido. Se puede asumir, de forma optimista, que la operación tendrá eventualmente éxito. Si no es así, la conexión se cerrará de todas formas debido a la superación del tiempo de espera. En implementaciones de este tipo (síncronas), existirán todavía algunas señales asíncronas, pero éstas estarán relacionadas con la propia conexión, no con segmentos o búferes específicos.

Con objeto de que el proceso distinga entre las distintas indicaciones de error o éxito correspondientes a las distintas llamadas SEND, puede resultar apropiado devolver la dirección del búfer junto con la respuesta codificada a la solicitud SEND. Las señales de TCP al usuario se discuten más abajo, indicando qué información debe ser devuelta al proceso llamador.

Receive

Formato: RECEIVE (nombre de la conexión local, dirección del búfer, número de bytes) -> número de bytes, indicador 'urgent', indicador 'push'.

Esta orden inicializa un búfer de recepción asociado con la conexión especificada. Si esta orden no viene precedida de una llamada OPEN o el proceso solicitante no está autorizado a usar

esta conexión, se devuelve un error.

En la implementación más simple, el control no volverá al programa solicitante hasta que el búfer se llene u ocurra algún error, pero este esquema tiene un alto riesgo de puntos muertos. Una implementación más sofisticada permitiría que varios RECEIVE se ejecutaran a la vez. Estos se irían completando según van llegando los segmentos. Esta estrategia permite incrementar el rendimiento a costa de un esquema más elaborado (posiblemente asíncrono) para notificar al programa solicitante que se ha detectado una llamada de entrega inmediata PUSH o se ha llenado un búfer.

Si llegan datos suficientes como para llenar el búfer antes de que se detecte el PUSH, el indicador PUSH no se activará en la respuesta al RECEIVE. El búfer será llenado con tantos datos como le quepan. Si se detecta un PUSH antes de que el búfer esté lleno, éste será devuelto parcialmente lleno y con el indicador PUSH activado.

Si hay datos urgentes el usuario habrá sido informado tan pronto como llegaron por medio de una señal de TCP al usuario. El usuario receptor debe entonces estar en 'modo urgente'. Si el indicador URGENT está activado, es que quedan todavía datos urgentes. Si está desactivado, esta llamada a RECEIVE ha devuelto todos los datos urgentes y el usuario puede ahora abandonar el "modo urgente". Nótese que los datos que siguen al puntero urgente (datos no urgentes) no pueden ser entregados al usuario en el mismo búfer con los datos urgentes precedentes a menos que el límite entre ellos esté claramente marcado para el usuario.

Para distinguir entre varias llamadas RECEIVE pendientes y para tener en cuenta el caso de un búfer no saturado, el código de retorno viene acompañado de un puntero al búfer y un número de bytes que indica la longitud de los datos recibidos.

En algunas implementaciones alternativas, el TCP se encargaría de reservar el espacio de almacenamiento para el búfer, o bien podría compartir un búfer circular con el usuario.

Close

Formato: CLOSE (nombre de la conexión local)

Esta orden cierra la conexión especificada. Si la conexión no está abierta o el proceso solicitante no está autorizado a usar dicha conexión, se devuelve un error. El cierre de conexiones está pensado para que sea una operación limpia en el sentido de que las llamadas SEND pendientes serán transmitidas (y retransmitidas), en la medida en que el control de flujo lo permita, hasta que todas hayan sido servidas. Por tanto, es aceptable enviar varias órdenes SEND seguidas de una llamada CLOSE, y suponer que todos los datos serán enviados a su destino. Debe quedar claro que los usuarios pueden continuar recibiendo por conexiones que se están cerrando, ya que el otro lado de la conexión puede estar intentando transmitir el resto de sus datos. Por tanto, cerrar una conexión significa "no tengo nada más que enviar" pero no "no recibiré nada más". Puede suceder (si el protocolo del nivel de usuario no está bien ideado) que el lado de la conexión que cierra no sea capaz de deshacerse de todos sus datos antes de que se cumpla el tiempo de espera. En este caso, la llamada CLOSE se convierte en una llamada ABORT, y el TCP que cierra la conexión deja de seguir.

El usuario puede cerrar la conexión en cualquier momento bajo su propia iniciativa, o en respuesta a varios avisos del TCP (i.e., ejecutado un cierre remoto, excedido el tiempo de espera de transmisión, destino inaccesible).

Dado que cerrar una conexión requiere la comunicación con el TCP remoto, las conexiones pueden permanecer en el estado de cierre durante un corto espacio de tiempo. Los intentos de reabrir la conexión antes de que el TCP responda a la orden CLOSE darán como resultado respuestas de error.

La orden CLOSE implica la función de entrega inmediata 'push'.

Status

Formato: STATUS (nombre de la conexión local) -> datos de estado

Esta orden de usuario depende de la implementación y puede excluírse sin efectos adversos. La información devuelta provendrá típicamente del TCB asociado con la conexión.

Esta orden devuelve un bloque de datos que contiene la siguiente información: conector local, conector remoto, nombre

de la conexión local, ventana de recepción, ventana de envío, estado de la conexión, número de búferes en espera del acuse de recibo, número de búferes pendientes de recepción, estado urgente, prioridad, seguridad/compartimentación, y tiempo de espera de transmisión.

Dependiendo del estado de la conexión, o de la implementación misma, parte de esta información puede no estar disponible o no tener ningún significado útil. Si el proceso solicitante no está autorizado a utilizar esta conexión se devuelve un error. Esto evita que los procesos no autorizados puedan obtener información sobre una conexión.

Abort

Formato: ABORT (nombre de la conexión local)

Esta orden aborta la ejecución de todas las órdenes SEND y RECEIVE pendientes, elimina el TCB y envía un mensaje especial RESET al otro lado de la conexión. Dependiendo de la implementación, los usuarios puede recibir avisos de abandono por cada orden SEND o RECEIVE pendiente, o simplemente recibir una confirmación de la ejecución de la orden ABORT.

Mensajes de TCP a usuario

Se da por supuesto que el sistema operativo proporciona algún medio para que TCP pueda enviar una señal asíncrona al programa de usuario. Cuando TCP envía una señal a un programa de usuario pasa cierta información al usuario. A menudo en la especificación la información será un mensaje de error. En otros casos habrá información relativa a la finalización del procesamiento de una orden SEND o RECEIVE o cualquier otra llamada del usuario.

Se proporciona la siguiente información:

Nombre de la conexión local	Siempre
Texto de respuesta	Siempre
Dirección del búfer	SEND y RECEIVE
Número de bytes (núm. de bytes recibidos)	RECEIVE
Indicador de entrega inmediata	RECEIVE
Indicador urgente	RECEIVE

Interfaz TCP/nivel inferior

Realmente, TCP realiza llamadas al módulo del protocolo de nivel inferior para enviar y recibir información a través de una red. Uno de estos casos es el del sistema de interconexión de redes ARPA, donde el módulo del nivel inferior es el protocolo de internet (IP) [2].

Si el protocolo de nivel inferior es IP, éste proporciona argumentos para un tipo de servicio y un tiempo de vida. TCP utiliza los siguientes valores para estos parámetros:

Tipo de servicio = prioridad: habitual,
retardo: normal,
rendimiento: normal,
fiabilidad: normal;
ó 00000000.

Tiempo de vida = un minuto;
ó 00111100.

Nótese que el tiempo de vida máximo asumido para un segmento es de dos minutos. Aquí se requiere explícitamente que un segmento sea destruido si no puede ser entregado por el sistema de internet en un minuto.

Si el nivel inferior es IP (u otro protocolo que proporcione esta característica) y se utiliza encaminamiento de origen ['source routing'], la interfaz debe permitir que se pueda comunicar la información sobre la ruta. Esto es especialmente importante de forma que las direcciones de origen y destino usadas en la suma de comprobación de TCP sean las direcciones de origen y destino definitivas. También es importante conservar la ruta de retorno para contestar preguntas sobre la conexión.

Todo protocolo de nivel inferior debe suministrar la dirección de origen, la dirección de destino, y los campos del protocolo, además de alguna forma de determinar la "longitud de TCP", para suministrar un servicio equivalente al de IP y poder ser utilizados en la suma de comprobación de TCP.

3.9. Procesamiento de eventos

El procesamiento explicado en esta sección es un ejemplo

de una posible implementación. Otras implementaciones pueden tener secuencias de procesamiento ligeramente diferentes, pero sólo en los detalles, no en lo esencial.

Se puede caracterizar la actividad del TCP como una serie de respuestas a eventos. Los eventos posibles se pueden clasificar en tres categorías: llamadas del usuario, segmentos entrantes y fin de tiempos de espera. Esta sección describe el procesamiento que realiza TCP en respuesta a cada uno de estos eventos. En muchos casos el procesamiento necesario depende del estado de la conexión

Eventos posibles

Llamadas del usuario

OPEN
SEND
RECEIVE
CLOSE
ABORT
STATUS

Segmentos entrantes

LLEGADA DEL SEGMENTO

Fin del tiempo de espera

FIN DEL TIEMPO DE ESPERA DE USUARIO
FIN DEL TIEMPO DE ESPERA DE RETRANSMISION
FIN DEL TIEMPO DE ESPERA EN EL ESTADO 'TIME-WAIT'

El modelo de la interfaz TCP/usuario se basa en que las órdenes de usuario reciben una respuesta inmediata y posiblemente otra retardada por medio de un evento o una pseudo-interrupción. En las descripciones siguientes, el término "señal" significa "causa una respuesta retardada".

Las respuestas de error se dan en forma de cadenas de caracteres. Por ejemplo, las órdenes de usuario que hacen referencia a conexiones que no existen reciben lo siguiente: 'error: connection not open' ("error: conexión sin abrir").

Téngase en cuenta en lo que sigue que toda la aritmética sobre números de secuencia, números de acuse de recibo,

números asociados a ventanas, etcétera, es módulo 2**32, el tamaño del espacio reservado para los números de secuencia. Nótese también que "=<" significa "menor o igual que (módulo 2**32)".

Una forma natural de pensar en el procesamiento de segmentos entrantes es imaginar que primero se comprueba que su número de secuencia sea el adecuado (es decir, que su contenido caiga dentro del rango de la "ventana de recepción" esperada en el espacio de números de secuencia) y que después generalmente son puestos en cola y procesados según el orden por números de secuencia.

Cuando un segmento se solapa con otros segmentos ya recibidos se reconstruye el segmento de forma que contenga únicamente los nuevos datos, y se ajustan los campos de cabecera para que sean consistentes.

Nótese que si no se menciona un cambio de estado TCP permanece en el mismo estado.

