Universidad del Azuay

Facultad de Administración

Escuela de Ingeniería de Sistemas

"TUTORIAL PARA LA IMPLEMENTACIÓN DE UN FIREWALL USANDO LINUX COMO SISTEMA OPERATIVO E IPTABLES Y CONTROL DE ANCHO DE BANDA"

Trabajo de graduación previo
a la obtención del título de
Ingeniera en Sistemas

Autor:

Elizabeth Marilú Gonzaga Gonzaga

Director:

Ing. Luis Calderon Peralta

Cuenca, Ecuador
2008

DEDICATORIA

Esta monografía como todos mis triunfos, va dedicada a mi madre, la Sr. Esthela Marilú Gonzaga.

Gracias madre por poner tu confianza en mí, y esforzarte sola por darme todas las herramientas, mapas y amor que necesité para formarme en la persona que soy. Este logro también es tuyo.

Te amo

Tu hija, Elizabeth

AGRADECIMIENTOS

Mis agradecimientos más sinceros a quienes me orientaron en mi proyecto académico. Como son: Mi director monográfico, el Ing. Luis Calderon Peralta; Mi tutor, el Ing. Pablo Esquivel León. Y en especial a un amigo incondicional, el Ing. Geovanny Orellana Morales.

Elizabeth

INDICE DE CONTENIDOS

RESUMEN

Este tutorial es una implementación de un firewall y el control de ancho de banda por usuario, bajo el sistema operativo Linux.

El primer capítulo, contiene la investigación de lo necesario para la creación de un *firewall* para la seguridad de una organización, usando *iptables* como herramienta principal, con el único objetivo de controlar cada paquete TCP/IP que ingresa/atraviesa/sale desde/hacia una máquina.

El segundo capítulo, consta de configuraciones necesarias de Squid, una herramienta que se utiliza para el tráfico de la web, mejorando el rendimiento, ofreciendo mayor rapidez de navegación y proporcionando contenido estático, dinámico y flujo de los usuarios de Internet dentro de la organización.

El tercer capítulo, es la implementación de todos estos conceptos, a través de ejercicios prácticos del funcionamiento de un firewall y el control de ancho de banda mediante las herramientas como iptables y Squid respectivamente.

ABSTRACT

This tutorial is a firewall implementation and bandwidth control per user under Linux operative system. The first chapter contains the necessary research for the cration of a firewall for the safety of an organization, using iptables as the main tool with the only objectives of controlling each TCP/IP package that enters, goes through, or goes out from or to a machine.

The second chapter is composed of necessary configurations of Squid, a tool that is used for the traffic of the web, improving the perfomance, offering greater surfing swiftness, and providing satic and dynamic content and a flow of Internet users within the organization.

The third chapter is the implementation of all these concepts through practical exercises on the functioning of a firewall and bandwidth control through iptables and Squid tools respectively.

INTRODUCCION

En la actualidad el uso de la Internet se esta haciendo más habitual, parte de la información que requerimos está en ella. Y la necesidad es asegurar que la información que atraviesa desde/hacia una red debe ser confiable por lo que es aconsejable tener un cortafuego o firewall, para la seguridad de la misma. Por otro lado quiero comentar que el término firewall es un término que involucra muchas cosas, pero en conclusión a la definición de firewalls es como un sistema de filtrado de paquetes. Para instalar un firewall es un asunto complejo que requiere una configuración cuidadosa. Los firewalls se pueden usar en cualquier red. Es habitual tenerlos como protección de internet en las organizaciones, aunque ahí también suelen tener una doble función: controlar los accesos externos hacia dentro y también los internos hacia el exterior; esto último se hace con el firewall o frecuentemente con un proxy.

Hay que recalcar que la demanda del servicio de internet esta en incremento, por lo que un control para el ancho de banda para los usuarios que disponen este servicio es oportuno para una utilización adecuada de los recursos de red disponibles.

El control se realizará con la configuración adecuada de los servicios que dispone de Linux Centos 5.1

1. CAPITULO UNO: FIREWALL

1.1 INTRODUCCION CONCEPTUAL

Un firewall o cortafuegos es un sistema informático que actúa de punto de conexión segura entre dos o más redes, es decir que sirve entre la comunicación cumpliendo determinadas trayectorias de seguridad, decidiendo si un paquete pasa, se modifica, se convierte o se descarta.

Según las necesidades de la red se puede dar seguridad con uno o más firewalls. Un uso típico es usarlo en medio de una red local y la red de internet. Teniendo como protección, controlando los accesos externos hacia dentro y también los internos hacia el exterior. Para permitir o denegar una comunicación, el firewall examina el tipo de servicio al que corresponde, como puede ser el web, el correo o ftp.

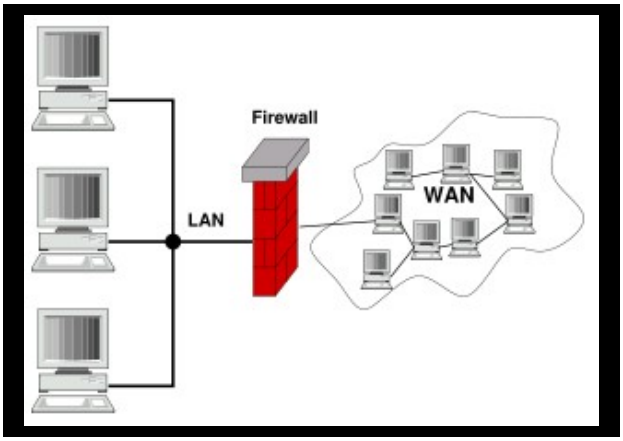
En su mayoría, un firewall, es un conjunto de reglas en las que se examina el origen y destino de los paquetes del protocolo TCP/IP, además de filtrar otros tipos de paquetes como los UDP, ICMP, GRE.

Un firewall puede ser un dispositivo software o hardware, es decir, un aparato que se conecta entre la red y el cable de la conexión a Internet, o bien un programa que se instala en la máquina que tiene el modem que conecta con Internet.

1.2 QUÉ ES UN FIREWALL?

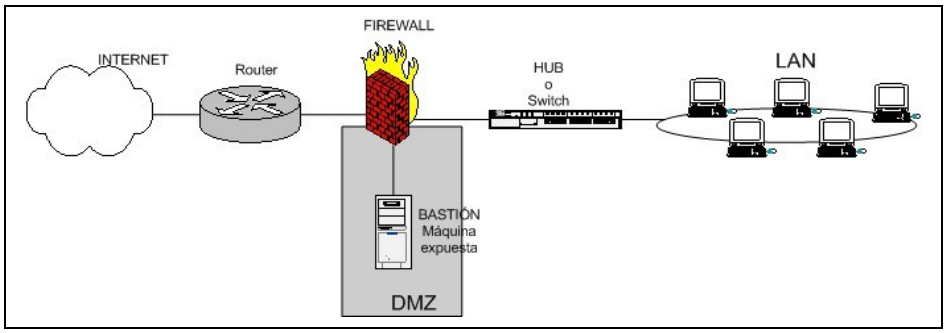
Un firewall es básicamente es un elemento de hardware o software utilizado en una red, que nos sirve para el control de las comunicaciones, permitiéndo o prohibiendo según se haya definido la organización responsable de la red.

La ubicación habitual de un firewall es el punto de conexión de la red interna de la organización con la red exterior, que puede ser Internet, así se protegerá las vulnerabilidades de los sistemas de la red interna.



Hay firewalls más complejos como los **DMZ o Zona Desmilitarizada** en la que se ubican los servidores de la organización que deben permanecer accesibles desde la red exterior. Este tipo de firewall se puede dar al establecer distintos perímetros de seguridad en torno a un sistema.

Es frecuente que se necesite exponer algún servidor a internet como es el caso de un servidor web o de correo, etc., y en esos casos en principio se debe aceptar cualquier conexión a ellos. Lo que se recomienda en esa situación es situar ese servidor en un lugar aparte de la red, permitiendo que el servidor sea accesible desde internet de tal forma que si es atacado y se gana acceso a él, la red local sigue protegida por el firewall.



Los firewalls pueden tener doble función: controlar los accesos externos hacia dentro y también los internos hacia el exterior.

1.2.1 Cómo funciona un firewall?

Es esencial saber primero cómo funciona una conexión y el tráfico en general. Cuando nos conectamos a internet nos estamos conectando a una red inmensa de ordenadores, es decir, tenemos acceso directo a todo aquel que está conectado en ese mismo momento.

Para que la comunicación sea establecida se abren una serie de puertos de comunicación en los ordenadores en donde los programas se ponen a “hablar y a escuchar” para interactuar.

- Un ordenador que ofrece una página web, se pone a escuchar y atender solicitudes en el puerto 80.
- Cuando se establece una solicitud, el servidor web dialoga con el cliente y si se cumple el proceso, la comunicación se establece y se muestra la página web.

Lo que ocurre es que los sistemas operativos actuales, e infinidad de programas, por defecto abren puertos sin notificar al usuario y por tanto estamos con unos puertos abiertos.

En el proceso de comunicación, o sea cuando se intenta establecer una conexión, es cuando se involucra a los iptables, que se pone por delante de los puertos y decide qué paquete de información puede pasar, según las reglas establecidas según la organización de la red.

Estas reglas se almacenan en unas tablas, con reglas en orden descendente, donde la última siempre será la última regla en aplicarse, antes de que se aplique la regla de defecto.

1.2.2 Tipos de firewall

Filtrado de paquetes o a nivel de capa de red

Sirve de filtrado de paquetes IP a nivel de red. Se puede filtrar según los distintos campos de los paquete IP: dirección IP origen, dirección IP destino.
Además a nivel de transporte se puede filtrar a partir del puerto origen y destino, y a nivel de enlace de datos se filtra con la dirección MAC.

Firewall a nivel de capa de aplicación

Los filtrados se pueden adaptar a características propias de los protocolos de este nivel. O sea, si se trata de tráfico HTTP se pueden realizar filtrados según la URL a la que se está intentando acceder. Un firewall a nivel de la capa de aplicación de tráfico HTTP es normalmente denominado Proxy y permite que los computadores de una organización entren a internet de una forma controlada.

Firewall personal

Simplemente se instala como software en un computador, filtrando las comunicaciones entre dicho computador y el resto de la red y viceversa.

1.2.2 Políticas del firewall

Política restrictiva: Se rechaza todo el tráfico excepto el que está explícitamente permitido. El firewall obstruye todo el tráfico y hay que habilitar expresamente el tráfico de los servicios que se necesiten.

Política permisiva: Se permite todo el tráfico excepto el que esté explícitamente denegado. Cada servicio potencialmente peligroso necesitará ser aislado básicamente caso por caso, mientras que el resto del tráfico no será filtrado.

1.2.4 Ventajas de un firewall

▪Protege de intrusiones, el acceso a ciertos segmentos de la red de una organización, sólo se permite desde máquinas autorizadas de otros segmentos de la organización o de Internet.

▪Protege la información privada, permitiendo definir distintos niveles de acceso a la información de manera que en una organización cada grupo de usuarios definido tendrá acceso sólo a los servicios y la información que le son estrictamente necesarios.

▪Optimiza el acceso, identifica los elementos de la red interno y optimiza que la comunicación entre ellos sea más directa. Estos ayuda a reconfigurar los parámetros de seguridad.

1.2.5 Limitaciones de un firewall

- Un firewall se limita a proteger contra aquellos ataques cuyo tráfico no pase a través suyo.

- El firewall no puede proteger de las amenazas a que está sometido por ataques internos o usuarios negligentes. Es decir, no puede prohibir a espías corporativos copiar datos sensibles en medios físicos de almacenamiento.

- El firewall no puede proteger contra los ataques posibles a la red interna por virus informáticos a través de archivos y software.

- El firewall no protege de los fallos de seguridad de los servicios y protocolos de los cuales se permite el tráfico. Hay que configurar correctamente y cuidar la seguridad de los servicios que se publiquen a Internet.

1.3 TCP/IP TRANSMISSION CONTROL PROTOCOL

1.3.1 Introducción

Hay que tener en cuenta que al conectarse por internet nos conectamos en ambos sentidos. Teniendo en cuenta que un firewall se encarga del filtrado de paquetes TCP/ UDP/ ICMP/ IP.

Iptables trabaja dentro de la capa de internet y de transporte, además de trabajar bajo la capa de aplicación, aunque no es lo usual.

TCP (*Transmission Control Protocol*), provee un servicio de transmisión confiable, orientado a la conexión, de flujo de bytes.

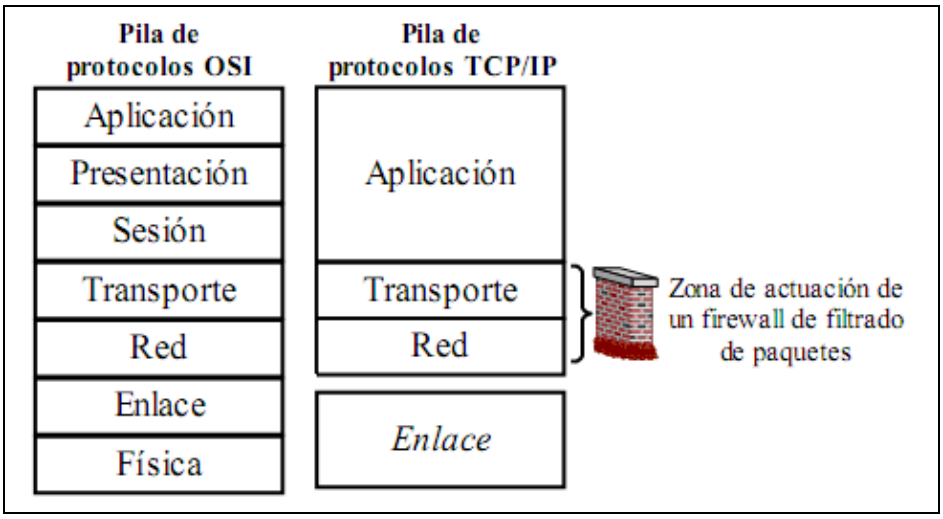
TCP/IP tiene varios campos, los más notables para el estudio serían los que nos ayudan con la vigilancia de iptables, como: dirección origen, dirección destino, puerto origen, puerto de destino, etc.

1.3.2 Historia

TCP o Protocolo de Control de Transmisión, es uno de los protocolos fundamentales en Internet. Fue creado entre los años de 1973 – 1974 por Vint Cerf y Robert Kahn. Muchos programas dentro de una red de datos compuesta por ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. TCP da soporte a muchas de las aplicaciones más populares de Internet, incluidas HTTP, SMTP y SSH.