Llamada OPEN

ESTADO 'CLOSED' (i.e., el TCB no está definido)

Se crea un nuevo bloque de control de transmisión (TCB) que soporta información sobre el estado de la conexión. Se completa el identificador de conector local e información relativa al conector remoto, prioridad, seguridad/compartimentación y tiempo de espera de usuario. Nótese que algunos detalles del conector remoto pueden quedar sin especificar durante un OPEN pasivo y habrán de ser cumplimentados en los parámetros del segmento SYN entrante. Están permitidas para este usuario, tanto verificaciones de seguridad, como solicitudes de prioridad. De no ser así, se devuelve 'error: precedence not allowed' ("error: prioridad no admitida") o 'error: security/compartment not allowed' ("error: seguridad/compartimentación no admitidas"). En el caso pasivo, se pasa al estado LISTEN y se retorna. En el caso activo, y si el conector remoto queda sin especificar, se devuelve 'error: foreign socket unspecified' ("error: conector remoto sin especificar"). En el caso activo, y si se especifica el conector remoto, se envía un segmento SYN. Se selecciona un número de secuencia inicial (ISS). Se envía un segmento SYN de la forma <SEQ=ISS><CTL=SYN>. Se fija SND.UNA a ISS, SND.NXT a ISS+1, se pasa al estado SYN-SENT y se retorna.

Si el llamador no tiene acceso al conector local especificado, se devuelve 'error: connection illegal for this process' ("error: conexión ilegal para este proceso"). Si no quedase espacio para crear una nueva conexión, se devuelve 'error: insufficient resources' ("error: recursos insuficientes").

ESTADO 'LISTEN'

Si es el caso activo y para un conector remoto especificado, se cambia la conexión de pasiva a activa, se selecciona un ISS. Se envía un segmento SYN, se fija SND.UNA a ISS, SND.NXT a ISS+1. Se pasa al estado SYN-SENT. Los datos asociados a SEND pueden enviarse mediante un segmento SYN o pasarse a la cola de transmisión después de entrar en el estado ESTABLISHED. El bit urgente, en caso de ser requerido en el comando, deberá enviarse junto con los segmentos de datos enviados como consecuencia de la ejecución de este comando. Si no existiera espacio para almacenar la petición en cola, se responde con 'error: insufficient resources'. Si el conector remoto no hubiera sido especificado, se devolverá 'error: foreign socket unspecified'.

ESTADO 'SYN-SENT'
ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se devuelve 'error: connection already exists' ("error: la conexión ya existe").

Llamada SEND

ESTADO 'CLOSED' (i.e., TCB no existe)

Si el usuario no tuviera acceso a tal conexión, entonces se devuelve 'error: connection illegal for this process' ("error: conexión ilegal para este proceso).

En caso contrario, se devuelve 'error: connection does not

exist' ("error: la conexión no existe").

ESTADO 'LISTEN'

Si el conector remoto hubiera sido especificado, la conexión pasa de pasiva a activa, se selecciona un ISS. Se envía un segmento SYN, se fija SND.UNA a ISS, SND.NXT a ISS+1. Se pasa al estado SYN-SENT. Los datos asociados con SEND podrán ser enviados mediante un segmento SYN o almacenados en la cola de transmisión tras activarse el estado ESTABLISHED. El bit urgente, en caso de ser requerido en el comando, deberá enviarse junto con los segmentos de datos enviados como consecuencia de la ejecución de este comando. Si no existiera espacio para almacenar la petición en cola, se responde con 'error: insufficient resources'. Si el conector remoto no hubiera sido especificado, devolverá 'error: foreign socket unspecified'.

ESTADO 'SYN-SENT'
ESTADO 'SYN-RECEIVED'

Almacenamiento en cola de los datos para la transmisión tras haber pasado al estado ESTABLISHED. Si no hubiera espacio disponible en la cola, responde con 'error: insufficient resources'.

ESTADO 'ESTABLISHED'
ESTADO 'CLOSE-WAIT'

Segmentar el búfer y enviarlo cargado con un acuse de recibo (valor de acuse de recibo = RCV.NXT). Si no hubiera espacio suficiente para almacenar el contenido de este búfer, se devuelve simplemente "error: insufficient resources".

Si el indicador de urgente vale uno, se envía SND.UP <- SND.NXT-1 y se pone el puntero urgente en los segmentos salientes.

ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se devuelve 'error: connection closing' y no presta el servicio solicitado.

Llamada RECEIVE

ESTADO 'CLOSED' (i.e., no existe TCB)

Si el usuario no tuviera acceso a tal conexión, se devuelve "error: connection illegal for this process".

En caso contrario se devuelve "error: connection does not exist".

ESTADO 'LISTEN'
ESTADO 'SYN-SENT'
ESTADO 'SYN-RECEIVED'

Almacenamiento en la cola de procesado tras haber entrado en el estado ESTABLISHED. Si no hubiera espacio para almacenar en cola esta solicitud, se responde con 'error: insufficient resources'.

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Si hubiera una cantidad insuficiente de segmentos entrantes en cola para satisfacer la solicitud, se almacena en cola la solicitud. Si no hubiera espacio en cola para recordar el RECEIVE, se responde con "error: insufficient resources".

Reensamblado de los segmentos entrantes en cola en el búfer de recepción y devolución al usuario. Se marca 'push seen' ("'push' visto") (PUSH) si este fuera el caso.

Si se adelanta un RCV.UP a los datos que actualmente se están pasando al usuario se notifica al usuario sobre la presencia

de datos urgentes.

Cuando el TCP se responsabiliza de la entrega de datos al usuario, este hecho debe ser comunicado al emisor por medio de un acuse de recibo. La construcción de tal acuse de recibo se describe más abajo en la discusión del procesado de un segmento entrante.

ESTADO 'CLOSE-WAIT'

Dado que la parte remota ya ha enviado un FIN, los RECEIVE deberán ser rellenados con texto ya disponible, pero aún no entregado al usuario. Si no hubiera texto en espera para ser entregado, el RECEIVE obtendrá una respuesta 'error: connection closing' ("error: cerrando la conexión"). En caso contrario, cualquier texto presente podrá usarse para alimentar el RECEIVE.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se devuelve 'error: connection closing'.

Llamada CLOSE

ESTADO 'CLOSED' (i.e., no existe TCB)

Se devuelve 'error: connection illegal for this process'.

En caso contrario se devuelve 'error: connection does not exist'.

ESTADO 'LISTEN'

A cualesquiera órdenes RECEIVE relevantes, se devuelven respuestas "error: closing". Se borra el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'SYN-SENT'

Se borra el TCB y se devuelven respuestas "error: closing" a cualesquiera órdenes SEND o RECEIVE.

ESTADO 'SYN-RECEIVED'

Si no se hubieran emitido órdenes SEND y no hubiera datos pendientes de envío, se construye un segmento FIN y se envía, tras lo que se pasa al estado FIN-WAIT-1; en caso contrario, se almacena en la cola de procesamiento tras haber pasado al estado ESTABLISHED.

ESTADO 'ESTABLISHED'

Se deja en cola hasta que todas las órdenes SEND precedentes hayan sido segmentadas, entonces se construye un segmento FIN y se envía. En cualquier caso, se pasa al estado FIN-WAIT-1.

ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Estrictamente hablando, se trata éste de un error, y debería recibir una respuesta 'error: connection closing'. Una respuesta 'ok' ("correcto") sería aceptable también, ya que no se ha emitido aún un segundo FIN (no obstante, el primer FIN puede ser retransmitido).

ESTADO 'CLOSE-WAIT'

Esta solicitud se almacena en cola hasta que todas las órdenes SEND precedentes hayan sido segmentadas; entonces se envía un segmento FIN, y se pasa al estado CLOSING.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se contesta con 'error: connection closing'.

Llamada ABORT

ESTADO 'CLOSED' (i.e., no existe TCB)

Si el usuario no tuviera acceso a tal conexión, se devuelve 'error: connection illegal for this process'.

En caso contrario se devuelve 'error: connection does not exist'.

ESTADO 'LISTEN'

A cualesquiera órdenes RECEIVE relevantes deben devolverse respuestas 'error: connection reset' ("error: conexión reiniciada"). Se borra el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'SYN-SENT'

Todas las órdenes SEND y RECEIVE en cola deberían recibir la notificación 'connection reset'. Borrado de TCB, paso al estado CLOSED, y retorno.

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'

Envío de un segmento de reinicio ('reset'):

<SEQ=SND.NXT><CTL=RST>

Todas las órdenes SEND y RECEIVE en cola deberían recibir una notificación 'connection reset'; todos los segmentos en cola de transmisión (excepto el RST construido más arriba) o retransmisión deberían ser abandonados; se borra el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Se responde con 'ok' y se borra el TCB, se pasa al estado CLOSED, y se retorna.

Llamada STATUS

ESTADO 'CLOSED' (i.e., TCB no está definido)

En caso de que el usuario no consiguiera acceso a tal conexión, se devuelve 'error: connection illegal for this process'.

En cualquier otro caso, se devuelve 'error: connection does not exist'.

ESTADO 'LISTEN'

Se devuelve 'state = LISTEN', y el puntero de TCB.

ESTADO 'SYN-SENT'

Se devuelve 'state = SYN-SENT', y el puntero de TCB.

ESTADO 'SYN-RECEIVED'

Se devuelve 'state = SYN-RECEIVED', y el puntero de TCB.

ESTADO 'ESTABLISHED'

Se devuelve 'state = ESTABLISHED', y el puntero de TCB.

ESTADO 'FIN-WAIT-1'

Se devuelve 'state = FIN-WAIT-1', y el puntero de TCB.

ESTADO 'FIN-WAIT-2'

Se devuelve 'state = FIN-WAIT-2', y el puntero de TCB.

ESTADO 'CLOSE-WAIT'

Se devuelve 'state = CLOSE-WAIT', y el puntero de TCB.

ESTADO 'CLOSING'

Se devuelve 'state = CLOSING', y el puntero de TCB.

ESTADO 'LAST-ACK'

Se devuelve 'state = LAST-ACK', y el puntero de TCB.

ESTADO 'TIME-WAIT'

Se devuelve 'state = TIME-WAIT', y el puntero de TCB.

LLEGADA DE SEGMENTO ('SEGMENT ARRIVES')

Si el estado es CLOSED (i.e., no existe TCB) entonces se descartan todos los datos del segmento entrante. Cualquier segmento que contenga un RST es descartado. Cualquier segmento entrante que no contenga un RST origina el envío de un RST como respuesta. Se seleccionan los valores de los campos de acuse de recibo y de secuencia para que el número de secuencia del segmento de reinicio sea aceptable para el TCP que envió el segmento.