1.3.3 TCP/IP Capas

Las capas están jerarquizadas. Cada capa se construye sobre su predecesora. El número de capas y, en cada una de ellas, sus servicios y funciones son variables con cada tipo de red. Sin embargo, en cualquier red, la misión de cada capa es proveer servicios a las capas superiores haciéndoles transparentes el modo en que esos servicios se llevan a cabo. De esta manera, cada capa debe ocuparse exclusivamente de su nivel inmediatamente inferior, a quien solicita servicios, y del nivel inmediatamente superior, a quien devuelve resultados.



El modelo TCP/IP no es dotado en los niveles inferiores como para detallar la auténtica estratificación en niveles: necesitaría tener una capa extra la de red entre niveles de transporte e internet.

Protocolos específicos de un tipo concreto de red, que se sitúan por encima del marco de hardware, pertenecen al nivel de red.

Ejemplos de estos protocolos son el ARP (Protocolo de resolución de direcciones) y el STP (*Spanning Tree Protocol*). De todas formas, estos son protocolos locales, y trabajan por debajo de las capas de Internet. Cierto es que situar ambos grupos, sin mencionar los protocolos que forman parte del nivel de Internet pero se sitúan por encima de los protocolos de Internet, como ICMP, todos en la misma capa puede producir confusión, pero el modelo OSI no llega a ese nivel de complejidad para ser útil como modelo de referencia.

7	**Aplicación**	HTTP, DNS, SMTP, SNMP, FTP, Telnet, SSH y SCP, NFS, RTSP, Feed, Webcal, POP3
6	**Presentación**	XDR, ASN.1, SMB, AFP
5	**Sesión**	TLS, SSH, ISO 8327/CCITT X.225, RPC, NetBIOS, TELNET
4	**Transporte**	TCP, UDP, RTP, SCTP, SPX
3	**Red**	IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IGRP, EIGRP, IPX, DDP
2	**Enlace de datos**	Ethernet, Token Ring, PPP, HDLC, Frame Relay, RDSI, ATM, IEEE 802.11, FDDI
1	**Fisico**	cable, radio, fibra óptica

Normalmente los tres niveles superiores del modelo OSI (Aplicación, Presentación y Sesión) son considerados simplemente como el nivel de aplicación en el conjunto TCP/IP. Como TCP/IP no tiene un nivel de sesión unificado sobre el que los niveles superiores se sostengan, estas funciones son típicamente desempeñadas por las aplicaciones de usuario. La diferencia más notable entre los modelos TCP/IP y OSI es el nivel de Aplicación.

5	**Aplicación**	HTTP, FTP, DNS protocolos de enrutamiento como BGP y RIP, que por varias eazones funcionan sobre TCP y UDP respectivamente, son considerados parte del nivel de red
4	**Transporte**	TCP, UDP, RTP, SCTP protocolos de enrutamiento como OSPF, que funcionen sobre IP, son considerados parte del nivel de red
3	**Internet**	Para TCP/IP este es el Protocolo Internet (IP), protocolos requeridos como ICMP eIGMP funcionan sobre IP, pero todavía se pueden considerar parte del nivel de red, ARP no funciona sobre IP
2	**Enlace**	Ethernet, Token Ring, PPH, HDLC, Frame Relay, RDSI, ATM, IEEE 802.11, FDDI
1	**Fisico**	Medio físico, y técnicas de codificación T1, E1

Nivel Físico: Describe las características físicas de la comunicación, como las convenciones sobre la naturaleza del medio usado para la comunicación como las comunicaciones por cable, fibra óptica o radio, y todo lo relativo a los detalles como los conectores, código de canales y modulación, potencias de señal, longitudes de onda, sincronización y temporización y distancias máximas.

Nivel de Enlace de Datos: Especifica como son transportados los paquetes sobre el nivel físico, incluyendo los delimitadores. Ethernet, por ejemplo, incluye campos en la cabecera de la trama que especifican que máquina o máquinas de la red son las destinatarias de la trama. Ejemplos de protocolos de nivel de enlace de datos son Ethernet, Wireless Ethernet, SLIP, Token Ring y ATM.

Nivel de Internet: Realiza las tareas básicas para conseguir transportar datos desde un origen a un destino. IP puede pasar lo datos a una serie de protocolos superiores, cada uno de esos protocolos es identificado con un número de protocolo IP.

Nivel de Transporte: Soluciona problemas como la fiabilidad y la seguridad de que los datos llegan en el orden correcto. En el conjunto de protocolos TCP/IP, los protocolos de transporte también determinan a qué aplicación van destinados los datos.

Nivel de Aplicación: Es el nivel que los programas más comunes utilizan para comunicarse a través de una red con otros programas. Los procesos que ocurren en este nivel son aplicaciones específicas que pasan los datos al nivel de aplicación en el formato que internamente use el programa y es codificado de acuerdo con un protocolo estándar.

1.3.4 Formato de los Segmentos TCP

Sabemos que en la pila del protocolo TCP/IP, TCP es la capa intermedia entre el protocolo de enlace y la aplicación. Frecuentemente, las aplicaciones necesitan que la comunicación sea fiable y, dado que la capa IP aporta un servicio de datagramas no fiable, TCP añade las funciones necesarias para prestar un servicio que permita que la comunicación entre dos sistemas se efectúe: libre de errores, sin pérdidas y con seguridad.

En el nivel de transporte, los paquetes de bits que constituyen las unidades de datos de protocolo o PDU se llaman segmentos.

+	Bits 0 - 3	4 - 7	8 - 15	16 - 31
0	Puerto Origen			Puerto Destino
32	Número de Secuencia			
64	Número de Acuse de Recibo (ACK)			
96	longitud cabecera TCP	Reservado	Flags	Ventana
128	Suma de Verificación (Checksum)			Puntero Urgente
160	Opciones + Relleno (opcional)			
224	Datos			

Las aplicaciones envían flujos de bytes a la capa TCP para ser enviados a la red TCP divide el flujo de bytes llegado de la aplicación en segmentos de tamaño apropiado (restringida por MTU) y le añade sus cabeceras. Como segundo paso TCP pasa el segmento resultante a la capa IP, donde a través de la red, llega a la capa TCP de la entidad destino. TCP comprueba que ningún segmento se ha perdido dando a cada uno un número de secuencia, que es también usado para asegurarse de que los paquetes han llegado a la entidad destino en el orden correcto. TCP devuelve una confirmación por bytes que han sido recibidos correctamente, un temporizador en la entidad origen del envío causará un timeout si la confirmación no es recibido en un

tiempo razonable, y el paquete será entonces retransmitido. TCP revisa que no haya bytes dañados durante el envío usando un cheksum, es calculado por el emisor en cada paquete antes de ser enviado, y comprobado por el receptor.

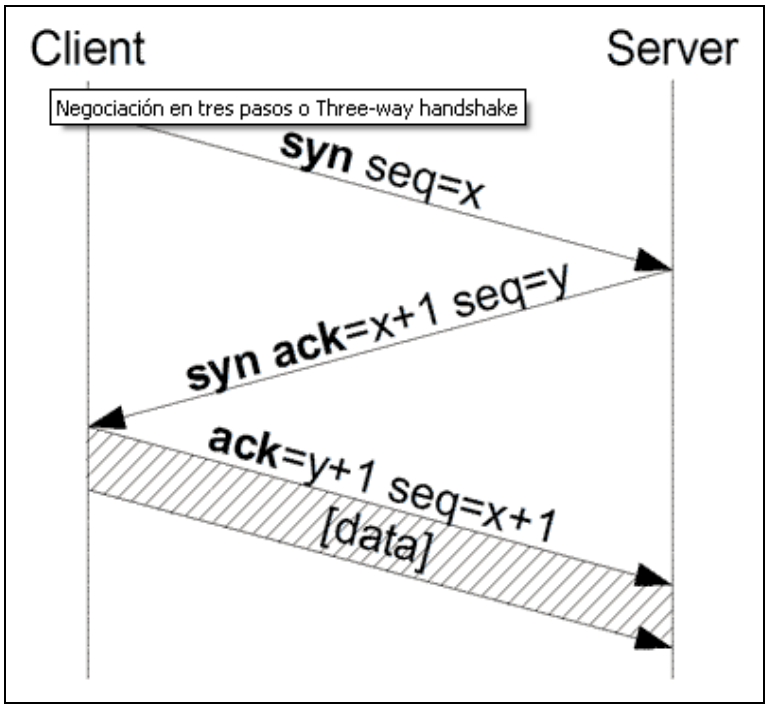
1.3.5 Cómo funciona el protocolo

Para que funciones la conexión, TCP cumple con tres etapas:

- Establecimiento de conexión
- Transferencia de datos
- Y el fin de conexión

Para el establecimiento de la conexión se usa el procedimiento llamado negociación en tres pasos. Y para la desconexión una negociación en cuatro pasos.

Conexión o negociación en tres pasos



1.3.5.1 Establecimiento de la conexión

Al establecer una conexión entre dos entidades finales, una de ellas abre un socket (puerto enlace) en un determinado puerto tcp, y se queda a la espera de escuchar una nueva conexión, llamada esta apertura pasiva, esperando que el lado del servidor determine una conexión. Por lo tanto, el lado del cliente de una conexión realiza una apertura activa, enviando una petición con el parámetro SYN inicial al servidor para dar inicio al proceso de establecimiento de conexión. El lado de servidor respondería a esta petición (SYN) con un paquete SYN/ACK, y finalmente, el cliente debería responderle al servidor con un ACK, dando por hecho el establecimiento de conexión.

1.3.5.2 Transferencia de datos

Varios procedimientos claves determinan la fiabilidad y robustez del protocolo TCP. Uno de los parámetros más importantes es el uso de número de secuencia para ordenar los segmentos TCP recibidos y detectar paquetes duplicados, checksums para detectar errores, y asentimientos y temporizadores para detectar pérdidas y retrasos.

Al momento de establecer la conexión, los números iniciales de secuencia son intercambiados entre las dos entidades TCP. Estos números de secuencia son usados para identificar los datos dentro del flujo de bytes y poder identificar los bytes de los datos de la aplicación. Siempre hay un par de números de secuencia incluidos en todo segmento TCP, referidos al número de secuencia y al número de asentimiento. El emisor se refiere a su propio número de secuencia, mientras que con el número de asentimiento se refiere al número de secuencia del receptor. Para mantener la fiabilidad, un receptor asiente los segmentos TCP indicando que ha recibido una parte del flujo continuo de bytes.

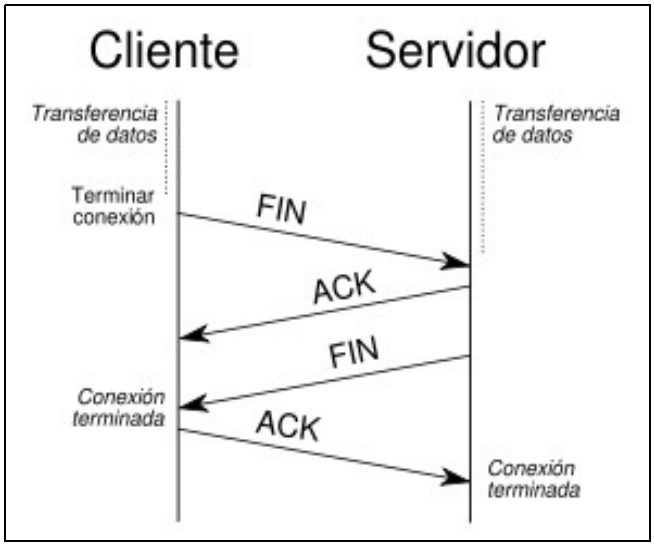
El uso de números de secuencia y asentimiento nos ayudan a pasar los segmentos recibidos en el orden correcto dentro del flujo de bytes al lado receptor.

El checksum de 16 bits, consiste en el complemento a uno de la suma en complemento a uno del contenido de la cabecera y datos del segmento TCP, es calculado por el emisor, e incluido en la transmisión del segmento. El receptor TCP recalcula el checksum sobre las cabeceras y datos recibidos. El complemento es usado para que el receptor no tenga que poner a cero el campo del checksum de la cabecera antes de hacer los cálculos, salvando en algún lugar el valor del checksum recibido. El receptor calcula la suma en complemento a uno con el checksum, y el resultado debe ser igual a 0, asumiendo que el segmento ha llegado intacto y sin errores.

Los asentimientos de los datos enviados o la falta de ellos, son usados por los emisores para interpretar las condiciones de la red entre el emisor y receptor TCP. Los asentimientos más los temporizadores, emisores y receptores pueden alterar el comportamiento del movimiento de datos.

La ventana deslizante, controla que el transmisor mande información dentro de los límites del buffer del receptor y algoritmos de control de flujo, consiguiendo un alto rendimiento y evitando la congestión de la red.

1.3.5.3 Fin de la conexión



La fase de finalización de la conexión usa una negociación en cuatro pasos, terminando la conexión desde cada lado independiente. Cuando uno de los dos extremos de la conexión desea para su mitad de conexión transmite un paquete FIN, que el otro interlocutor asentirá con un ACK. Efectivamente, una desconexión típica requiere un par de segmentos FIN y ACK desde cada lado de la conexión.

Una conexión puede estar "media abierta" en el caso de que uno de los lados la finalice pero el otro no. El lado que ha dado por finalizada la conexión no puede envías más datos pero la otra parte si podrá.

1.3.6 Puertos TCP

TCP usa el número de puerto para identificar a las aplicaciones emisoras y receptoras. Cada lado de la conexión tiene asociado un número de puerto de 16 bits con 65536 posibilidades de puertos posibles. Los puertos son clasificados en tres categorías: bien conocidos, registrados y dinámicos/privados. Los puertos bien conocidos son asignados por la Internet, por LANA (Assigned Numbers Authority),

van del 0 al 1023 y son usados normalmente por el sistema o por procesos con privilegios. Las aplicaciones que usan este tipo de puertos son ejecutadas como servidores y se quedan a la escucha de conexiones como FTP, SSH, Telnet, SMTP y HTTP. Los puertos registrados son normalmente empleados por las aplicaciones de usuario de forma temporal cuando hayan sido registrados por un tercero, van desde 1024 a 49151. Los puertos dinámicos/privados también pueden ser usados por las aplicaciones de usuario, pero este caso es menos común. Los puertos dinámicos/privados no tienen significado fuera de la conexión TCP en la que fueron usados, estos van dinámicos/privados 49152 al 65535.

APLICACIÓN	TCP / UDP	PUERTO RESERVADO	FUNCIONALIDAD
TELNET	TCP	23	Ejecución de procesos en sistemas remotos
FTP	TCP	21 Y 20	Transferencia Archivos
TFTP	UDP	69	Transferencia Simple de Archivos
DNS	TCP / UDP	53	Resolución de nombres para internet
SMTP	TCP	25	Intercambio de mensajes de correo
SNMP	UDP	161 / 162	Protocolo de gestión y monitoreo de dispositivos
HTTP	TCP	80	Transferencia de objetos

1.4 IPTABLES

1.4.1 Historia

La primera pila IP para Linux la desarrolló Ross Biro llamada NET1. Al mismo tiempo, Orest Zborowski y Laurence Culthane trabajaban en la API sockets y en los controladores SLIP. Sin embargo, la verdadera integración de Linux a la red llegó de la mano de Alan Cox con NET-2 y NET-3. Esta última, es la actual pila de protocolos TCP/IP en la que además de Cox participaron muchos otros como Donnald Becker.

El sistema operativo Linux, ha contado con herramientas de filtrado de paquetes (IPFW) incorporadas en su núcleo desde la versión del kernel 1.1.

Este primera versión con filtrado, contaba con una adaptación de la herramienta ipfw del sistema operativo BSD llevada a cabo por Alan Cox en 1994.

El holandés Jos Vos junto a otras personas, mejoró el sistema de filtrado para las series 2.0 del kernel, e introdujo la utilidad de configuración ipfwadm.

Todo comenzó con un proyecto en el año de 1998 dirigido por Rusty Russell, siendo en 1999 el fundador de Netfilter Core Team o Coreteam, un grupo de personas encargadas del desarrollo y mantenimiento del proyecto.

En Linux 2.2, el programa ipchains era el más usado para crear cortafuegos, iptables fue incorporado en marzo del 2000 en el núcleo Linux 2.3. La razón por la cuál se reemplazo iptables por ipchains, es por que este último no puede manipular correctamente los paquetes bajo las normas precisas de una red, e iptables es una modificación que permite la construcción de reglas más precisas y teniendo un mejor aprovechamiento de los recursos.

1.4.2 Cómo funciona Iptables?

Iptables permite al administrador definir las reglas acerca de qué hacer con los paquetes de red, agrupándose la reglas en cadenas ordenadas en forma descendente, o sea, donde la última regla va a ser la última en ejecutarse.

Cada regla especifica qué paquete la cumple por medio del procedimiento match, y un destino que indica qué hacer con el paquete si éste cumple la regla. Cada paquete de red que llega a una computadora o que la envía desde la computadora recorre por lo menos una cadena y cada regla de esa cadena se comprueba con el paquete. Si la regla cumple con el datagrama, el recorrido se detiene y el destino de la regla dicta lo que se debe hacer con el paquete.

En iptables, las reglas se agrupan en cadenas. Una cadena es un conjunto de reglas para paquetes IP, que determinan lo que se debe hacer con ellos. Cada regla puede desechar el paquete de la cadena, con el cual otras cadenas no serán consideradas. Una cadena puede contener un enlace a otra cadena: si el paquete pasa a través de esa cadena entera o si cumple una regla de destino de retorno, va a continuar en la primera cadena. No hay un limite respecto de cuán anidadas pueden estar las cadenas. Hay tres cadenas básicas ENTRADA, SALIDA Y REENVÍO y el usuario puede crear tantas como desee. Una regla puede ser simplemente un puntero a una cadena.

1.4.3 Tablas

Hay tres tablas, cada una de las cuales contiene ciertas cadenas predefinidas. Es posible crear nuevas tablas mediante módulos de extensión. El administrador puede crear y eliminar cadenas definidas por usuarios dentro de cualquier tabla. Inicialmente todas las cadenas están vacías y tienen una política de destino que permite que todos los paquetes pasen sin ser bloqueados o alterados.

4.1.1 Filter Table o Tabla de Filtros

Es la responsable del filtrado de paquetes, es decir, de bloquear o permitir que un paquete continúe su camino. Todos los paquetes pasan a través de la tabla de filtros. Contiene las siguientes cadenas:

- **Input:** Todo el tráfico entrante, se la llama algunas veces LOCAL_INPUT o ENTRADA_LOCAL.
- **Output:** Todo el tráfico saliente, se lo llama a veces LOCAL_OUTPUT o SALIDA_LOCAL.
- **Forward:** Todos los paquetes que pasan por este sistema para ser encaminados a su destino recorren esta cadena

1.4.3.2 NAT Table o Tabla de Traducción de Direcciones de Red

Esta tabla es la responsable de configurar las reglas de reescritura de direcciones o de puertos de los paquetes. Sirve básicamente para hacer que otros ordenadores se conecten a través del nuestro a una serie de servicios pero con nuestra IP, pareciendo que esas conexiones vienen de nuestro equipo. Contiene las siguientes cadenas redefinidas:

- **Prerouting:** Cadena de Preruteo. Los paquetes entrantes pasan a través de esta cadena antes de que se consulte la tabla de ruteo local, principalmente para DNAT o traducción de direcciones de red de origen.
- **Postrouting:** Cadena de Posruteo. Los paquetes salientes pasan por esta cadena después de haberse tomado la decisión del ruteo, principalmente SNAT o traducción de direcciones de red de origen.
- **Output:** Cadena de Salida. Permite hacer un DNAT limitado en paquetes generados localmente.

1.4.3.3 Mangle Table o Tabla de destrozo

Esta tabla es la responsable de ajustar las opciones de los paquetes, como por ejemplo la calidad de servicio. Todos los paquetes pasan por esta tabla. Debido a

que está diseñada para efectos avanzados, contiene todas las cadenas predefinidas posibles:

- **Prerouting:** Todos los paquetes que logran entrar a este sistema, antes de que el ruteo decida si el paquete debe ser reenviado o si tiene destino local
- **Input:** Todos los paquetes destinados para este sistema pasan a través de esta cadena.
- **Forward:** Todos los paquetes que exactamente pasan por este sistema pasan a través de esta cadena.
- **Output:** Todos los paquetes creados en este sistema pasan a través de esta cadena.
- **Postrouting:** Todos los paquetes que abandonan este sistema pasan a través de esta cadena.

Teniendo el módulo ipt_conntrack disponible, dispondremos de herramientas para controlar el estado de la conexión:

- **New:** Nuevo paquete que viene hacia nosotros.
- **Related:** Paquetes nuevos pero que ya están relacionados con una conexión existente.

Cuando usamos un ftp se abren varias conexiones para poder bajar correctamente lo que necesitamos. Si no lo activamos, sólo sería posible la primera conexión y los demás paquetes, aunque relacionados con la primera conexión, no se dejarían pasar y la transferencia se interrumpirá.

- **Established:** Paquetes asociados a una conexión nueva.
- **Invalid:** Todos los demás paquetes, que no coincidan con ninguno de los estados descritos.

1.4.4 Destinos de reglas

El destino de una regla puede ser el nombre de una cadena definida por el usuario o uno de los destinos ya incorporados ACCEPT, DROP, QUEUE, o RETURN. Cuando un destino es el nombre de una cadena definida por el usuario, al paquete se lo dirige a esa cadena para que sea procesado. Si el paquete consigue atravesar la cadena definida por el usuario sin ninguna de las reglas de esa cadena actúe sobre él, el procesamiento del paquete continúa donde había quedado en la cadena actual. Estos llamados entre cadenas se pueden anidar hasta cualquier nivel deseado.

Existen los siguientes destinos ya incorporados:

- **Accept (aceptar)**

Este destino hace que el netfilter acepte el paquete. El significado de esto depende de cuál sea la cadena realizando esta aceptación. Efectivamente, a un paquete que se lo acepta en la cadena de ENTRADA se le permite ser recibido por la terminal (host), a un paquete que se lo acepta en la cadena de SALIDA se le permite abandonar la terminal y a un paquete que se lo acepta en la cadena de REDIRECCIÓN se le permite ser ruteado a través de la terminal.

- **Drop (descartar)**

Este destino hace que netfilter descarte el paquete sin ningún otro tipo de procesamiento. El paquete simplemente desaparece sin indicación de que fue descartado al ser entregado a la terminal de envío o a una aplicación. Esto se le refleja al que envía, a menudo, como un communication timeout (alcance del máximo tiempo de espera en la comunicación), lo que puede causar confusión (aunque el descarte de paquetes entrantes no deseados se considera a veces una buena política de seguridad, pues no da ni siquiera el indicio a un posible atacante de que la terminal existe).

- **Queue (encolar)**

Este destino hace que el paquete sea enviado a una cola en el espacio de usuario. Una aplicación puede usar la biblioteca libipq, también parte del proyecto netfilter/iptables, para alterar el paquete. Si no hay ninguna aplicación que lea la cola, este destino es equivalente a DROP.

- **Return (retorno)**

Hace que el paquete en cuestión deje de circular por la cadena en cuya regla se ejecutó el destino RETURN. Si dicha cadena es una subcadena de otra, el paquete continuará por la cadena superior como si nada hubiera pasado. Si por el contrario la cadena es una cadena principal (por ejemplo la cadena INPUT), al paquete se le aplicará la política por efecto de la cadena en cuestión (ACCEPT, DROP o similar).

Hay muchos destinos de extensión disponibles. Algunas de los más comunes son:

- **Reject (rechazo)**

Este destino tiene el mismo efecto que DROP, salvo que envía un paquete de error a quien envió originalmente. Se usa principalmente en las cadenas de ENTRADA y de REDIRECCIÓN de la tabla de filtrado. El tipo de paquete se puede controlar a través del parámetro *"--reject-with*". Un paquete de rechazo puede indicar explícitamente que la conexión ha sido filtrada (un paquete ICMP filtrado administrativamente por conexión), aunque la mayoría de los usuarios prefieren que el paquete indique simplemente que la computadora no acepta ese tipo de conexión (tal paquete será un paquete tcp-reset para conexiones TCP denegadas, un icmp-port-unreachable para sesiones UDP denegadas o un icmp-protocol-unreachable para paquetes no TCP y UDP). Si el parámetro "- - reject-with" no se especifica, el paquete de rechazo por defecto es siempre icmp-port-unreachable.

- **Log (bitácora)**

Este destino lleva un log o bitácora del paquete. Puede usarse en cualquier cadena en cualquier tabla, y muchas veces se usa para debuggear (análisis de fallos, como ser la verificación de qué paquetes están siendo descartados)

- **Ulog**

Este destino lleva un log o bitácora del paquete, pero no de la misma manera que el destino LOG. El destino LOG le envía información al log del núcleo, pero ULOG hace multidifusión de los paquetes que matchean esta regla a través de un socket netlink, de manera que programas del espacio de usuario puedan recibir este paquete conectándose al socket.

- **Dnat**

Este destino hace que la dirección (y opcionalmente el puerto) de destino del paquete sean reescritos para traducción de dirección de la red. Mediante la opción "--to-destination" debe indicarse el destino a usar. Esto es válido solamente en las cadenas de SALIDA y PRERUTEO dentro de la tabla de NAT. Esta decisión se recuerda para todos los paquetes futuros que pertenecen a la misma conexión y las respuestas tendrán su dirección y puerto de origen cambiados al original (es decir, la inversa de este paquete).

- **Snat**

Este destino hace que la dirección (y opcionalmente el puerto) de origen del paquete sean reescritos para la traducción de dirección de red. Mediante la opción "- - to-source" debe indicarse el origen a usar. Esto es válido solamente en la cadena de POSRUTEO dentro de la tabla de NAT y, como DNAT, se recuerda para todos los paquetes que pertenecen a la misma conexión

- **Masquerade**

Este es una forma especial, restringida de SNAT para direcciones IP dinámicas, como las que proveen la mayoría de los proveedores de servicios de Internet (ISPs) para modems o línea de abonado digital (DSL). En vez de cambiar la regla de SNAT cada vez que la dirección IP cambia, se calcula la dirección IP de origen a la cual hacer NAT fijándose en la dirección IP de la interfaz de salida cuando un paquete coincide con esta regla. Adicionalmente, recuerda cuales conexiones usan MASQUERADE y si la dirección de la interfaz cambia (por ejemplo, por reconectarse al ISP), todas las conexiones que hacen NAT a la dirección vieja se olvidan.

1.4.5 Opciones Comunes

Iptables es un aplicativo del espacio de usuario que le permite a un administrador de sistema configurar las tablas, cadenas y reglas de netfilter. Debido a que iptables requiere privilegios elevados para operar, el único que puede ejecutarlo es el superusuario. En la mayoría de los sistemas Linux, iptables está instalado como /sbin/iptables. La sintaxis detallada del comando iptables está documentada en su manual, la cual puede verse tipeando el comando “man iptables” desde la línea de comandos.

En cada una de las formas de invocación de iptables que se muestra a continuación, las siguientes opciones comunes están disponibles:

-t tabla

Hace que el comando se aplique a la tabla específica. Si esta opción se omite, el comando se aplica a la tabla filter por defecto.

-v

Produce una salida con detalles.

-n

Produce una salida numérica (es decir, números de puerto en lugar de nombres de servicio y direcciones IP en lugar de nombres de dominio).

--line-numbers

Cuando se listan reglas, agrega números de línea al comienzo de cada regla, correspondientes a la posición de esa regla en su cadena.

1.4.6 Especificaciones de las reglas

La mayoría de las formas de comandos deiptables requieren que seles indiquen una especificación de reglas, que es usada para machear un subconjunto particular del tráfico de paquetes de red procesados por una cadena. La especificación de regla incluye también un destino que especifica qué hacer con paquetes que son matcheados por la regla.

Las siguientes opciones se usan para crear especificaciones de reglas:

-j (destino) o --jump (destino)

Especifica el destino de una regla. El destino es el nombre de una cadena definida por el usuario (creada usando la opción –N, uno de los destinos ya incorporados, ACCEPT, DROP, QUEUE, o RETURN, o un destino de extensión, como REJECT, LOG, DNAT, o SNAT). Si esta opción es omitida en una regla, entonces el matcheo de la regla no tendrá efecto en el destino de un paquete, pero los contadores en la regla se incrementarán.