Si el bit de ACK vale cero, se utiliza el número de secuencia cero,

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Si el bit ACK vale uno,

<SEQ=SEG.ACK><CTL=RST>

Se retorna.

Si el estado es LISTEN, entonces:

Primero se comprueba si es un RST. Se deberá ignorar cualquier RST entrante. Se retorna.

Segundo se comprueba si es un ACK. Cualquier acuse de recibo será defectuoso si llega cuando una conexión está todavía en estado LISTEN. Debería construirse un segmento de reinicio aceptable para cualquier segmento entrante que transporte un ACK. El RST deberá tener el siguiente formato:

<SEQ=SEG.ACK><CTL=RST>

Se retorna.

Tercero se comprueba si es un SYN. Si el bit SYN vale 1, se comprueba la seguridad. Si la seguridad / compartimentación del segmento entrante no coincidiera con la seguridad / compartimentación del TCB, se envía un segmento de reinicio y se retorna.

<SEQ=SEG.ACK><CTL=RST>

Si SEG.PRC es mayor que TCB.PRC, y en caso de que el usuario y el sistema lo permita, se fija TCB.PRC<-SEG.PRC, si no estuviera permitido, se envía un reinicio y se retorna.

<SEQ=SEG.ACK><CTL=RST>

Si SEG.PRC es menor que TCB.PRC, se prosigue.

Se pone RCV.NXT a SEG.SEQ+1, IRS a SEG.SEQ y cualquier otro control o texto deberá quedar en la cola para su procesado posterior. Deberá seleccionarse ISS y enviarse un segmento SYN de la forma:

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

SND.NXT se fija a ISS+1 y SND.UNA a ISS. El estado de la conexión deberá cambiar a SYN-RECEIVED. Nótese que será procesado cualquier otro control entrante o datos (combinados con SYN) en el estado SYN-RECEIVED, pero el procesado de SYN y ACK no deberían repetirse. Si la escucha no hubiera sido completamente especificado (i.e., el conector remoto no hubiera sido completamente especificado), deberán cumplimentarse en este momento los campos que quedaron sin especificar.

Cuarto otro texto o control. Cualquier otro segmento portador de control o de texto (que no contenga un SYN) deberá llevar un ACK, y así será descartado por el procesado del ACK. Un segmento entrante RST podría no ser válido, ya que podría

no haber sido enviado como respuesta a nada que haya sido enviado por esta encarnación de la conexión. Así que, es improbable encontrarse en esta situación, pero si fuera así, ha de descartarse el segmento y retornar.

Si el estado es SYN-SENT, entonces:

Primero se comprueba el bit de ACK. Si el bit ACK vale uno. Si SEG.ACK =< ISS, o bien SEG.ACK > SND.NXT, se envía una orden de reinicio (a no ser que el bit de RST esté puesto a uno. Si fuera así, se descarta el segmento y se retorna)

<SEQ=SEG.ACK><CTL=RST>

y se descarta el segmento. Se retorna.

Si SND.UNA =< SEG.ACK =< SND.NXT entonces el ACK es aceptable.

Segundo, se comprueba el bit RST. Si el bit RST vale uno, entonces si el ACK era aceptable, entonces se indica al usuario "error: connection reset", se abandona el segmento, se pasa al estado CLOSED, se borra el TCB, y se retorna. En cualquier otro caso (sin ACK) se abandona el segmento y se retorna.

Tercero, se comprueba la seguridad y la prioridad. Si la seguridad/compartimentación del segmento no coincidieran exactamente con la seguridad/compartimentación del TCB, se envía una orden de reinicio. Si hubiera un ACK

<SEQ=SEG.ACK><CTL=RST>

En otro caso

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Si hubiera un ACK la prioridad del segmento deberá coincidir con la prioridad del TCB. De no ser así, se enviará una orden de reinicio.

<SEQ=SEG.ACK><CTL=RST>

Si no hubiera ACK, si la prioridad del segmento fuera mayor que la prioridad del TCB entonces, si lo permiten el usuario y el sistema, elévese la prioridad del TCB hasta el valor

correspondiente al segmento. Si no se concediera tal permiso para elevar la prioridad, deberá enviarse una orden de reinicio.

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Si la prioridad del segmento fuera menor que la prioridad en el TCB, continúese.

Si se hubiera enviado una orden de reinicio, descártese el segmento y retórnese.

Cuarto, comprobar el bit SYN. Este paso sólo debería tener lugar si el ACK fuera correcto, o en caso de no haber ACK, y siempre que el segmento no contuviera una orden RST.

Si el bit SYN vale uno y la seguridad/compartimentación y prioridad tienen valores aceptables RCV.NXT se pone a SEG.SEQ+1, IRS se pone a SEG.SEQ. SND.UNA debería incrementarse hasta igualar SEG.ACK (si hubiera un ACK), y cualesquiera segmentos en cola de retransmisión que queden así confirmados deberán ser eliminados.

Si SND.UNA > ISS (nuestro SYN ha sido confirmado mediante un segmento ACK), se cambia al estado de la conexión a ESTABLISHED (establecida), se construye un segmento ACK

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

y se envía. Podrán incluirse cualesquiera datos o controles que estuvieran almacenados para su transmisión. Si existieran otros controles o texto en el segmento, entonces se continúa procesando en el sexto paso indicado más abajo donde el bit URG queda comprobado, en cualquier otro caso, se retorna.

En caso contrario, se pasa a SYN-RECEIVED, se construye un segmento SYN,ACK

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

y se envía. Si existieran otros controles o texto en el segmento, se ponen en cola para procesar después de que el paso al estado ESTABLISHED se haya verificado. Se retorna.

Quinto, si ninguno de los bits SYN o RST valiera uno, entonces se abandona el segmento y se retorna.

En caso contrario, primero se comprueba el número de secuencia.

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Los segmentos se procesan secuencialmente. Aunque se practican comprobaciones iniciales de llegada con objeto de descartar los duplicados caducados, se efectúa un procesado ulterior en el orden según SEG.SEQ. Si los contenidos de un segmento produjeran un solapamiento de fronteras entre segmentos viejos y nuevos, sólo deberán procesarse las partes nuevas.

Existen cuatro casos posibles para la comprobación de aceptabilidad de un segmento entrante:

Segment Longit	Ventan Recepc	Comprobación
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	no aceptable
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND o bien RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Si el RCV.WND valiera cero, no se aceptará ningún segmento, pero deberá dejarse margen a la aceptación de órdenes ACK válidas, así como de segmentos URG y RST.

Si un segmento entrante fuera no aceptable, deberá enviarse un acuse de recibo en respuesta (siempre y cuando el bit RST valga uno. Si es así, se desecha el segmento y se retorna):

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Tras enviar el acuse de recibo, deshéchese el segmento no aceptable y se retorna.

En lo que sigue se supondrá siempre que el segmento es un segmento ideal que comienza con RCV.NXT y no sobrepasa el tamaño de la ventana. Uno podría siempre ajustar los segmentos dados de forma que satisfagan esta suposición recortando cualquier fragmento que caiga fuera de la ventana (incluyendo SYN y FIN), y continuando con el procesamiento únicamente si el segmento comienza con RCV.NXT tras la operación. Los segmentos con números de secuencia mayores se podrán mantener para su posterior procesamiento.

Segundo, se comprueba el bit RST.

ESTADO 'SYN-RECEIVED'

Si el bit RST bit vale uno:

Si esta conexión hubiera sido iniciada con un OPEN pasivo (i.e., provino de un estado LISTEN), entonces póngase de nuevo la misma al estado LISTEN y se retorna. El usuario no tiene por qué ser informado de esto. Si esta conexión se hubiera iniciado con un OPEN activo (i.e., provino de un estado SYN- SENT), entonces es que la conexión fue rechazada, se indica al usuario "connection refused". En cualquiera de ambos casos, todos los segmentos en la cola de retransmisión deberán ser eliminados. Y en el caso de un OPEN activo, se pasa al estado CLOSED, se borra el TCB, y se retorna.

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'

Si el bit RST vale uno, cualesquiera órdenes relevantes RECEIVEs y SEND deberían recibir respuestas "reset". Todas las colas de segmento deberán ser inicializadas. Los usuarios deberán también recibir un aviso unilateral (no solicitado) general de "connection reset". Se pasa al estado CLOSED, se borra el TCB, y se retorna.

ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Si el bit RST vale a uno, se pasa al estado CLOSED, se borra el TCB, y se retorna.

Tercero, comprobación de la seguridad y prioridad.

ESTADO 'SYN-RECEIVED'

Si la seguridad / compartimentación y prioridad del segmento no coincidieran exactamente con la seguridad / compartimentación y prioridad del TCB, deberá enviarse una orden de reinicio ('reset'), y retornar.

ESTADO 'ESTABLISHED'

Si la seguridad/compartimentación y prioridad del segmento no coincidieran exactamente con la seguridad/compartimentación y prioridad del TCB, deberá enviarse una orden de reinicio, y cualesquiera RECEIVEs y SEND deberán recibir respuestas "reset".

Todas las colas de segmento deberán ser inicializadas. Los usuarios deberán también recibir un aviso unilateral general de "connection reset". Se pasa al estado CLOSED, se borra el TCB, y se retorna.

Nótese que esta comprobación se ubica a continuación de la comprobación de secuencia para evitar que un segmento procedente de una conexión anterior entre los mismos puertos y con distintos valores de seguridad o prioridad origine una interrupción de la conexión actual.

Cuarto, comprobación del bit SYN.

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'

ESTADO 'FIN-WAIT-2'
ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Si el SYN estuviera en la ventana, denotará un error. Se envía un 'reset', y cualesquiera órdenes RECEIVE y SEND relevantes, deberían recibir respuestas "reset", todas las colas de segmentos deberán ser inicializadas, el usuario deberá también recibir una señal no solicitada y general de "connection reset". Se pasa al estado CLOSED, se borra el TCB, y se devuelve el resultado.

Si el SYN no estuviera en la ventana, no se alcanzaría este paso y se habría recibido un mensaje de acuse de recibo en el primer paso (comprobación del número de secuencia).

Quinto, comprobación del campo de ACK.

Si el bit de ACK vale cero, se descarta el segmento y se retorna.

Si el bit de ACK vale uno:

ESTADO 'SYN-RECEIVED'

Si SND.UNA =< SEG.ACK =< SND.NXT entonces se pasa al estado ESTABLISHED y se continúa procesando.

Si el acuse de recibo del segmento no fuera aceptable, se construye un segmento de reinicio,

<SEQ=SEG.ACK><CTL=RST>

y se envía.

ESTADO 'ESTABLISHED'

Si SND.UNA < SEG.ACK =< SND.NXT entonces, se envía SND.UNA <- SEG.ACK. Cualesquiera segmentos en la cola de retransmisión que se confirmen en esta llegada de segmentos serán eliminados. Los usuarios deberían recibir confirmaciones

positivas de los búferes que hayan sido completamente enviados y confirmados (i.e., el buffer SEND debería ser respondido con un mensaje de 'ok'). Si el ACK está duplicado (SEG.ACK < SND.UNA), puede ignorarse. Si el ACK confirma algún tramo aún sin enviar (SEG.ACK > SND.NXT), entonces se envía un ACK, se descarta el segmento, y se retorna.

Si SND.UNA < SEG.ACK =< SND.NXT, la ventana de envío deberá actualizarse. Si (SND.WL1 < SEG.SEQ o bien (SND.WL1 = SEG.SEQ y SND.WL2 =< SEG.ACK)), establézcase SND.WND <- SEG.WND, así como SND.WL1 <- SEG.SEQ, y envíese SND.WL2 <- SEG.ACK.

Nótese que SND.WND es un desplazamiento desde SND.UNA, que SND.WL1 registra el número de secuencia del último segmento usado para actualizar SND.WND, y que SND.WL2 registra el número de acuse de recibo correspondiente al último segmento usado para actualizar SND.WND. La comprobación aquí impide el uso de segmentos antiguos para actualizar la ventana.

ESTADO 'FIN-WAIT-1'

Además del procesamiento para el estado ESTABLISHED, si nuestro FIN queda ahora confirmado con un acuse de recibo, se pasa a FIN-WAIT-2 y se continúa procesando en ese estado.

ESTADO 'FIN-WAIT-2'

Además del procesamiento para el estado ESTABLISHED, si la cola de retransmisión estuviera vacía, podrá reconocerse un CLOSE por parte del usuario ('ok') pero no deberá borrarse el TCB.

ESTADO 'CLOSE-WAIT'

Repítase el mismo procesamiento válido para el estado ESTABLISHED.

ESTADO 'CLOSING'

Además del procesado para el estado ESTABLISHED, si el

ACK acusa recibo de nuestro FIN, entonces se pasa al estado TIME- WAIT, en cualquier otro caso, se ignora el segmento.

ESTADO 'LAST-ACK'

Lo único que puede llegar en este estado es un acuse de recibo de nuestro FIN. Si nuestro FIN no fuera reconocido, se borra el TCB, se pasa al estado CLOSED, y se retorna.

ESTADO 'TIME-WAIT'

Lo único que puede llegar en este estado es la retransmisión de un FIN remoto. Se realiza el acuse de recibo correspondiente y se reinicia la cuenta del tiempo de vida 2 MSL.

Sexto, se comprueba el bit URG:

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Si el bit URG vale uno, RCV.UP <- max(RCV.UP,SEG.UP), se informe al usuario de que la parte remota tiene datos urgentes si el puntero de urgencia (RCV.UP) está adelantado respecto a los datos pasados al usuario. Si el usuario ya hubiera sido informado (o estuviera aún en el "modo urgente") para esta secuencia continua de datos urgentes, no se informará al usuario de nuevo.

ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Esto no debería suceder, ya que se ha recibido un FIN desde la parte remota. Ignórese el URG.

Séptimo, se procesa el texto del segmento:

ESTADO 'ESTABLISHED'
ESTADO 'FIN-WAIT-1'
ESTADO 'FIN-WAIT-2'

Una vez en el estado ESTABLISHED, es posible entregar texto del segmento a los búferes RECEIVE del usuario. El texto de los segmentos podrá depositarse en los búferes hasta que, o bien el búfer esté lleno, o el segmento quede vacío. Si el segmento se vacía y lleva un indicador PUSH, entonces se notifica al usuario, cuando el búfer le es devuelto, de que se ha recibido un PUSH.

Cuando el módulo de TCP asume la responsabilidad de la entrega de datos al usuario, deberá también confirmar la recepción de los datos.

Toda vez que el módulo de TCP asuma la responsabilidad de los datos, avanzará RCV.NXT respecto a los datos aceptados, y ajustará RCV.WND en lo apropiado respecto a la disponibilidad del buffer actual. El total de RCV.NXT y RCV.WND así asignados no deberá ser reducido.

Se ruega tener en cuenta las sugerencias sobre gestión de ventanas en la sección 3.7.

Envíese un acuse de recibo de la forma:

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Este acuse de recibo deberá ser transportado ('piggybacked') en un segmento y transmitido si fuera posible sin incurrir en retraso indebido.

ESTADO 'CLOSE-WAIT'
ESTADO 'CLOSING'
ESTADO 'LAST-ACK'
ESTADO 'TIME-WAIT'

Esto no debería suceder, ya que se ha recibido un FIN desde la parte remota. Ignórese el texto del segmento.

Octavo, se comprueba el bit FIN:

No deberá procesarse el FIN si el estado fuera CLOSED, LISTEN o SYN-SENT, ya que el SEG.SEQ no podrá ser validado;

se descarta el segmento y se retorna.

Si el bit FIN vale uno, se informa al usuario mediante "connection closing" y se devuelve cualesquiera mensajes RECEIVE pendientes mediante el mismo mensaje, se avanza RCV.NXT respecto al FIN, y se envía un acuse de recibo para el FIN. Nótese que FIN implica PUSH para cualquier texto del segmento que aún no ha sido entregado al usuario.

ESTADO 'SYN-RECEIVED'
ESTADO 'ESTABLISHED'

Se pasa al estado CLOSE-WAIT.

ESTADO 'FIN-WAIT-1'

Si nuestro FIN ha sido confirmado por un acuse de recibo (quizá en este mismo segmento), entonces se pasa a TIME-WAIT, se inicia el contador de tiempo 'time-wait', se interrumpen todos los otros contadores; en cualquier otro caso, se pasa al estado CLOSING.

ESTADO 'FIN-WAIT-2'

Se pasa al estado TIME-WAIT. Se inicia el contador de tiempo 'time-wait', se interrumpen todos los demás contadores de tiempo.

ESTADO 'CLOSE-WAIT'

Permanecer en el estado CLOSE-WAIT.

ESTADO 'CLOSING'

Permanecer en el estado CLOSING.

ESTADO 'LAST-ACK'

Permanecer en el estado LAST-ACK.

ESTADO 'TIME-WAIT'

Permanecer en el estado TIME-WAIT. Reiniciar el tiempo de vida 2 MSL y retornar.

TIEMPO DE ESPERA DE USUARIO

Si expirase el tiempo de espera del usuario para cualquier estado, inicializar todas las colas, informar al usuario mediante "error: connection aborted due to user timeout". En general, y para cualesquiera llamadas relevantes, se borra el TCB, se pasa al estado CLOSED y se retorna.

TIEMPO DE ESPERA DE RETRANSMISIÓN

Para cualquier estado, si expirase el tiempo de vida de retransmisión de un segmento en la cola de retransmisión, se envía de nuevo el segmento que está al comienzo de la cola de retransmisión, se reinicia el contador de tiempo de retransmisión, y se retorna.

TIEMPO DE ESPERA EN EL ESTADO 'TIME-WAIT'

Si expirase el tiempo de espera en el estado TIME-WAIT de una conexión, se borra el TCB, se pasa al estado CLOSED y se retorna.

Glosario

1822
Informe BBN 1822 "The Specification of the Interconnection of a Host and an IMP". La especificación de la interfaz entre un 'host' u ARPANET.

ACK
Un bit de control (del inglés 'acknowledge') que no ocupa posición del espacio de números de secuencias y que indica que el campo de acuse de recibo de este segmento especifica el próximo número de secuencia que el emisor de este segmento espera recibir, por lo que también realiza un acuse de recibo de todos los números de secuencia anteriores.

ARPANET, mensaje de
La unidad de transmisión entre un 'host' y un IMP en ARPANET. Su tamaño máximo es de 1012 octetos (8096 bits).

ARPANET, paquete de
Una unidad de transmisión que ARPANET utiliza internamente entre los IMP. Su tamaño máximo es de 126 octetos (1008 bits).

cabecera
Información de control al comienzo de un mensaje, segmento, fragmento, paquete o bloque de datos.

conector
En inglés: 'socket'. Dirección que específicamente incluye un identificador de puerto, es decir, la concatenación de una dirección de internet con un puerto de TCP.

conexión
Un camino lógico de comunicación identificado por un par de conectores.

datagrama
Un mensaje que se envía en una red de comunicaciones de ordenadores por intercambio de paquetes.

dirección de destino
La dirección de destino, habitualmente los identificadores

de red y de 'host'.

dirección de origen
: la dirección de origen, generalmente los identificadores de red y de 'host'.

envío, secuencia de
: Siguiente número de secuencia que el TCP local (emisor) utilizará en la conexión. Se selecciona inicialmente de una serie de números de secuencias iniciales (ISN, 'initial sequence number') y se incrementa por cada octeto de datos o de control que se transmita.

envío, ventana de
: Representa los números de secuencia que el TCP remoto (receptor) espera recibir. Es el valor del campo de ventana que se especifica en los segmentos provenientes del TCP remoto (receptor de los datos). El rango de los nuevos números de secuencia que un TCP puede emitir va desde SND.TXT hasta SND.UNA + SND.WND - 1. (Las retransmisiones de los números de secuencia entre SND.UNA y SND.NXT son de esperar, por supuesto).

FIN
: Un bit de control (de 'finis') que ocupan una posición del espacio de números de secuencia y que indica que el emisor no enviará más datos o información de control que ocupe posiciones del espacio de números de secuencia.

fragmento
: Una porción de una unidad lógica de datos, en particular, un fragmento de internet es una porción de un datagrama de internet.

FTP
: Un protocolo de transferencia de ficheros.

host
: Un computador. En particular, un origen o destino de los mensajes desde el punto de vista de la red de comunicaciones.

identificación
: Un campo del protocolo de internet. Identifica el valor asignado por el emisor que sirve de ayuda para el ensamblaje de los fragmentos de un datagrama.

IMP
: La interfaz de procesamiento de mensajes o 'Interface Message Processor', el commutador de paquetes de ARPANET.

internet, dirección de
: Una dirección de origen o de destino específica del nivel de 'host'.

internet, datagrama de
: La unidad de datos que se intercambia entre un módulo de internet y el protocolo de nivel superior más la cabecera de internet.

internet, fragmento de
: Una porción de los datos de un datagrama de internet con una cabecera de internet.