-i [!] (in-interface) o –in-interface [!] (in-interface)

Nombre de una interfaz a través de la cual un paquete va a ser recibido (solo para paquetes entrando en las cadenas de INPUT, FORWARD Y PREROUTING). Cuando

se usa el argumento '!' antes del nombre de la interfaz, el significado se invierte. Si el nombre de la interfaz termina con '+', entonces cualquier interfaz que comience con este nombre será matcheada. Si esta opción se omite, se matcheará todo nombre de interfaz.

-o [!] (out-interface) o –out-interface [!] (out-interface)

Nombre de una interfaz a través de la cual un paquete va a ser enviada (para paquetes entrando en las cadenas de FORWARD, OUTPUT y PREROUTING). Cuando se usa el argumento '!' antes del nombre de la interfaz, el significado se invierte. Si el nombre de la interfaz termina con '+', entonces cualquier interfaz que comience con este nombre será matcheada. Si esta opción se omite, se matcheará todo nombre de interfaz.

-p [!] (protocol) o –protocol [!] (protocol)

Matchea paquetes del nombre de protocolo especifico. Si '!' precede el nombre de protocolo, se matchean todos los paquetes que no son el protocolo especifico. Nombres de protocolos válidos son icmp, udp, tcp, etc. Una lista de todos los protocolos válidos puede encontrarse en el archivo /etc/protocols.

-s [!] (origen [/prefijo]) o –source [!] (origen [/prefijo])

Matchea paquetes IP viniendo de la dirección de origen especificada. La dirección de origen puede ser una dirección IP, una dirección IP con un prefijo de red asociado, o un nombre de terminal (hostname). Si '!' precede al origen, se matchean todos los paquetes que no vienen del origen especificado.

-d [!] (destino [/prefijo]) o –destination [!] (destino [/prefijo])

Matchea paquetes IP yendo a la dirección de destino especificada. La dirección de destino puede ser una dirección IP, una dirección IP con un prefijo de red asociado, o un nombre de terminal (hostname). Si '!' precede al origen, se matchean todos los paquetes que no van al destino especificado.

--dport[!] (puerto[:puerto]) o --destination-port[!] (puerto[:puerto])

Matchea paquetes TCP o UDP (dependiendo del argumento a la opción –p) destinados a los puertos o rango de puertos (cuando se usa la forma puerto:puerto) especificados. Si '!' precede la especificación de puertos, se matchean todos los paquetes TCP o UDP que no están destinados a los puertos o rango de puertos especificados.

--source-port [!] (puerto [:puerto]) o --sport[!] (puerto [:puerto]

Matchea paquetes TCP o UDP (dependiendo del argumento a la opción -p) que vienen de los puertos o rango de puertos (cuando se usa la forma puerto:puerto) especificados. Si '!' precede la especificación de puertos, se matchean todos los paquetes TCP o UDP que no vienen de los puertos o rango de puertos especificados.

--tcp-flags [!] mask comp

Matchea paquetes TCP que tienen marcadas o desmarcadas ciertas banderas del protocolo TCP. El primer argumento especifica las banderas a examinar en cada paquete TCP, escritas en una lista separada por comas (no se permiten espacios). El segundo argumento es otra lista separada por comas de banderas que deben estar marcadas dentro de las que se debe examinar. Estas banderas son: SYN, ACK, FIN, RST, URG, PSH, ALL, y NONE. Por lo tanto, la opción "—tcp-flags SYN, ACK, FIN, RST SYN" solo va a matchear paquetes con la bandera SYN marcada y las banderas ACK, FIN y RST desmarcadas.

[!] --syn

Matchea paquetes TCP que tienen la bandera SYN marcada y las banderas ACK, FIN y RST desmarcadas. Estos paquetes son los que se usan para iniciar conexiones TCP. Al bloquear tales paquetes en la cadena de INPUT, se previenen conexiones TCP entrantes, pero conexiones TCP salientes no serán afectadas.. esta opción puede combinarse con otras, como --source, para bloquear o dejar pasar conexiones TCP entrantes solo de ciertas terminales o redes. Esta opción es equivalente a "—tcp-flags SYN, RST, ACK, SYN". Si '!' precede a --syn, el significado de la opción se invierte.

1.4.7 Sintaxis para reglas de Iptables

El comando iptables tiene las siguientes formas de invocación:

- Esta forma de invocación del comando agrega (-A o --append) o elimina (-D o --delete) una regla de la cadena especificada.

```
iptables { -A | –append | -D | --delete} cadena especificación de regla [opciones]
```

Para agregar una regla a la cadena de INPUT en la tabla filter (la tabla por defecto cuando la opción –t no se especifica) que descarte todos los paquetes UDP, usamos este comando:

```
iptables -A INPUT -p udp -j DROP
```

Para borrar la regla agregada por el comando anterior, usamos este comando:

```
iptables -D INPUT -p udp -j DROP
```

Este comando borra la primera regla de la cadena INPUT que matchea la especificación de regla "-p udp –j DROP". Si hay varias reglas idénticas en la cadena, solo se borra la primera regla que matchea.

- Esta forma de invocación del comando reemplaza (-R o --replace) una regla existente o inserta (-I o --insert) una regla nueva en la cadena especificada.

```
iptables {-R | --replace | -I | --insert} numregla especificación-de-regla [opciones]
```

Para reemplazar la cuarta regla en la cadena INPUT por una regla que descarte todos los paquetes ICMP, usamos este comando:

```
iptables –R INPUT 4 –p icmp –j DROP
```

Para insertar una regla nueva en el Segundo lugar de la cadena de OUTPUT que descarte todo el tráfico TCP dirigido al puerto 80 en cualquier terminal, usamos este comando:

```
iptables –I OUTPUT 2 –p tcp –dport 80 –j DROP
```

- Esta forma de invocación del comando elimina una regla del índice numérico especificado en la cadena especificada. Las reglas se numeran comenzando desde 1.

```
iptables {-D | --delete} cadena numregla [opciones]
```

Para eliminar la tercera regla de la cadena FORWARD, usamos este comando:

```
iptables  -D     FORWARD 3
```

- Esta forma de invocación del comando se usa para listar las reglas en una cadena (-L o --list), eliminar todas las reglas de una cadena (-F o --flush), o para poner en cero el byte y los contadores de paquete de una cadena (-Z o --zero). Si no se especifica ninguna cadena, la operación se realiza para todas las cadenas.

```
iptables {-L | –list | -F | --flush | -Z | --zero} [cadena] [opciones]
```

Para listar las reglas en la cadena de OUTPUT, usamos este comando:

```
iptables  -L  OUTPUT
```

Para eliminar todas las cadenas, usamos este comando:

```
iptables  -F
```

Para poner en cero el byte y los contadores de paquetes de la cadena de PREROUTING en la tabla nat, usamos este comando:

```
iptables -t nat  -Z  PREROUTING
```

- La siguiente forma de invocación del comando se usa para crear (-N o --new-chain) una cadena nueva definida por el usuario.

```
iptables {-N | --new-chain} [cadena]
```

- Para eliminar (-X o *--delete-chain*) una cadena definida por el usuario existente. Si no se especifica ninguna cadena con las opciones –X o –delete-chain, se elimina todas las cadenas definidas por el usuario. No es posible eliminar cadenas ya incorporadas, como ser las cadenas de INPUT y OUTPUT en la tabla filter.

```
iptables {-X | –delete-chain} [cadena]
```

- Esta forma de invocación del comando se usa para especificar la política de destino para una cadena.

```
iptables {-P | –policy} [cadena] [destino]
```

Para especificar la política de destino para la cadena de INPUT en DROP, usamos este comando:

```
iptables -P INPUT DROP
```

- Esta forma de invocación del comando se usa para renombrar una cadena definida por el usuario.

```
iptables {-E | –rename-chain} cadena-vieja cadena-nuevo
```

2. CAPITULO DOS: CONTROL DE ANCHO DE BANDA

2.1. INTRODUCCION

En términos digitales, definimos que el ancho de banda es la capacidad de una línea para transmitir información, compartida frecuentemente por varios usuarios.

El ancho de banda se lo compara como un teleférico con rapidez[1], habiendo varios paquetes por subirse al telesilla, y creando una cola de acceso. Hay que tener en cuenta, que hay paquetes que permiten a otros saltarse la cola por ser demasiados lentos al subir al teleférico.

Se cree que cuando más ancho de banda, hay más rapidez de acceso. Pero no es del todo esto cierto, sino se llega a apreciar que hay más usuarios compartiendo este canal de acceso, y que es diferente cuánto de ancho de banda necesitamos a qué línea necesitaremos como mínimo aunque ambas van ligadas.

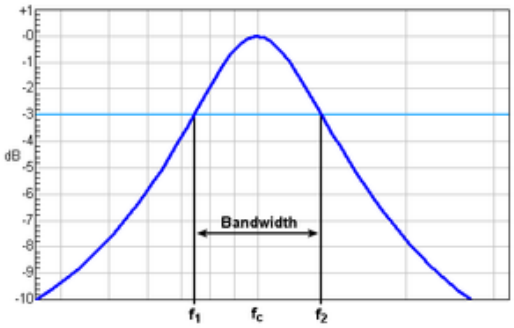
Este documento especificará las diversas herramientas que sirven para el control de ancho de banda y el filtrado de paquetes.

2.2. ANCHO DE BANDA

2.2.1 Señales Analógicas

El ancho de banda en las señales analógicas es la anchura del rango de frecuencias en el que se concentra la mayor parte de la potencia de la señal, esta es medida en hercios. Puede ser calculado a partir de una señal temporal mediante el análisis de Fourier. También son llamadas frecuencias efectivas las pertenecientes a este rango.

[1] "Analogía original de Nicholas Negroponte en su libro El mundo digital"



Así, el ancho de banda de un filtro es la diferencia entre las frecuencias en las que su atenuación al pasar a través de filtro se mantiene igual o inferior a 3 dB comparada con la frecuencia central de pico (fc) en la figura expuesta.

La frecuencia es la magnitud física que mide las veces por unidad de tiempo en que se repite un ciclo de una señal periódica. Una frecuencia tiene un ancho de banda mínimo. En general, si la señal periódica tiene componentes en varias frecuencias, su ancho de banda es mayor, y su variación temporal depende de sus componentes frecuenciales.

El uso común del ancho de banda digital a la cantidad de datos que se pueden transmitir en una unidad de tiempo. Por ejemplo, una línea ADSL de 256 kbps puede enviar 256000 bits por segundo. Esto es en realidad la tasa de transferencia máxima permitida por el sistema, que depende del ancho de banda analógico, de la potencia de la señal, de la potencia de ruido y de la codificación de canal.

2.2.2 Señales Digitales

En sistemas digitales, el ancho de banda digital es la cantidad de datos que pueden ser transportados por algún medio en un determinado período de tiempo. Por lo tanto a mayor ancho de banda, mayor transferencia de datos por unidad de tiempo (mayor velocidad).

Algunas de las variantes de los servicios de línea de abonado digital DSL, son de banda ancha en el sentido en que la información se envía sobre un canal y la voz por otro canal, pero compartiendo el mismo par de cables. Los modems analógicos que operan con velocidades mayores a 600 bps también son técnicamente banda ancha, pues obtiene velocidades de transmisión efectiva mayores usando muchos canales en donde la velocidad de cada canal se limita a 600 baudios. Pro ejemplo, un modem de 2400 bps usa cuatro de 600 baudios. Este método de transmisión contrasta con la transmisión en banda base, en donde un tipo de señal usa todo el ancho de banda del medio de transmisión, como por ejemplo Ethernet 100BASE-T.

2.1.1. Multiplexación

Multiplexar, es tener acceso múltiple, haciendo que las comunicaciones pueden utilizar distintos canales físicos simultáneamente. Tales canales pueden distinguirse uno de otro por estar separados en: tiempo (multiplexación por división de tiempo o TDM), frecuencia de portadora (multiplexación por división de frecuencia, FDM o WDM), o por código (multiplexación por división de código, CDMA). Cada canal que toma parte en la multiplexación es por definición de banda estrecha.

2.3. CONFIGURACIONES DE SQUID

2.3.1 Introducción

Hay un número cada vez mayor de empresas que utilizan Squid para salvar a su web en Internet de tráfico, mejorar el rendimiento, ofrecer mayor rapidez de navegación a sus clientes finales y proporcionar contenido estático, dinámico y flujo de millones de usuarios de Internet en todo el mundo.

Muchos utilizamos Squid sin siquiera saberlo. Algunas empresas han incorporado dispositivos cortafuegos Squid en casas u oficinas, otros usan Squid en gran escala como Web Proxy para acelerar la línea telefónica de banda ancha y acceso a Internet. Squid se está utilizando cada vez más en entrega de contenido para ofrecer arquitecturas estática y flujo de vídeo / audio a los usuarios de Internet en todo el mundo.

Squid ofrece un control de acceso, autorización y registro de medio ambiente para desarrollar un Web Proxy y el contenido al servicio de aplicaciones.

2.3.2 Historia

Squid se basa en el *Harvest Cache Daemon* desarrollado a principios de los 90's. Es una de los dos bifurcaciones de código después a la terminación del proyecto. La otra bifurcación se convirtió en NetApp del Netcache.

El proyecto de Squid fue financiado por una subvención de NSF (NCR-9796082) que abarca la investigación en tecnologías de almacenamiento caché. La financiación corrió ircache a cabo de unos años más tarde y el proyecto squid continuó a través de donaciones voluntarias y la inversión comercial de manera ocasional.

Actualmente, Squid se está desarrollando por varias personas que donan su tiempo y esfuerzo a la creación actual y la siguiente generación de almacenamiento en caché de contenido y tecnologías de entrega.

2.3.3 Servidor Intermediario (Proxy)

Un Servidor Proxy se define como una computadora o dispositivo que ofrece un servicio de red que consiste en permitir a los clientes realizar conexiones de red indirectas hacia otros servicios de red.

Siendo el cliente quien se conecta hacia el Servidor Intermediario, solicita una conexión, fichero u otro recurso disponible en un servidor distinto. El servidor intermediario proporciona el recurso ya sea conectándose hacia el servidor especificado o sirviendo éste desde un caché. En algunos casos el Servidor Intermediario puede alterar la solicitud del cliente o bien la respuesta del servidor para diversos propósitos.

Un Proxy trabaja generalmente como un muro cortafuegos a Nivel de Red, filtrando paquetes con iptables, o bien a Nivel de Aplicación, controlando diversos servicios. Además una aplicación común de los Proxies es funcionar como caché de contenido de Red, permitiendo a los clientes de la red local acceder hacia éstos de forma más rápida y confiable.

Los Servidores Intermediarios para contenido de Red (Web Proxies) también pueden actuar como filtros del contenido servidor.