IP
: El protocolo de internet o 'Internet Protocol'.

IRS
: El número de secuencia de recepción inicial ('Initial Receive Sequence number'). El primer número utilizado por el emisor en una conexión.

ISN
: El número de secuencia inicial ('Initial Sequence Number'). El primer número de secuencia utilizado en una conexión (el ISS o el IRS). Se selecciona por un procedimiento basado en el tiempo de reloj.

ISS
: El número de secuencia de envío inicial ('Initial Send Sequence'). El primer número de secuencia utilizado por el emisor en la conexión.

líder
: Información de control del comienzo de un mensaje o bloque de datos. En particular, en ARPANET, la información en un mensaje de ARPANET en la interfaz 'host'-IMP.

módulo
: Una implementación, habitualmente de 'software', de un protocolo u otro procedimiento.

MSL
: Vida máxima del segmento ('Maximum Segment Lifetime'), el tiempo que un segmento de TCP puede existir en el sistema de internet. Se define por convención en 2 minutos.

octeto
: Un byte de ocho bits.

opciones
: Un campo de opción puede contener varias opciones, y cada opción puede tener varios octetos de longitud. Las opciones se utilizan sobre todo en situaciones de pruebas; por ejemplo para transportar marcas de tiempo.

paquete
: Un conjunto de datos con una cabecera que puede estar o no lógicamente completa. Más a menudo, se refiere a un empaquetamiento físico de datos que lógico.

paquete local
: La unidad de transmisión dentro de una red local.

puerto
: La porción de un conector que especifica qué entrada lógica o canal de salida se asocian con los datos.

puntero urgente
: Un campo de control significativo solamente cuando el bit URG está establecido a uno. Este campo comunica el valor del puntero urgente que indica el octeto de datos asociado con la llamada urgente del usuario emisor.

proceso
: Un programa en ejecución. Un origen o destino de datos desde el punto de vista de TCP o cualquier otro protocolo de 'host' a 'host'.

PUSH
: Un bit de control que no ocupa posición en el espacio de número de secuencias, y que indica que un segmento que contiene datos debe entregarse inmediatamente ('pushed') al usuario receptor.

RCV.NXT
: Siguiente número de secuencia de recepción.

RCV.UP
: Puntero urgente de recepción.

RCV.WND
: Ventana de recepción.

recepción, número de secuencia de recepción
: Número de secuencia que el TCP local espera recibir.

recepción, ventana de
: Representa los números de secuencias que el TCP local (receptor) espera recibir. De este modo, el TCP local considera que los segmentos que intersecten con el rango RCV.NXT a RCV.NXT + RCV.WND - 1 transportan datos o bits de control aceptables. Los segmentos que contienen números de secuencia completamente fuera de este rango se consideran duplicados y se descartan.

RST
: Un bit de control ('reset' o reinicio), que no ocupa posición en el espacio de secuencias, y que indica que el receptor debe eliminar la conexión sin más interacción. El receptor puede determinar, basándose en los campos de número de secuencia y de número de acuse de recibo, si debe obedecer o ignorar la orden de reinicio. En ningún caso, la recepción de un segmento que contiene un RST dará lugar a una respuesta con otro RST.

RTP
: Protocolo de tiempo real ('Real Time Protocol'): un protocolo de nivel 'host' a 'host' para la comunicación de información crítica de tiempo.

secuencia a la izquierda
: Número de secuencia siguiente en espera de que el TCP reciba su acuse de recibo (o el número de secuencia más bajo en la actualidad sin confirmación) y que a veces se denomina el borde izquierdo de la ventana de envío.

SEG.ACK
: Acuse de recibo del segmento.

SEG.LEN
: Longitud del segmento.

SEG.PRC

Valor de prioridad del segmento.

SEG.SEQ
: Secuencia del segmento.

SEG.UP
: Campo de puntero urgente del segmento.

SEG.WND
: Campo de ventana del segmento.

segmento
: Una unidad lógica de datos, en particular un segmento de TCP es la unidad de datos transferida entre dos módulos de TCP.

segmento, acuse de recibo del
: El número de secuencia en el campo de acuse de recibo del segmento entrante.

segmento, longitud del
: La cantidad del espacio de número de secuencias utilizada por un segmento, incluyendo cualquier bit de control que ocupe posiciones del espacio de número de secuencias.

segmento, secuencia del
: El número en el campo de número de secuencia del segmento entrante.

SND.NXT
: Secuencia de envío.

SND.UNA
: Secuencia a la izquierda.

SND.UP
: Puntero urgente de envío.

SND.WL1
: Número de secuencia del segmento utilizado en la última actualización de la ventana.

SND.WL2
: Número de acuse de recibo del segmento utilizado en la última actualización de la ventana.

SND.WND
: Ventana de envío.

SYN
: Un bit de control del segmento entrante, que ocupa un número de secuencia, y que se utiliza en el inicio de una conexión para indicar dónde empezará la numeración secuencial.

TCP
: Bloque de control de la transmisión ('Transmission control block'), la estructura de datos que registra el estado de una conexión.

TCB.PRC
: La prioridad de una conexión

TCP
: Protocolo de control de transmisión ('Transmission Control Protocol'): un protocolo de 'host' a 'host' para las comunicaciones fiables en entornos de internet.

TOS
: Tipo de servicio ('Type of Service'), un campo del protocolo de internet.

Tipo de servicio
: Un campo del protocolo de internet que indica el tipo de servicio para este fragmento de internet.

URG
: Un bit de control (urgente), que no ocupa posición en el espacio de números de secuencia y que se utiliza para indicar que se le debería notificar al usuario receptor que realice un procesamiento urgente tan pronto como los datos le vayan siendo entregados mientras que sus números de secuencia sean menores que los indicados en el puntero urgente.

Referencias

[1] Cerf, V., and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. COM-22, Nº. 5, págs 637-648, Mayo de 1974.

[2] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, Septiembre de 1981.

[3] Dalal, Y. and C. Sunshine, "Connection Management in Transport Protocols", Computer Networks, Vol. 2, No. 6, pp. 454-473, Diciembre de 1978.

[4] Postel, J., "Assigned Numbers", RFC 790, USC/Information Sciences Institute, September de 1981.

Planos

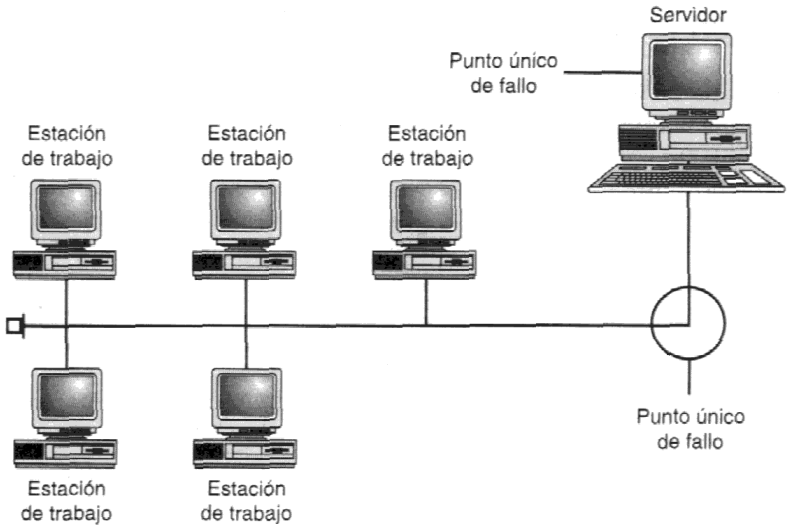


Figura 2-1. Topología en bus.

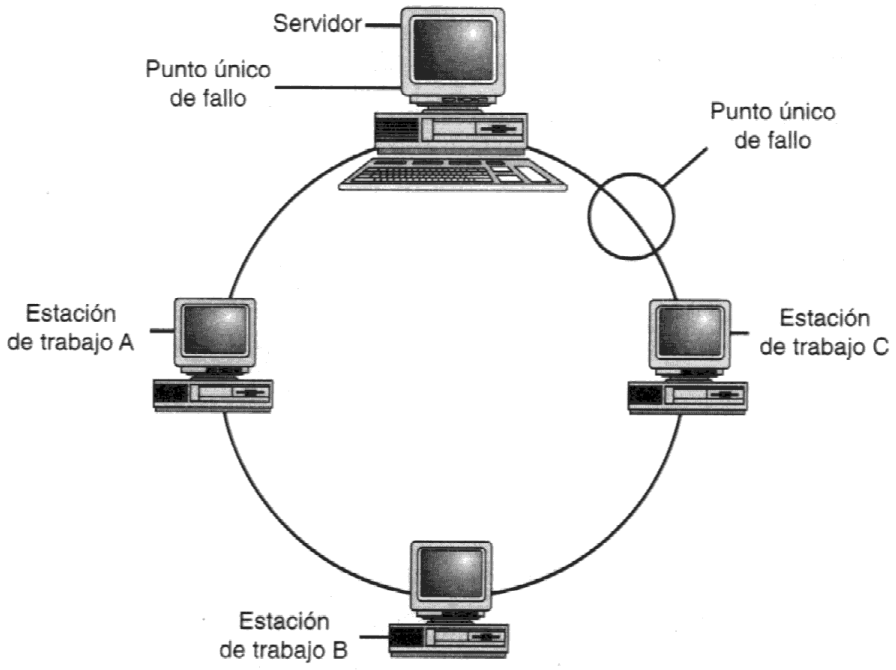


Figura 2-2. Topología en anillo.

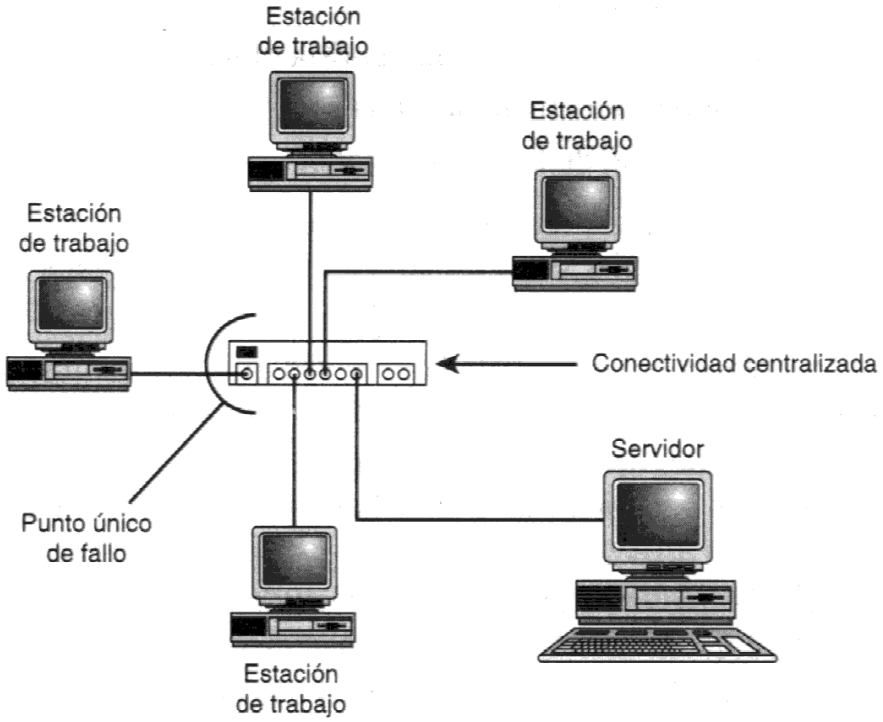


Figura 2-3. Topología en estrella.

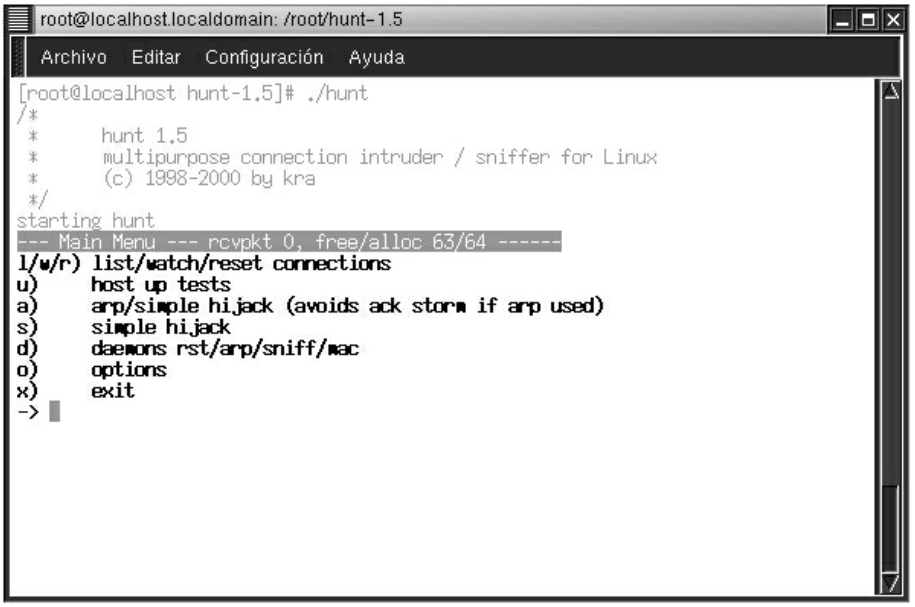


Figura 4-1. Menú principal de Hunt.

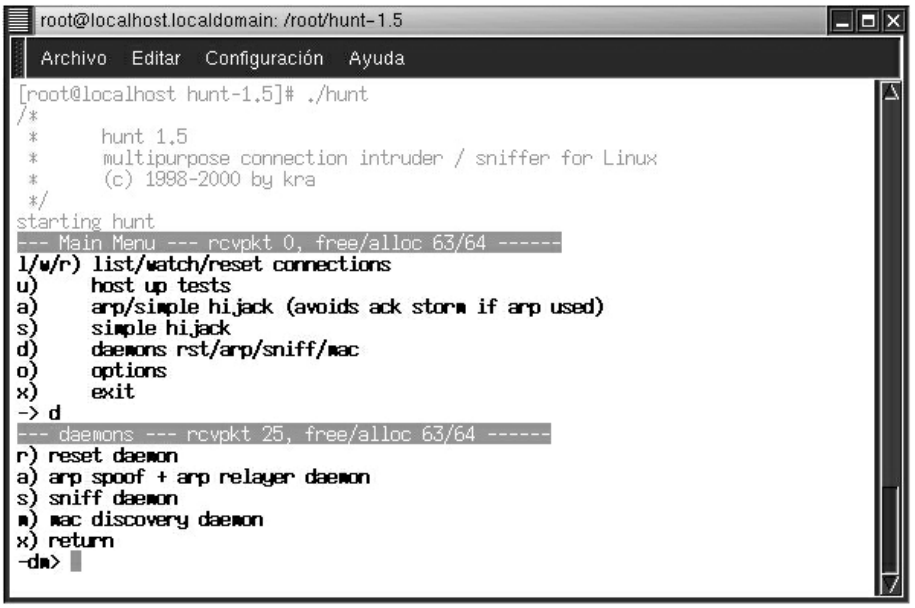


Figura 4-2. Menú de demonios de Hunt.

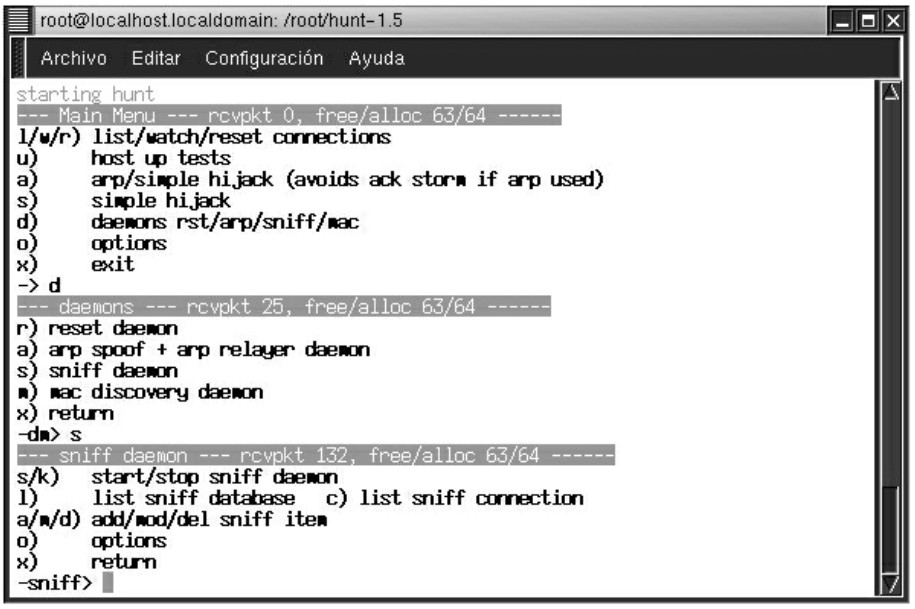


Figura 4-3. Menú del demonio de sniffing de Hunt.

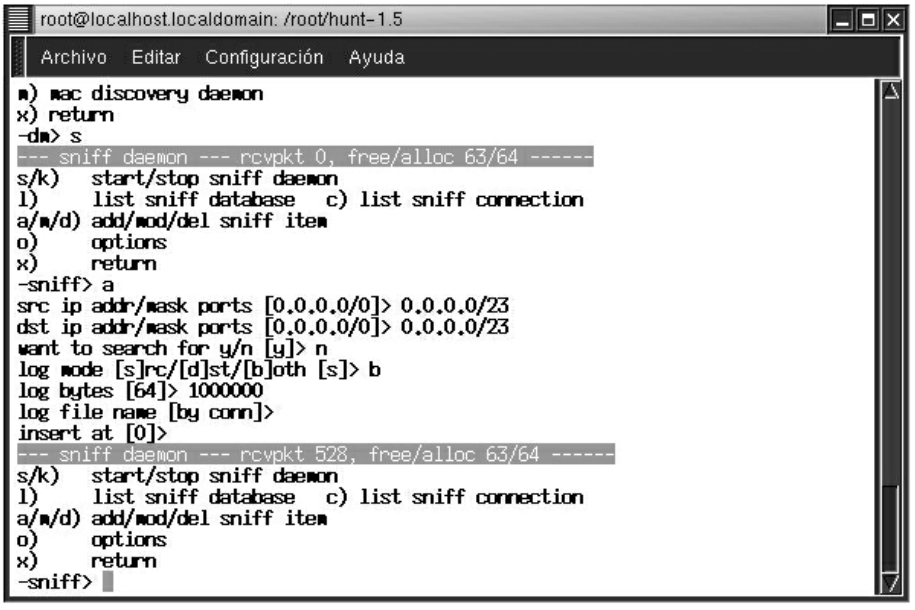


Figura 4-4. Menú de demonios de Hunt tras introducir los parámetros de escucha electrónica.

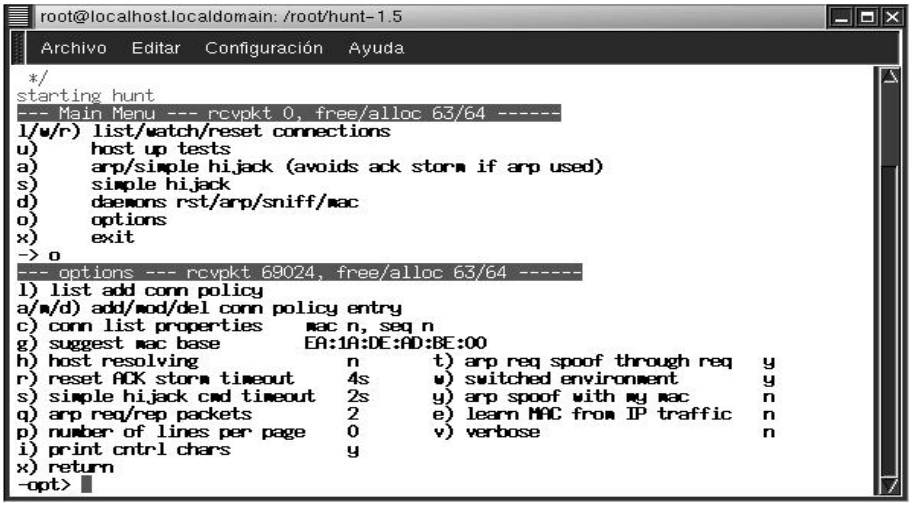


Figura 4-5. Menú de opciones de Hunt.