2.3.4 Squid

Squid es un Servidor Intermediario o Proxy de alto desempeño que se ha venido desarrollando desde hace varios años y hoy en día es muy popular. Es muy

confiable, robusto y versátil y se distribuye bajo términos de la Licencia Pública General GNU.

Squid puede funcionar como Proxy y caché de contenido de Red para protocolos HTTP, FTP, GOPHER y WAIS, Proxy de SSl, caché transparente, WWCP, aceleración HTTP, caché de consultas DNS y otras muchas más como filtración de contenido y control de acceso por IP y por usuario.

Squid consiste de un programa principal como servidor, un programa para búsqueda en servidores DNS, programas opcionales para reescribir solicitudes y realizar autenticación y algunas herramientas para administración y clientes. Al iniciar Squid de origen a un número configurable de procesos de búsqueda en servidores DNS, cada uno de los cuales realiza búsqueda única en servidores DNS, reduciendo la cantidad de tiempo de espera para las búsquedas en servidores DNS.

3.1.1. Algoritmos de caché utilizados por Squid

LRU (Least Recently Used)

En este algoritmo los objetos que no han sido accedidos en mucho tiempo son eliminados primero, manteniendo siempre en el caché a los objetos más recientemente solicitados. Ésta política es la utilizada por Squid de modo predefinido.

LFUDA (Least Frequently Used with Dynamic Aging)

En este algoritmo los objetos más solicitados permanecen en el caché sin importar su tamaño optimizando la eficiencia (hit rate) por octetos (Bytes) a expensas de la

eficiencia misma, de modo que un objeto grande que se solicite con mayor frecuencia impedirá que se pueda hacer caché de objetos pequeños que se soliciten con menor frecuencia.

GDSF (GreedyDual Size Frequency)

Es el algoritmo sobre el cual se basa GDSF. Optimiza la eficiencia (hit rate) por objeto manteniendo en el caché los objetos pequeños más frecuentemente solicitados de modo que hay mejores posibilidades de lograr respuesta a una solicitud (hit). Tiene una eficiencia por octetos (Bytes) menos que el algoritmo LFUDA debido a que descarta del caché objetos grandes que sean solicitado con frecuencia.

3.1.2. Características de Squid

Proxy y Caché de HTTP, FTP y otras URL

Squid proporciona un servicio de Proxy que soporta peticiones HTTP, HTTPS y FTP a equipos que necesitan acceder a Internet y a su vez provee la funcionalidad de caché especializado en el cual almacena de forma local las páginas consultadas recientemente por los usuarios. De esta forma, incrementa la rapidez de acceso a los servidores de información Web y FTP que se encuentre fuera de la red interna.

Proxy para SSL

Squid también es compatible con SSL (Secure Socket Layer) con lo que también acelera las transacciones cifradas, y es capaz de ser configurado con amplios controles de acceso sobre las peticiones de usuarios.

Jerarquías de caché

Squid puede formar parte de una jerarquía de caches. Diversos Proxies trabajan conjuntamente sirviendo las peticiones de las páginas. Un navegador solicita siempre

las páginas a un solo Proxy, si este no tiene la página en la caché hace peticiones a sus hermanos, que si tampoco las tienen las hacen a sus padres. Estas peticiones se pueden hacer mediante dos protocolos: HTTP e ICMP.

ICP, HTCP, CARP y Caché Digests

Squid sigue los protocolos ICP, HTCP, CARP y Caché Digests que tienen como objetivo permitir a un Proxy "preguntar" a otros proxies caché si poseen almacenado un recurso determinado.

Caché Transparente

Squid puede ser configurado para ser usado como Proxy transparente de manera que las conexiones son enrutadas dentro del Proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia. De modo predefinido Squid utiliza el puerto 3128 para atender peticiones, sin embargo se puede especificar que lo haga en cualquier otro puerto disponible o bien que lo haga en varios puertos disponibles a la vez.

WCCP

A partir de la versión 2.3 Squid implementa WCCP (Web Caché Control Protocol). Permite interceptar y redirigir el tráfico que recibe un router hacia uno o más proxies caché, haciendo control de la conectividad de los mismos. Además permite que uno de los proxies caché designado puede determinar como distribuir el tráfico redirigido a lo largo de todo el array de proxies caché.

Control de Acceso

Ofrece la posibilidad de establecer reglas de control de acceso. Esto permite establecer políticas de acceso en forma centralizada, simplificando la administración de una red.

Aceleración de Servidores HTTP

Cuando un usuario hace petición hacia un objeto en Internet, este es almacenado en el caché, si otro usuario hace petición hacia el mismo objeto, y este no ha sufrido modificación alguna desde que lo accedió el usuario anterior, Squid mostrará el que ya se encuentra en el caché en lugar de volver a descargarlo desde Internet. Esta función permite navegar rápidamente cuando los objetos ya están en el caché y además optimiza enormemente la utilización del ancho de banda.

SNMP

Squid permite activar el protocolo SNMP, este proporciona un método simple de administración de red, que permite supervisar, analizar y comunicar información de estado entre una gran variedad de máquinas, pudiendo detectar problemas y proporcionar mensajes de estados.

Caché de resolución DNS

Squid está compuesto también por el programa *dnsserver*, que se encarga de la búsqueda de nombres de dominio. Cuando se ejecuta Squid, produce un número configurable de procesos *dnsserver*, y cada uno de ellos realiza su propia búsqueda en DNS. De este modo, se reduce la cantidad de tiempo que la caché debe esperar a estas búsquedas DNS.

2.3.5 Configuraciones Squid

2.3.5.1 Equipamiento y configuraciones necesarias

Para poder llevar a cabo los procedimientos descritos en este tutorial, se necesita tener instalado lo siguiente:

▪El paquete squid-2.6.STABLE, se recomienda utilizar las versiones estables más recientes de todo sustento lógico que vaya a ser instalado para realizar los procedimientos descritos en este tutorial.

▪Además debemos instalar las dependencias del squid.

▪Iptables-1.2.4

▪Kernel-2.4.9

2.3.5.2 Instalación del software necesario

Regularmente Squid no se instala de manera predeterminada a menos que especifique lo contrario durante la instalación del sistema operativo, sin embargo viene incluido en casi todas las distribuciones actuales.

Instalación a través de yum

Para un sistema con Centos o White box Enterprise Linux 3 o versiones posteriores, utilice lo siguiente y se instalará todo lo necesario junto con sus dependencias:

```
yum –y install squid httpd
```

Instalación a través de rpm

```
mount /mnt/cdrom
rpm –uvh /mnt/cdrom/*/RPMS/squid-*.i386.rpm
eject
```

2.3.5.3 Otros componentes necesarios

El mandato iptables se utilizará para generar las reglas necesarias para el guión de Enmascaramiento de IP. Si cuenta con un sistema con Centos o White Box Enterprise Linux 3 o versiones posteriores, utilice lo siguiente para actualizar el núcleo del sistema operativo e iptables si acaso fuera necesario:

```
yum –y update kernel iptables
```

es importante tener actualizado el kernel por diversas cuestiones de seguridad. No es recomendable utilizar versiones del kernel anteriores a la 2.4.9. en el manual "Cómo actualizar el Kernel a partir de paquetes RPM®" se describe a detalle lo necesario.

2.3.5.4 Configuración Básica de Squid

Squid utiliza el fichero de configuración localizado en */etc/squid/squid.conf*, y se puede trabajar sobre este utilizando su editor de texto preferido.

Existen un gran número de parámetros, de los cuales recomendamos configurar los siguientes:

- http_port
- cache_men
- ftp_user
- ftp_passive
- cache_dir
- Al menos una Lista de Control de Acceso (ACL's)
- Al menos una Regla de Control de Acceso
- cache_mgr
- httpd_accel_host
- httpd_accel_port
- httpd_accel_with_proxy

Parámetro http_port

Squid por defecto utilizará el puerto 3128 para atender peticiones, sin embargo se puede especificar que lo haga en cualquier otro puerto o bien que lo haga en varios puertos a la vez.

En el caso de un Proxy Transparente, regularmente se utilizará el puerto 80 y se valdrá del re-direccionamiento de peticiones de modo tal que no habrá necesidad alguna de modificar la configuración de los navegadores Web para utilizar el servidor Proxy, bastará con utilizar como puerta de enlace al servidor. Es importante recordar que los servidores Web, como Apache, también utilizan dicho puerto, por lo que será necesario reconfigurar el servidor Web para utilizar otro puerto disponible, o bien desinstalar o deshabilitar el servidor Web.

Hoy en día ya no es del todo práctico el utilizar un Proxy Transparente, a menos que se trate de un servicio Café Internet u oficina pequeña, siendo que uno de los principales problemas con los que lidian los administradores es el mal uso y/o abuso del acceso a Internet por parte del personal. Es por esto que puede resultar más conveniente configurar un servidor Proxy con restricciones por contraseña, lo cual no puede hacerse con un Proxy Transparente, debido a que se requiere un diálogo de nombre de usuario y contraseña.

Regularmente algunos programas utilizados comúnmente por los usuarios suelen tener por defecto el puerto 8080 –servicio de cacheo www- para utilizarse al configurar que servidor Proxy utilizar. Si queremos aprovechar esto a nuestro favor y ahorrarnos el tener que dar explicaciones innecesarias al usuario, podemos especificar que Squid escuche peticiones en dicho puerto también. Siendo así localice la sección de definición de http_port y especifique:

```
# You may specify multiple socket addresses on multiple lines.
# Default: http_port 3128
http_port 3128
http_port 8080
```

Parámetro cache_mem

El parámetro cache_mem establece la cantidad ideal de memoria para lo siguiente:

- Objetos de tránsito
- Objetos Hot
- Objetos negativamente almacenados en el caché

Los datos de estos objetos se almacenan en bloques de 4Kb. El parámetro *cache_mem* especifica un límite máximo en el tamaño total de bloques acomodados, donde los objetos en tránsito tienen mayor prioridad. Sin embargo los objetos Hot y aquellos negativamente almacenados en el caché podrán utilizar la memoria no utilizada hasta que esta sea requerida. De ser necesario si un objeto en tránsito es mayor a la cantidad de memoria especificada, Squid excederá lo que sea necesario para satisfacer la petición.

Por defecto se establecerá 8MB. Puede especificarse una cantidad mayor si así se considera necesario, dependiendo esto de los hábitos de los usuarios o necesidades establecidas por el administrador.

Si se posee un servidor con al menos 128MB de RAM, establezca 16MB como valor para este parámetro:

```
cache_mem 16 MB
```

Parámetro cache_dir

Este parámetro se utiliza para establecer que tamaño se desea que tenga el caché en el disco duro para Squid. Para entender esto un poco mejor, responda a esta pregunta: ¿Cuánto desea almacenar de Internet en el disco duro? Por defecto Squid utilizará un caché de 100MB, de modo tal que encontrará la siguiente línea:

```
cache_dir ufs /var/spool/squid 100 16 256
```

Se puede incrementar el tamaño del caché hasta donde lo desee el administrador. Mientras más grande el caché, más objetos de almacenarán en éste y por lo tanto se utilizará menos el ancho de banda. La siguiente línea establece un caché de 700 MB:

```
cache_dir ufs /var/spool/squid 700 16 256
```

Los números 16 y 256 significan que el directorio del caché contendrá 16 subdirectorios con 256 niveles cada uno. Es recomendable no modificar estos números.

Es muy importante considerar que si se especifica un determinado tamaño de caché y este excede al espacio real disponible en el disco duro, Squid se bloqueará inevitablemente.

Parámetro ftp_user

Al acceder a un servidor FTP de manera anónima, por defecto Squid enviará como contraseña Squid@. Si se desea que el acceso anónimo a los servidores FTP sea más informativo, o bien si se desea acceder a servidores FTP que validan la autenticidad de la dirección de correo especificada como contraseña, puede especificar la dirección de correo electrónico que uno considere pertinente.

```
ftp_user proxy@su-dominio.net
```

Parámetro ftp_passive

Si se tiene un muro cortafuegos que no permite acceder servidores FTP más que de modo pasivo, debe habilitarse *ftp_passive* con el valor *on*.

```
ftp_passive on
```

Listas de Control de Acceso

Es necesario establecer Listas de Control de Acceso que definen una red o bien cierta máquinas en particular. A cada lista se le asignará una Regla de Control de Acceso que permitirá o denegará el acceso a Squid. Procedamos a entender como definir unas y otras.

Regularmente una lista de control de acceso se establece siguiendo la siguiente sintaxis:

```
acl [nombre de la lista] src [lo que compone a la lista]
```

Si uno desea establecer una lista de control de acceso que defina sin mayor trabajo adicional a toda la red local definiendo la IP que corresponda a la red y la máscara de la sub-red. Por ejemplo, si se tienen una red donde las máquinas tienen direcciones IP 192.168.1.n con máscaras de sub-red 255.255.255.0, podemos utilizar lo siguiente:

```
acl miredlocal src 192.168.1.0/255.255.255.0
```

También pueden definirse una Lista de Control de Acceso invocando un fichero localizado en cualquier parte del disco duro, y en el cual se encuentra una lista de direcciones IP. Ejemplos:

```
acl permitidos "/etc/squid/permitidos"
```

El fichero */etc/squid/permitidos* contendrá algo como lo siguiente:

```
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.15
192.168.1.16
```

Lo anterior estaría definiendo que la Lista de Control de Acceso denominada *permitidos* estaría compuesta por las direcciones IP incluidas en el fichero */etc/squid/permitidos.*

Reglas de Control de Acceso

Estas reglas definen si se permite o no el acceso a Squid. Se aplican a las Listas de Control de Acceso. Deben colocarse en la sección de reglas de control de acceso definidas por el administrador, es decir, a partir de donde se localiza la siguiente leyenda:

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM
# YOUR CLIENTS
```

La sintaxis básica es la siguiente:

```
http_access  [deny o allow] [lista de control de acceso]
```

En el siguiente ejemplo consideremos una regla que establece acceso permitido a Squid a la Lista de Control de Acceso denominada *permitidos*:

```
http_access allow permitidos
```

También pueden definirse reglas valiéndose de la expresión !, la cual significa *excepción*. Pueden definirse, por ejemplo, dos listas de control de acceso, una denominada *lista1* y otra denominada *lista2*, en la misma regla de control de acceso, en donde se asigna una expresión a una de estas. La siguiente establece que se permite el acceso a Squid a lo que comprenda *lista1* excepto aquello que comprenda *lista2:*

```
http_access allow lista1 !lista2
```

Este tipo de reglas son útiles cuando se tiene una IP dentro de un rango de red al que se debe permitir acceso, y otro grupo dentro de la misma red al que se debe denegar el acceso.

Parámetro cache_mgr

Por defecto, si algo ocurre con el caché, como por ejemplo que muere el proceso, se enviará un mensaje de aviso a la cuenta *webmaster* del servidor. Puede especificarse una distinta si acaso se considera conveniente.

```
cache_mgr joseperez@midominio.net
```

Estableciendo el Idioma por defecto

Squid incluye traducción a distintos idiomas de las distintas páginas de error e informativas que son desplegadas en un momento dado. Dichas traducciones se pueden encontrar en */usr/lib/squid/errors/.* Para poder hacer uso de las páginas de error traducidas al español, es necesario cambiar un enlace simbólico localizado en */etc/squid/errors* para que apunte hacia */usr/lib/squid/errors/Spanish* en lugar de hacerlo hacia */usr/lib/squid/errors/English.*

Elimine primero el enlace simbólico actual:

```
rm -f /etc/squid/errors
```

Coloque un nuevo enlace simbólico apuntando hacia */usr/lib/squid/errors/Spanish.*

```
ln -s /usr/lib/squid/errors/Spanish /etc/squid/errors
```

2.3.5.4 Iniciar, reiniciar y añadir el servicio al arranque del sistema

Una vez terminada la configuración, ejecute el siguiente comando para iniciar por primera vez Squid:

```
/etc/rc.d/init.d/squid start
```

ó

```
service squid start
```

Si necesita reiniciar para probar cambios hechos en la configuración, ejecute lo siguiente:

```
/etc/rc.d/init.d/squid restart
```

ó

```
service squid restart
```

Si desea que Squid inicie de manera automática la próxima vez el sistema, ejecute lo siguiente:

```
/sbin/chkconfig --level 345 squid on
```

ó

```
chkconfig squid on
```

Lo anterior habilitará a Squid en todos los niveles de corrida.

2.3.5.5 Depuración de errores

Cualquier error al inicio de Squid solo significa que hubo errores de sintaxis, errores de digitación o bien están citando incorrectamente las rutas hacia los ficheros de las Listas de Control de Acceso.

Puede realizar diagnóstico de problemas indicándole a Squid que vuelva a leer configuración, lo cual devolverá los errores que existan en el fichero */etc/squid/squid.conf.*

```
service squid reload
```

Cuando se trata de errores graves que no permiten iniciar el servicio, puede examinarse el contenido del fichero */var/log/squid/squid.out* con el mandato less, more o cualquier otro visor de texto:

```
less /var/log/squid/squid.out
```

2.3.5.6 Iptables reemplaza ipchain

Desde el kernel2.4, GNU/Linux utiliza Netfilter, el cual se configura a través de iptables. La sintaxis cambia con respecto a ipchains, y a fin de permitir a los administradores darse tiempo de adaptarse, distribuciones como Red Hat (TM) incluyeron soporte para ipchains a manera de aplicación de legado.

Se recomienda desinstalar ipchains y los paquetes que dependan de este.
Es importante utilizar la más reciente versión de iptables para la distribución utilizada.

Antes de desinstalar ipchains, primero debe eliminarse cualquier regla que pudiese existir

```
# /sbin/ipchains -X
# /sbin/ipchains -F
# /sbin/ipchains -Z
```

A continuación debe removerse el módulo de ipchains para permitir la carga de módulo *ip_tables.*

```
# /sbin/rmmod ipchains
# /sbin/modprobe ip_tables
```

Para terminar, se desinstala *ipchains* y toda la paquetería que dependa de éste.

```
# rpm -e ipchains lokkit gnome-lokkit firewall-config
```

Estos ajustes deben poder permitir utilizar *iptables* en lugar de *ipchains* sin mayor problema.

2.3.5.7 Re-direccionamiento de peticiones

En un momento dado se requerirá tener salida transparente hacia Internet para ciertos servicios, pero al mismo tiempo se necesitará re-direccionar peticiones hacia servicio Web, Web SSL, ftp, gopher o WAIS hacia el puerto donde escucha

peticiones Squid (3128), de modo que no haya salida alguna hacia alguno de estos protocolos sin qué esta pase antes por Squid.

El re-direccionamiento lo hacemos a través de *iptables.* Considerando para este ejemplo que la red local se accede a través de una interfaz *eth0*, el siguiente esquema ejemplifica un re-direccionamiento, haciendo que cualquier petición hacia el puerto 80 (servidor HTTP) hecha desde la red local hacia el exterior, se redirecciona hacia el puerto 3128 del servidor:

```
#  /sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80
# -j REDIRECT --to-port 3128
```

Con el fin de re-direccionar peticiones hacia los puertos 20 (FTP-data), 21 (FTP), 70 (GOPHER), 80 (HTTP), 210 (WAIS) y 443 (HTTPS), podemos añadir al guión del muro cortafuegos lo siguiente:

```
# FTP-data
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 20 -j REDIRECT --to-port
3128

# FTP
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 21 -j REDIRECT --to-port
3128

# GOPHER
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 70 -j REDIRECT --to-port
3128
```

```
# HTTP
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port
3128

# WAIS
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 210 -j REDIRECT --to-port
3128

# HTTPS
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port
3128
```

Puede añadirse re-direccionamiento hacia otros puertos menos usuales pero que llegan ser utilizados para acceder hacia algún servicio, como por ejemplo el 81, utilizado en ocasiones como alternativo para HTTP, y el 563, utilizado para NNTP sobre SSL.

También se pueden re-direccionar los puertos utilizados por los clientes de mensajería instantánea, siempre que estos permitan hacer uso de un servidor Proxy, ya que de lo contrario quedarían bloqueados, como sería el caso del protocolo ICQ, mismo que no tiene soporte para servidor Proxy.

- AIM: puertos 9898, 5190 al 5193
- Yahoo! Messenger: puertos 5050 u 80 para mensajes, 5000 al 5010 para conversaciones por voz y 5100 para vídeo.
- MSN Messenger: puerto 1863 y 80 si no puede usarse el primero.

```
# A veces utilizado para HTTP
```

```
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 81 -j REDIRECT --to-port 3128

# NNTP sobre SSL
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 563 -j REDIRECT --to-port 3128

# AIM
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 9898 -j REDIRECT --to-port 3128
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 5190:5193 -j REDIRECT --to-port 3128

# Yahoo! Messenger
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 5000:5010 -j REDIRECT --to-port 3128
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 5050 -j REDIRECT --to-port 3128

# MSN Messenger
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 1863 -j REDIRECT --to-port 3128
```

3.5.6 Delays Pools

2.3.6.1 Introducción

Una alternativa para controlar el ancho de banda con Squid es la herramienta de los Delays Pools.

Squid es probablemente el mejor proxy-caché que existe para HTTP. Es estable, fácil de configurar y seguro. Una de las razones de instalar Squid es disminuir la carga que el tráfico HTTP sobre el canal de Internet

2.3.6.2 Resolviendo el Problema

Identificando el problema

Si un usuario o conjunto reducido de usuarios son los que consumen la mayor parte del canal o es simplemente que de verdad necesita más ancho de banda. Hay que tener en cuenta que GNU/Linux y Squid no hacen milagros como la generación espontánea de ancho de banda. Para grupos de 300 usuarios o más se estima un consumo promedio de 2.5Kb/s de bajada por 1.2Kb/s de subida. No espere que un enlace telefónico de 56Kb/s o de banda ancha de 128Kb/s con reuso de 1:4 ó 1:6haga que 50 usuarios naveguen como si estuvieran pegados directamente al backbone.

Identificación del cuello de botella

No todo se le puede atribuir "al poco ancho de banda que le comprende al IPS". Puede estar en su LAN o en el propio servidor Proxy con Squid como Proxy/caché de Internet usted no controla el tráfico interno.

Revise la instalación de Squid

Si instala el Proxy en modo transparente lo más probable es que usted no haga pasar por él todas las peticiones HTTP y el tráfico de todas las aplicaciones que puedan hablar HTTP. Existen servidores que reciben peticiones en puertos no estándar y siempre hay un puerto a la espera de ser redirigido.

Controlar el P2P (Mensajería instantánea)

No se puede detenerlos únicamente con Squid. Se los controla con una agresiva política de firewall, parches para le kernel de Linux/iptables (POM de netfilter) y un Proxy No Transparente con Delay Pools.

2.3.6.3 Clases de Delay Pool

Los Delay Pools son la herramienta para llevar a cabo el control de ancho de banda de Proxy (*rate limiting y traffic shaping*). Lo mejor de esto radica en que controlan el ancho de banda, sin causar penalidades sobre los objetos traídos desde el caché. En lenguaje técnico de Proxy, los Delay pools afectan los *caché misses*, no los *caché hits.*

Un Delay Pool es como un tanque de agua el cual tiene un tubo de entrada y otro de salida. El tubo de entrada debido a su diámetro y a la apertura de la llave de paso solo permite que el agua entre a una rata fija. El tubo de salida debido a su gran diámetro no tiene estas limitaciones y puede vaciar el tanque inmediatamente si la llave de paso se abre lo suficiente.

Finalmente, el tanque siempre almacena una cantidad máxima de agua. Entendido esto asociamos:

1. El diámetro del tubo de entrada representa el ancho de banda disponible en total. Usted puede abrir la llave tanto como pueda pero la cantidad máxima de agua que sale no sobrepasa un tope máximo.
2. La apertura de la llave de paso de entrada representa el canal destinado para uso del Proxy/Delay Pool respectivo. Esta es la rata (de llenado) del Delay Pool.
3. La cantidad máxima de agua que almacena el tanque es el tamaño (*size*) del Delay Pool. Esta es la parte menos comprendida: el size del Delay Pool permite ráfagas (*burst*) de descarga mientras el Delay Pool se vacía. Cuando el Delay Pool queda con un tamaño de cero solo es posible descargar a la rata definida.

4. La apertura de la llave de salida representa su demanda de canal. Entre más abre la llave, más agua intenta obtener. Entre más archivos trate de traer desde Internet, más canal consume.

Existen tres clases de Delay Pool lo que nos permite tener cierta flexibilidad en su uso:

Clase 1

El Delay Pool clase 1 define una única estructura de control. Este limita el uso del canal de manera global sin importar cómo lo usan los clientes internamente o cómo está definida lógicamente la LAN. En el inglés técnico se habla de la definición de un único *agrégate bucket.* Esta es la opción indicada si usted desea limitar el ancho de banda que usa Squid, sin importar cómo lo emplean los usuarios.

Clase 2

Este es un Delay clase 1 con un 256 Delay Pools clase 1 subordinados a este. En inglés técnico un *agregate bucket y 256 individual buckets.* En nuestra abstracción un tanque principal y 256 tanques secundarios alimentados por el tanque principal. Con este Delay es posible controlar el canal que usan 256 clientes.

Para asignar el canal a cada cliente, Squid asume que su LAN es una LAN clase C y usa los últimos 8 bits del número IP del cliente para identificarlo y manejarlo en su individual bucket correspondiente. En la práctica solo se pueden controlar 253 clientes descontando la dirección de red, la dirección de broadcast y la dirección del Proxy.

Clase 3

Este es un Delay Pool clase 1 con 256 Delay Pools clase 2 subordinados a este. En ingles técnico un *agrégate bucket*, 256 *network buckets*, y 65.536 *individual buckets.* Está orientado para manejar la asignación de ancho de banda en redes clase B. los bits 17 a 24 del número IP identifican la red y los bits 17 a 32 el cliente.

2.3.6.4 Definición de Delay Pools

Cuando se compila squid con la opción *–enable-delay-pools* se tiene acceso a esta característica. En *squid.conf* la cantidad de Delays Pools a emplear se define con la directiva *delay_pools* **.**

Sintaxis:

```
delay_pools  N
```

Donde N>0 representa la cantidad de Delay Pools a usar.

Ejemplo:

```
delay_pools 3
```

Con esto se le dice a Squid que se van a usar y definir tres Delay Pools.

2.3.6.5 Definición de la clase

La clase del Delay Pool se especifica con la directiva *delay_class.*

Sintaxis:

```
delay_class  id class
```

Donde id>0, class=[1| 2 | 3]; id es el identificador y class la clase.

Ejemplo:

```
delay_class 1 3  #el Delay Pool número 1 será clase 3
delay_class 2 1 #el Delay Pool número 2 será clase 1
delay_class 3 2 #el Delay Pool número 3 será clase 2
```

Los Delay Pools no tienen nombre, se identifican con un número que empieza en 1 y termina en N.

2.3.6.6 Parámetros del Delay Pool

Los parámetros de cada Delay Pool se definen por medio de la directiva *delay_parameters.*

Sintaxis:

```
delay_parameters id rate/size [ rate/size [rate/size]]
```

Los valores de rate y size son dados en Bytes. Por ende no olvide hacer la conversión respectiva de Kbits como le venden el canal a Bytes. Size es dos o tres veces el valor de rate.

Ejemplos:

```
delay_parameters 1 76800/230400 42800/10000 10000/7000
```

Un Delay clase 3 con 600Kb/s (76800B/s) en total para navegación, con un tamaño para ráfagas (burst) globales de 1800Kb (230400Bytes). Para cada subred se asigna un canal máximo de 334.3Kb/s un tamaño para ráfagas de 781.2Kb (100000bytes) con un ancho de banda para cada host de 78.1Kb/s (10000B/s) con la posibilidad de ráfagas de descargas de 546.8Kb (70000Bytes). Note que los valores para cada subred y host exceden los límites de canal disponibles si hay más de 4 clientes navegando en una subred o dos subredes demandan todo el canal asignado. En este caso se produce una condición de competencia y el primero que solicita elcanal es el que lo obtiene. Es de suponer que esta asignación es para una organización con usuarios que navegan poco y requieren un buen desempeño al momento de solicitar un archivo. Este es un buen ejemplo de cómo se puede jugar con los parámetros del Delay Pool teniendo en cuenta las costumbres de navegación de la organización.

```
delay_parameters 1 76800/230400
```

Un Delay clase 1. Se usan máximo 600Kb/s de ancho de banda con ráfagas de descarga de 1800Kb. Solo se limita el ancho de banda que usa en total sin importar cómo se distribuye el canal entre los clientes, lo cual da la posibilidad a condiciones de competencia por el ancho de banda en todo momento.