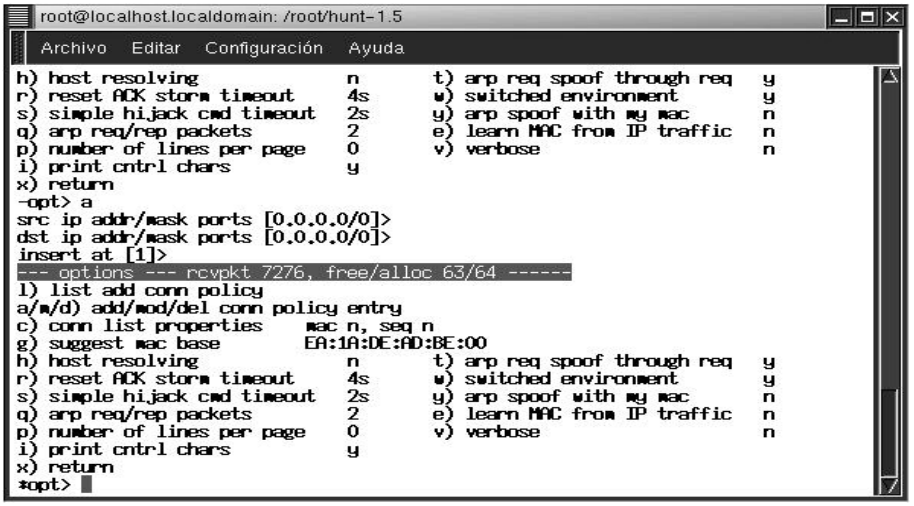


Figura 4-6. Menú de opciones de Hunt después de introducir las direcciones fuente y destino que se van a vigilar.

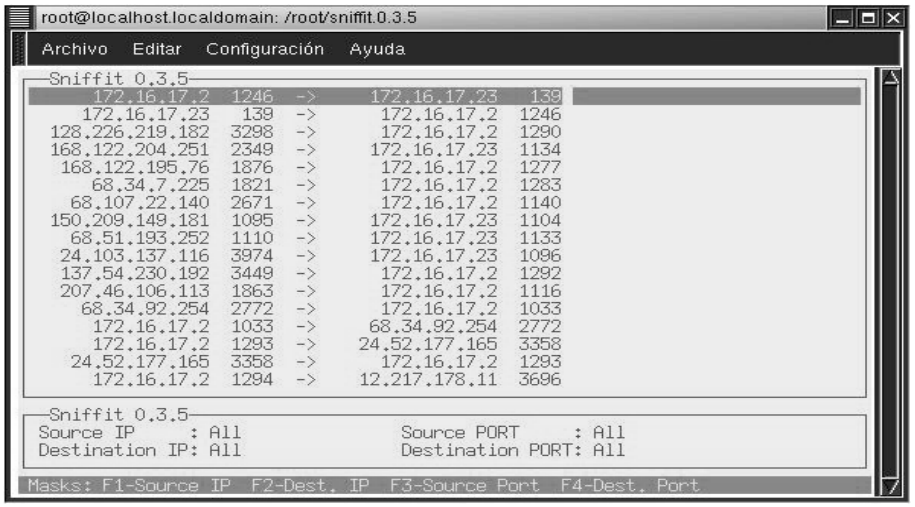


Figura 4-7. Modo interactivo de Sniffit mostrando la lista de conexiones activas.

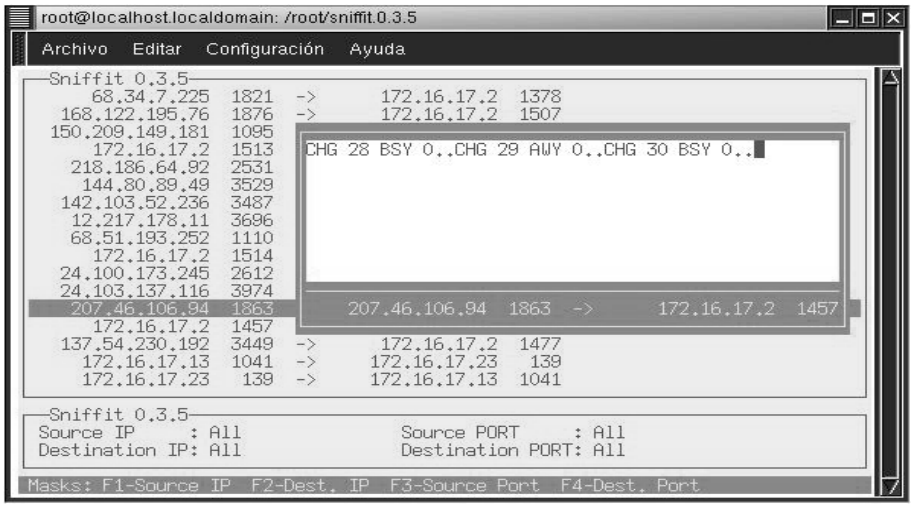


Figura 4-8. Modo interactivo de Sniffit mostrando los datos capturados a una conexión a MSN Instant Message Service.

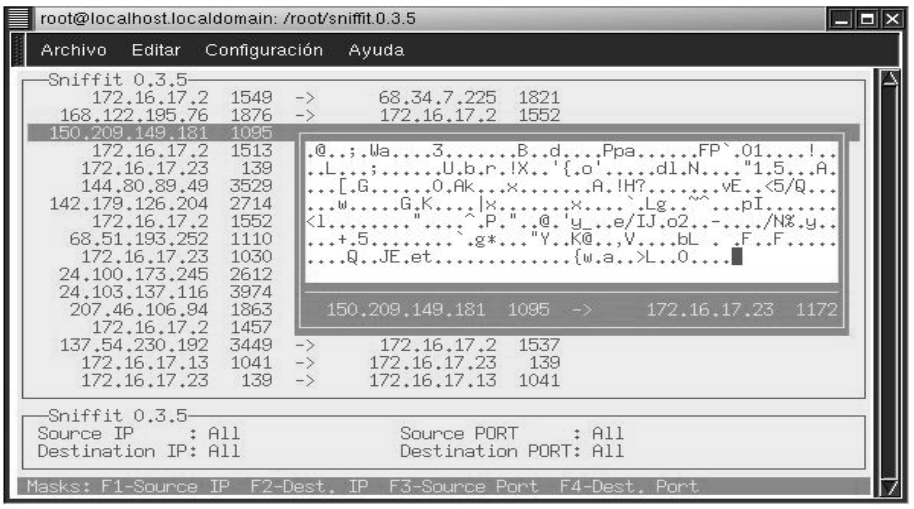


Figura 4-9. Modo interactivo de Sniffit mostrando los datos capturados en una conexión FTP.

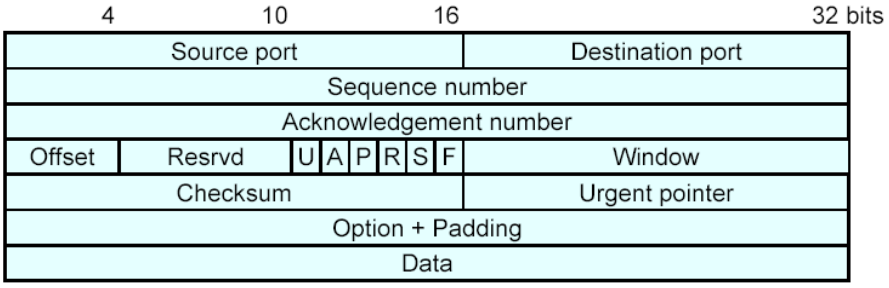


Figura 5-1. Formato de un segmento TCP.

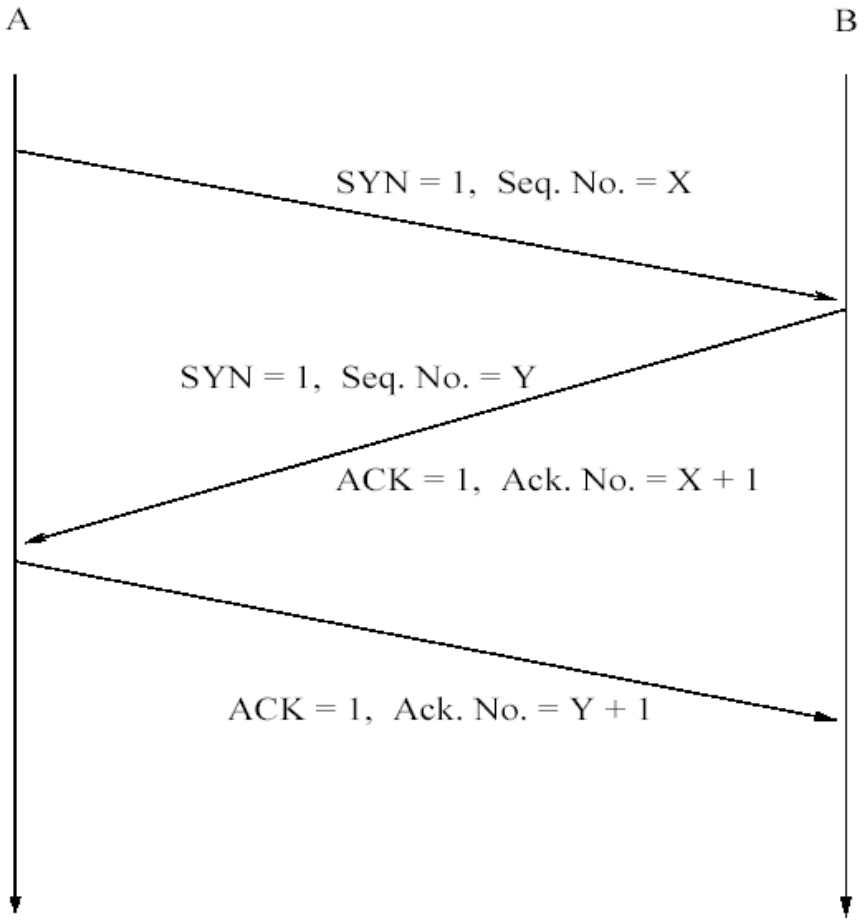


Figura 5-2. Establecimiento de una conexión TCP mediante el protocolo de sincronización a tres bandas.