```
delay_parameters 1 340787/1022361 10000/200000
```

Un Delay clase 2. Define que se usarán máximo 2.6Mb/s para navegación con ráfagas de 7.8Mb para una asignación de canal a máximo 256 clientes de 78.1Kb/s con ráfagas de descarga de 195.3Kb (esto significa que pueden descargar 195.3KB a todo lo que de el canal si tienen el individual bucket lleno). Este montaje es para una

organización cuyos usuarios demandan una gran cantidad de canal. Aquí el peor caso se presenta cuando hay 34 clientes demandando todo el canal asignado de forma continua.

2.3.6.7 Deshabilitando los buckets

En los Delay Pools clase 2 y clase 3 es posible deshabilitar los buckets que no se desea utilizar colocando -1/-1 en el bucket correspondiente. Por ejemplo:

```
#asigno el canal a hosts en una red clase C sin límite global
delay_parameters 1 -1/-1 10000/200000

#asigno canal en una red dase B a hosts individuales sin #límites por subred
delay_parameters 2 340787/1022362 -1/-1 10000/200000

#asigno canal en una red dase B a cada su red sin #importar los hosts
delay_parameters 3 340787/1022361 10000/200000 -1/-1
```

2.3.6.8 Delay_access

Definen por medio de acl's cuáles peticiones pasan por el Delay y cuáles no. Ver los ejemplos de uso en la sección de implementaciones.

Sintaxis:

```
delay_access    id      allow acl name | deny acl name
```

2.3.6.9 Nivel de bucket al inicio

La directiva *delay_initial_bucket_level*, controla qué tan lleno estará el bucket al arranque del Squid o cuando se reconfigura. Esto afecta tanto a los *individual buckets* como a los *agrégate buckets.* Note que los *buckets* no se crean sino hasta que son referenciados por primera vez. El valor es un porcentaje.

Sintaxis:

```
delay_initial_bucket_level    porcentaje
```

Porcentaje toma valores entre 0-100

Ejemplo:

```
delay_initial_bucket_level 90

#el bucket inicia un 90% lleno (con un 90% de su size #definido)
```

2.3.6.10Monitoreo y prueba

Puede monitorear los Delay Pools con el comando:

```
squidclient    mgr:delay   |   less
```

Para probar el funcionamiento de los Delay Pools pruebe a descargar un archivo grande. Una vez haya descargado dos o tres veces el tamaño del *size* del *bucket* respectivo, notará que el archivo baja a la rata fijada. Si puede usar *wget* o *iptraf* para monitorear el ancho de banda usado en la descarga. IE y Firefox indican promedios en la velocidad de descarga, y por lo tanto pueden producir la sensación de que los Delay Pools no funcionan bien.

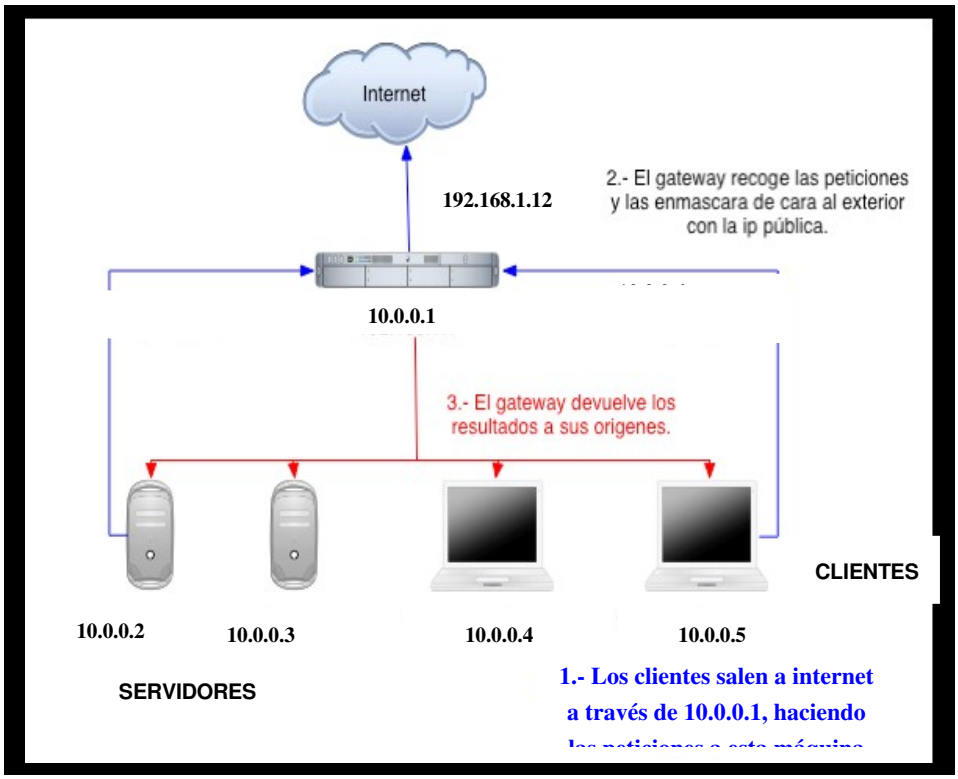
3. CAPITULO TRES: CONFIGURACIONES

3.1 CONFIGURACIÓN DE UNA NAT CON IPTABLES

3.1.1 Requerimientos

- Un CPU PII o más
- Sistema Operativo GNU/Linux con Centos 5.1
- Software Iptables
- Dos tarjetas de conexión a interfaz

3.1.2 Topología de la Red



La red consta de una red LAN y otra WAN:

- Ip WAN en la eth0 con dirección de red 192.168.1.12/24

- Ip LAN en la eth1con dirección de red 10.0.0.1/8

3.1.3 Procedimiento

Configuración de la Red

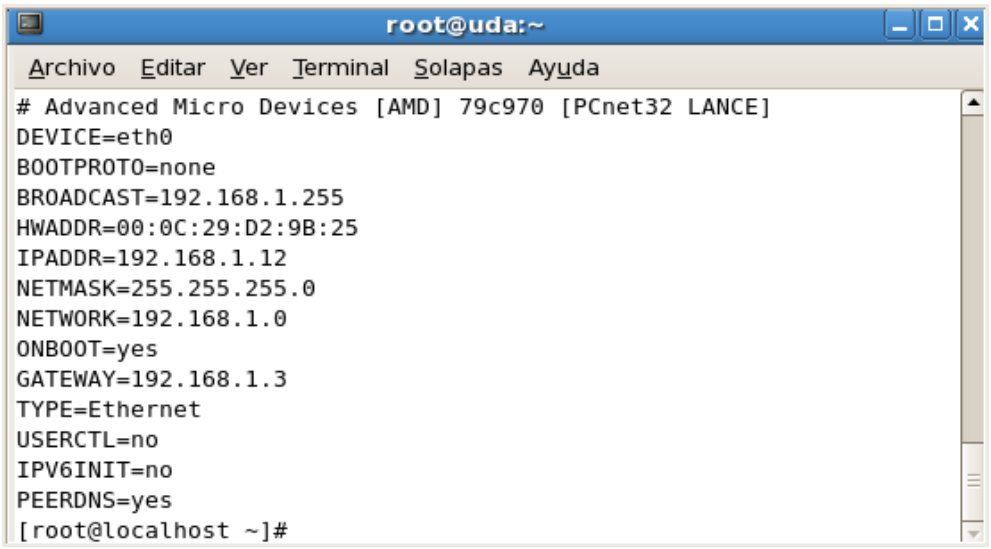
- Primero agregamos las dos tarjetas de red.
- Verificamos si estas tarjetas de red, están funcionando correctamente con el siguiente comando:

```
ls /etc/sysconfig/network-scripts/ifcfg-eth* | wc -l
```

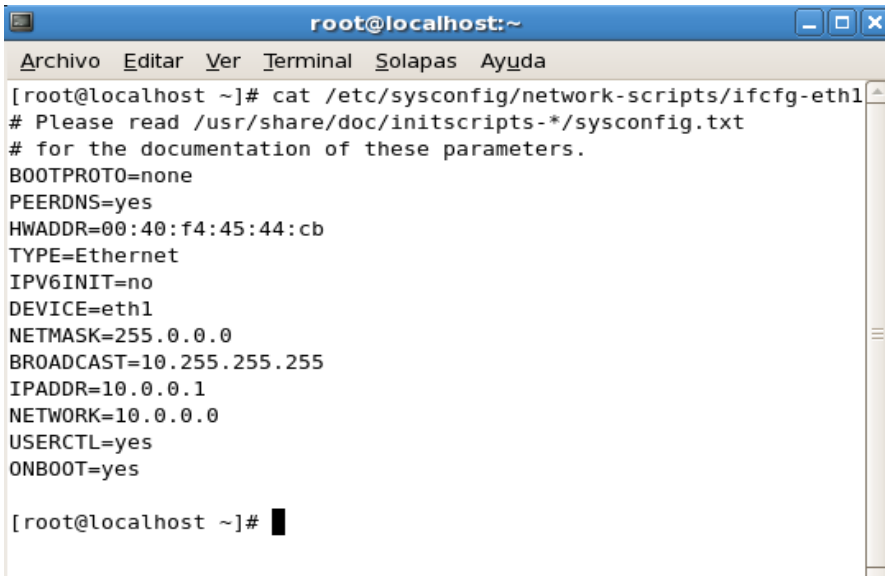
Su resultado debe ser "2" tarjetas detectadas.

- Configurar la eth0 con dirección pública para la Internet (ISP), con el siguiente comando:

```
cat /etc/sysconfig/network-scripts/ifcfg-eth0
```

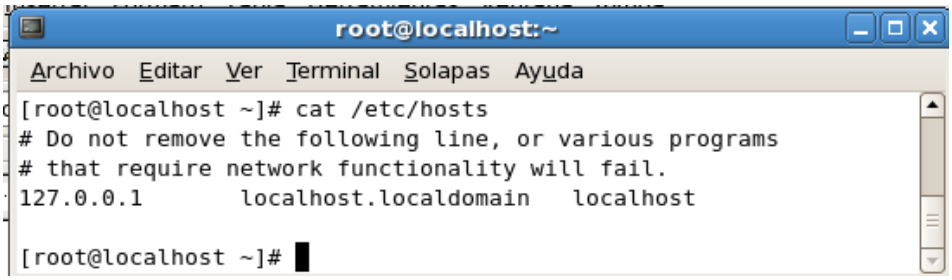


- Configurar la eth1 para la LAN con IP privada



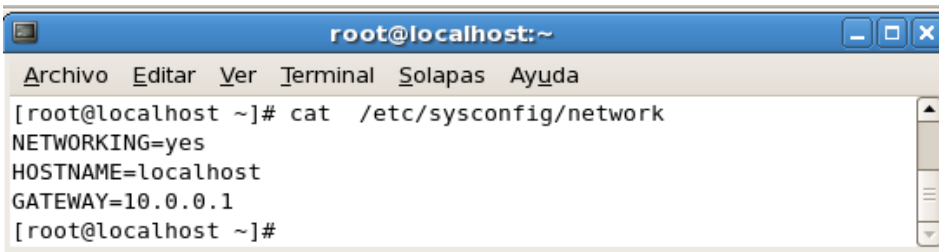
- Configurar los Hosts

cat /etc/hosts



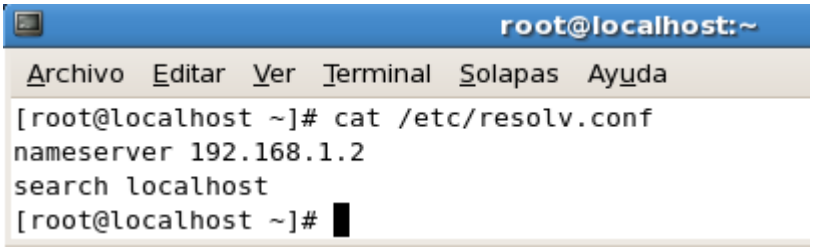
- Configuración de la Puerta de Enlace

cat /etc/sysconfig/network



- Configuración DNS

```
cat /etc/resolv.conf
```



Configuración de Iptables

- Encontrar el archivo fw.sh

```
get fw.sh
```

- Copiar el archivo fw.sh en otro archivo llamado firewall.sh

```
cp fw.sh firewall.sh
```

- Crear las reglas del firewall:

No se debe permitir tráfico hacia el firewall, salvo ssh(22) y dns(53).

```
$I -A INPUT -p tcp --sport 22 -j ACCEPT
$I -A INPUT -p tcp --sport 53 -j ACCEPT
$I -A INPUT -p udp --sport 53 -j ACCEPT
```

No se debe permitir el tráfico hacia Internet con puertos origen menor a 1024 ICMP (well know ports).

```
$I -A FORWARD -p tcp –sport 0:1023 -j REJECT
$I -A FORWARD -p tcp –sport :>1024 -j ACCEPT
```

Solo se debe permitir el tráfico hacia Internet de http (tcp 80), https(tcp 443), ssh(tcp 22), dns(tcp y udp 53), ftp (tcp 20 y 21), smtp (tcp 25), imap (tcp 143) y mensajes icmp.

```
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 80 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 443 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 22 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 53 -j ACCEPT
$I -A FORWARD -p udp -s 10.0.0.0/8 --dport 53 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 20 -j ACCEPT
$I -A FORWARD -p udp -s 10.0.0.0/8 --dport 21 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 25 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 143 -j ACCEPT
$I -A FORWARD -p ICMP -s 10.0.0.0/8 -j ACCEPT
```

No se debe permitir el tráfico entrante desde Internet, excepto conexiones tcp ya establecidas, dns, servidor web (únicamente a la ip que tiene el servicio) y mensajes icmp.