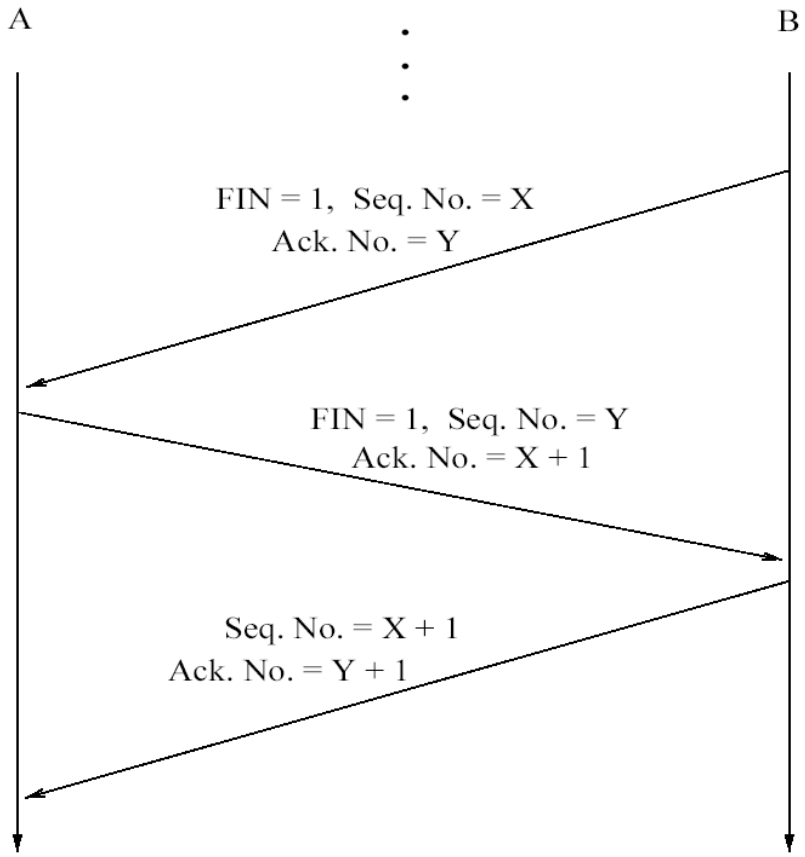


Figura 5-3. Liberación de una conexión TCP mediante el protocolo de acuerdo a tres bandas.

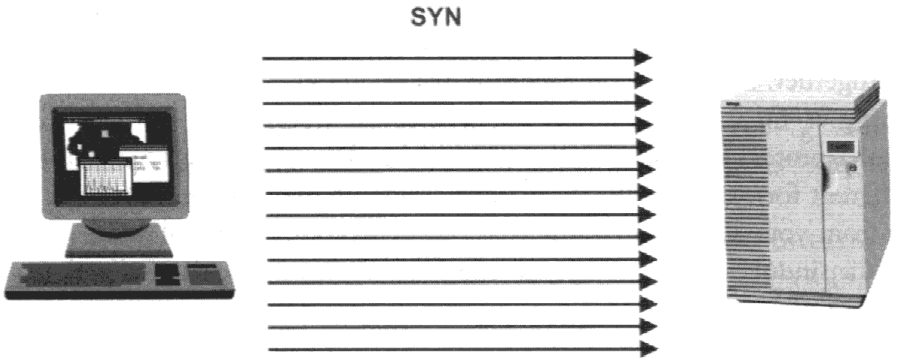


Figura 6-1. Ataque TCP SYN Flooding de denegación del servicio.

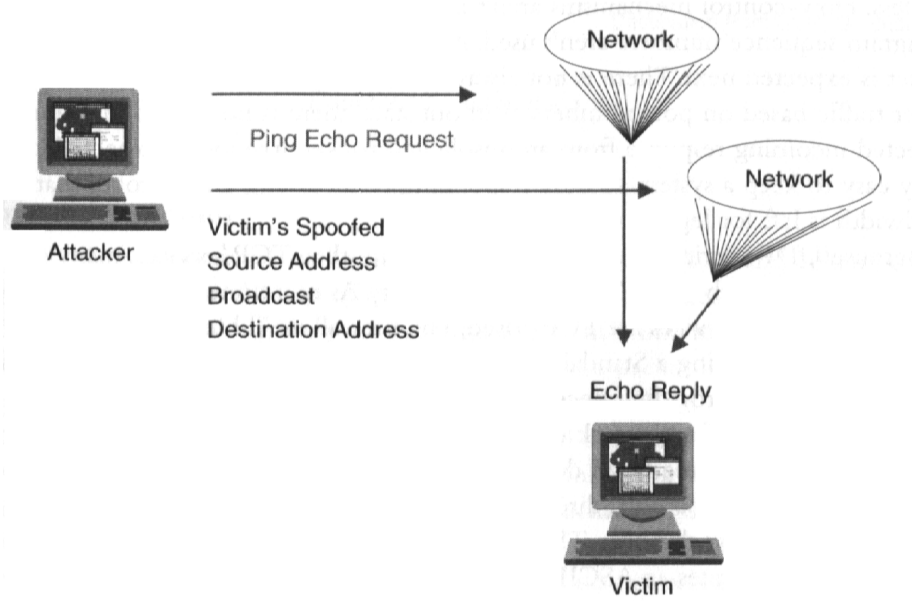


Figura 6-2. Ataque Ping Flooding de denegación del servicio.

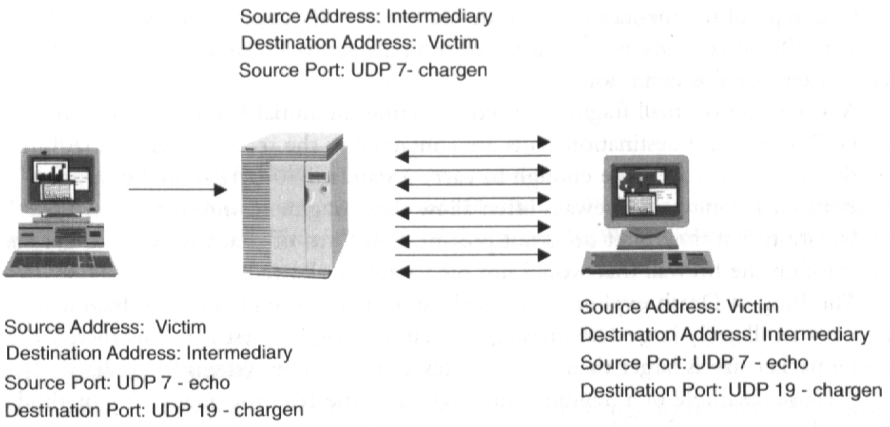


Figura 6-3. Ataque UDP Flooding de denegación del servicio.

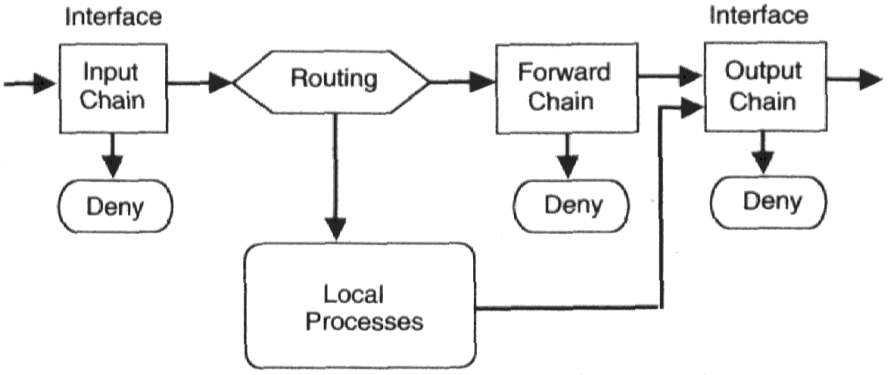


Figura 7-1. Recorrido de los paquetes con el firewall IPFW.

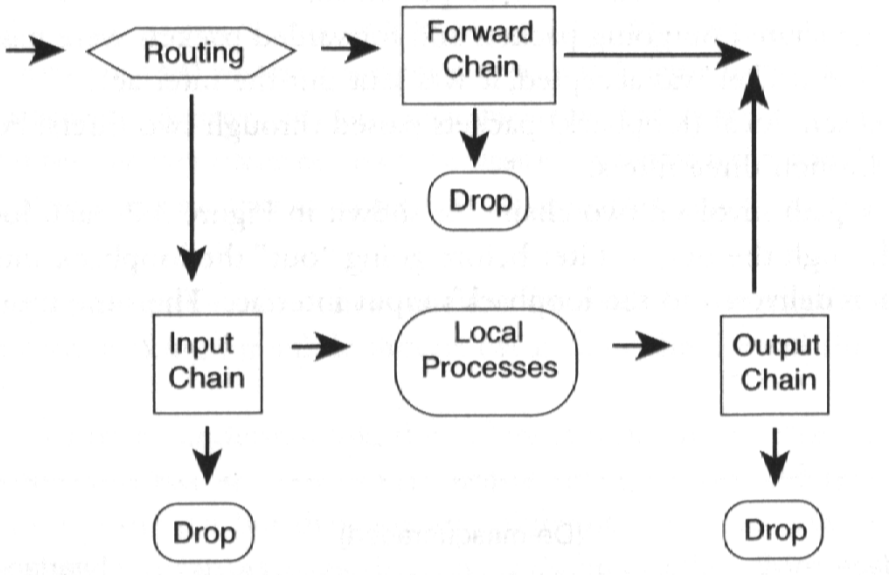


Figura 7-2. Recorrido de los paquetes con el firewall Netfilter.

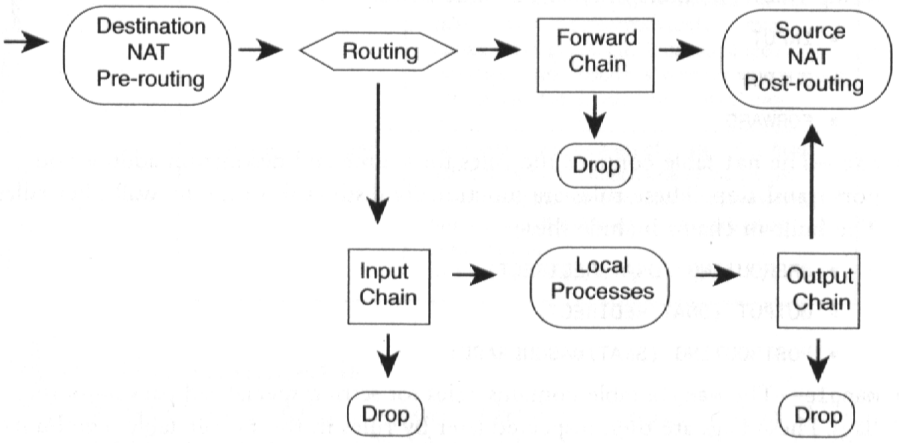


Figura 7-3. Recorrido de los paquetes en la traducción de direcciones (NAT).

Presupuesto

Presupuesto

Para llevar acabo este estudio de seguridad ha sido necesario el trabajo de un ingeniero durante 4 meses a jornada completa. Se detallan a continuación los costes correspondientes.

Uds.	Concepto	Precio unitario (€)	Precio total (€)
688,00	Hora de trabajo de Ingeniero de Telecomunicación	15,00	10.320,00
		TOTAL PRESUPUESTO	**10.320,00**