```
$I -A FORWARD -p tcp -d 10.0.0.0/8 --sport 53 -j ACCEPT
$I -A FORWARD -p udp -d 10.0.0.0/8 --sport 53 -j ACCEPT
$I -A FORWARD -p tcp -d 10.0.0.0/8 -s 10.0.0.2 -j ACCEPT
$I -A FORWARD -p tcp -d 10.0.0.0/8 -j REJECT
```

- Ejecutamos el archivo firewall.sh

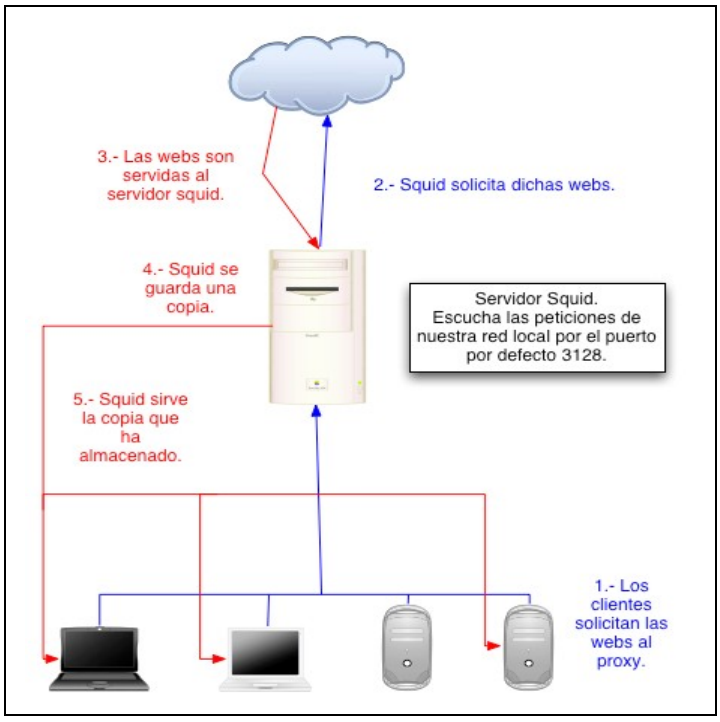
```
./firewall.sh
```

- Reiniciamos el servicio de iptables

```
service iptables restart
```

3.2 CONFIGURACIONES DE SQUID COMO FIREWALL

3.2.1 Topología de la Red



3.2.2 Equipamiento lógico necesario

Para poder llevar acabo los procedimientos descritos en el tutorial, usted necesitará tener instalado al menos lo siguiente:

- squid-2.5-STABLE3
- http-2.0.x (opcional)
- Todos los parches de seguridad disponibles para la versión del sistema operativo que esté utilizando

En el capítulo 2 especificamos las configuraciones básicas necesarias para el inicio correcto de Squid.

3.2.3 Restricciones de Acceso a Sitios de Red

Denegar el acceso a ciertos Sitios de Red permite hacer un uso más racional del ancho de banda con el que se dispone.

- Primero generamos una lista la cuál contendrá direcciones de Red y palabras que prefiera el administrador, la cuál la guardaremos como */etc/squid/sitiosdenegados.*

```
www.sitioporno.com
www.otrositioporno.com
sitioindeseable.com
otrositioindeseable.com
napster
sex
porn
mp3
xxx
adult
warez
celebri
```

- Debemos definir una Lista deControl de Acceso en */etc/squid/squid.conf* que a su vez defina al fichero */etc/squid/sitiosdenegados.* Esta lista la denominaremos como "sitiosdenegados". De modo tal, la línea correspondiente quedaría del siguiente modo:

```
acl sitiosdenegados url_regex "/etc/squid/sitiosdenegados"
```

Habiendo hecho lo anterior, deberemos tener en la sección de Lista de Control de Acceso algo como lo siguiente:

```
#
# Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl redlocal src 192.168.1.0/255.255.255.0
acl password proxy_auth REQUIRED
acl sitiosdenegados url_regex "/etc/squid/sitiosdenegados"
```

A continuación modificaremos una Regla de Control de Acceso existente agregando con un símbolo de ! Que se denegará el acceso a la Lista de Control de Acceso denominada *sitiosdenegados*:

```
http_access allow redlocal !sitiosdenegados
```

- Hay veces que incluir una palabra en los sitios denegados puede afectar el acceso a un sitio de Red, que contengan un patrón que consideramos apropiado.

En la Lista de Control de Acceso de sitios denegados está la palabra *sex*, esta denega*ría e*l acceso a cualquier nombre de dominio que incluya dicha cadena de caracteres, como www.extremesex.com. Sin embargo también estaría bloqueando a sitios como www.sexualidadjovel.cl, el cuál no tiene que ver en lo absoluto con pornografía, sino orientación sexual para la juventud. Podemos añadir este nombre de dominio en un fichero que denominaremos */etc/squid/sitios-inocentes.*

Este fichero será definido en una Lista de Control de Acceso del mismo modo en que se hizo anteriormente con el fichero que contiene dominios y palabras denegadas.

```
acl inocentes url_regex "/etc/squid/sitios-inocentes"
```

Para hacer uso del fichero, solo bastará utilizar la expresión **!** en la misma línea utilizada para la Regla de Control de Acceso establecida para denegar el mismo.

```
http_access allow all inocentes
```

- Finalizamos el procedimiento reiniciando Squid para que tomen efecto los cambios.

```
service squid restart
```

3.2.4 Restricción de Acceso a Contenido por Extensión

Definiendo elementos de la Lista de Control de Acceso

Lo primero será generar una lista llamada *"listaextensiones"* definida en el fichero */etc/squid/listaextensiones* la cual contendrá las extensiones que vamos a restringir.

```
\.avi$
\.mp4$
\.mp3$
\.mp4$
\.mpg$
\.mpeg$
\.mov$
\.ra$
\.ram$
\.rm$
\.rpm$
\.vob$
\.wma$
\.wmv$
\.wav$
```

Parámetros en /etc/squid/squid.conf

Debemos definir una Lista de Control de Acceso que a su vez defina al fichero /etc/squid/listaextensiones.

```
acl listaextensiones urlpath_regex "/etc/squid/listaextensiones"
```

Deberemos tener en la sección de Listas de Control de Acceso algo como lo siguiente:

```
#
# Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl redlocal src 192.168.1.0/255.255.255.0
acl password proxy_auth REQUIRED
acl sitiosdenegados url_regex "/etc/squid/sitiosdenegados"
acl listaextensiones urlpath_regex "/etc/squid/listaextensiones"
```

A continuación modificaremos una Regla de Control de Acceso existente agregando con un símbolo de ! que se denegará el acceso a la Lista de Control de Acceso denominada listaextensiones:

```
http_access allow redlocal !listaextensiones
```

Finalizamos el procedimiento reiniciando Squid para que tomen efecto los cambios y podamos hacer pruebas.

```
service squid restart
```

3.3 CONTROL DE ANCHO DE BANDA CON DELAYS POOLS

3.3.1 Equipamiento necesario

Para hacer que funcione necesitamos al menos el proxy Squid; si queremos afinar en su configuración tendremos que familiarizarnos con iptables y CBQ.

3.3.1.1 Instalar Squid con la prestación Delay Pools

Hay que recalcar que no todas las distribuciones Squid vienen con esta característica, así que si ya se tiene instalado se deberá desinstalarlo e instalarlo de nuevo con las delay pools activadas.

Descargamos las fuentes de Squid de http://www.squid-cache.org.

Desempaquetamos el archivo squid-2.6.STABLE-src.tar.gz

```
tar xzpf squid-2.6.STABLE-src.tar.gz
```

Compilamos e instalamos Squid, todo va en una sola línea

```
#./configure --prefix=/opt/squid --exec-prefix=/opt/squid –enable-delay-pools --enable-cache-digests --enable-poll –disable-ident-lookups --enable-truncate --enable-removal-policies
#make all
#make install
```

3.3.2 Configurar Squid para poder usar Delay Pools

Configuración previa

Configuramos el archivo squid.conf localizado en /etc/squid/squid.conf

- Los puertos por los que escuchará nuestro Squid.

```
http_port 8080
icp_port 3130
```

- Los cgi-bin no se cachearán.

```
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
```

- La memoria que usará Squid. Bueno, Squid usará mucha más que ésa.

```
cache_mem 16 MB
```

- El número 250 significa que Squid usará 250 megabytes de espacio en disco.

```
cache_dir ufs /cache 250 16 256
```

- Lugares en los que irán los archivos de bitácora de Squid.

```
cache_log /var/log/squid/cache.log
cache_access_log /var/log/squid/access.log
cache_store_log /var/log/squid/store.log
cache_swap_log /var/log/squid/swap.log
```

- Cuántas veces rotar los archivos de bitácora antes de borrarlos.

```
logfile_rotate 10
redirect_rewrites_host_header off
```

```
cache_replacement_policy GDSF
acl localnet src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
acl Safe_ports port 80 443 210 119 70 20 21 1025-65535
acl CONNECT method CONNECT
acl all src 0.0.0.0/0.0.0.0
http_access allow localnet
http_access allow localhost
http_access deny !Safe_ports
http_access deny CONNECT
http_access deny all
maximum_object_size 3000 KB
store_avg_object_size 50 KB
```

- Configure esto si quiere que su proxy funcione de manera transparente. Eso significa que por lo general no tendrá que configurar todos los navegadores de sus clientes, aunque tiene algunos inconvenientes. Si deja esto sin comentar no pasará nada peligroso.

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

- Todos los usuarios de nuestra LAN serán vistos por los servidores web externos como si usasen Mozilla en Linux.

```
anonymize_headers deny User-Agent
```

fake_user_agent Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.6+) Gecko/20011122

- Para acelerar aún más nuestra conexión ponemos dos líneas similares a las de más abajo. Apuntarán a un servidor proxy [parent] que usará nuestro propio Squid. No olvide cambiar el servidor por uno más rápido para usted. Puede utilizar ping, traceroute y demás herramientas para comprobar la velocidad. Asegúrese de que los puerto http e icp son los correctos.

Descomente las líneas que comienzan por "cache_peer" de ser necesario. Éste es el Proxy que va a usar para todas las conexiones excepto para las direcciones e IPs que comiencen por "!".

cache_peer w3cache.icm.edu.pl parent 8080 3130 no-digest default

- Esto resulta útil cuando queremos usar el Cache Manager. Copie cachemgr.cgi al cgi-bin de su servidor web. Podrá acceder a él una vez lo haya hecho introduciendo en un navegador la dirección http://su-servidor-web/cgi-bin/cachemgr.cgi

cache_mgr your@email
cachemgr_passwd secret_password all

- Éste es el nombre de usuario con el que trabajará nuestro Squid.

cache_effective_user squid
cache_effective_group squid
log_icp_queries off
buffered_logs on

Configuración Delay Pools

- No queremos limitar las descargas en nuestra red local.

```
acl red_local url_regex -i 10.0.0.0/8
```

- Queremos limitar la descarga de este tipo de archivos. Ponga todo esto en una única línea. No bloqueamos .html, .gif, .jpg y archivos similares porque por lo general no consumen demasiado ancho de banda.

```
acl extensiones url_regex -i ftp .exe .mp3 .vqf .tar.gz .gz .rpm .zip .rar .avi .mpeg .ram .rm .iso .raw .wav .mov
```

- Queremos limitar el ancho de banda durante el día permitiendo el ancho de banda completo durante la noche. El acl de abajo interrumpirá sus descargas a las 23:59.

```
acl dia time 09:00-23:59
```

- Especificamos que tenemos dos delay_pools diferentes

```
delay_pools 2
```

- Primer delay pool: No queremos retrasar nuestro tráfico local. Hay tres clases de pools; aquí sólo hablaremos de la segunda. Primera clase de retraso (1) de segundo tipo (2).

```
delay_class 1 2
```

- Parámetros de 1/-1 significa que no hay límites.

```
delay_parameters 1 -1/-1 -1/-1
```

- redlocal: 10.0.0.0/8 que ya hemos puesto antes

```
delay_access 1 allow redlocal
```

- Segundo delay pool: Retrasa la descarga de los archivos mencionados en extensiones. Segunda clase de retraso (2) de segundo tipo (2).

```
delay_class 2 2
```

- Los números siguientes son valores en bytes. Debemos recordar que Squid no tiene en cuenta los bits de inicio/parada:

5000/150000 son valores para la red al completo

5000/120000 son valores para la IP independiente

Una vez los archivos descargados exceden los 150000 bytes (o el doble o el triple), las descargas proseguirán a 5000 bytes/sg

```
delay_parameters 2 5000/150000 5000/120000
```

- Ya hemos configurado antes el día de 09:00 a 23:59.

```
delay_access 2 allow dia
delay_access 2 deny !dia
delay_access 2 allow extensiones
```

- Para que nos coja los cambios, simplemente debemos reestablecer el servicio.

```
service squid restart
```

3.4 APENDICES

3.4.1 APENDICE UNO: Configuraciones del Firewall

Configuración de las Interfaces

Archivo vi /etc/sysconfig/network-scripts/ifcfg-eth0

#Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=none
BROADCAST=192.168.1.255
HWADDR=00:0C:29:D2:98:25
IPADDR=192.168.1.12
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
GATEWAY=192.168.1.3
TYPE=Ethernet
USERCTL=no
IPV6INIT=no
PEERDNS=yes

ARCHIVO vi /etc/sysconfig/network-scripts/ifcfg-eth1

#Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth1
BOOTPROTO=none
BROADCAST=10.255.255.255
HWADDR=00:40:f4:45:44:cb
IPADDR=10.0.0.1
NETMASK=255.255.255.0
NETWORK=10.0.0.0
ONBOOT=yes
GATEWAY=192.168.1.3
TYPE=Ethernet
USERCTL=yes
IPV6INIT=no
PEERDNS=yes

Configuración del Host

Archivo vi /etc/hosts

#Do not remove the following line, or various programs
#that require network functionality will fail.
127.0.0.1 localhost.localdomail localhost

Configuración de la Puerta de Enlace

Archivo vi /etc/sysconfig/network

NETWORK=yes
HOSTNAME=localhost
GATEWAY=10.0.0.1

Configuración del DNS

Archivo vi /etc/resolv.conf

nameserver 192.168.1.2
search localhost

Configuración del archivo firewall.sh

$I -A INPUT -p tcp --sport 22 -j ACCEPT #ssh
$I -A INPUT -p tcp --sport 53 -j ACCEPT #dns tcp
$I -A INPUT -p udp --sport 53 -j ACCEPT #dns udp
$I -A FORWARD -p tcp --sport 0:1023 -j REJECT #<1024
$I -A FORWARD -p tcp --sport :>1024 -j ACCEPT
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 80 -j ACCEPT #http
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 443 -j ACCEPT #https
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 22 -j ACCEPT #ssh
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 53 -j ACCEPT #dns tcp
$I -A FORWARD -p udp -s 10.0.0.0/8 --dport 53 -j ACCEPT #dns udp
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 20 -j ACCEPT #ftp tcp
$I -A FORWARD -p udp -s 10.0.0.0/8 --dport 21 -j ACCEPT #ftp udp
$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 25 -j ACCEPT #smtp

$I -A FORWARD -p tcp -s 10.0.0.0/8 --dport 143 -j ACCEPT #imap

$I -A FORWARD -p ICMP -s 10.0.0.0/8 -j ACCEPT #mensajes icmp

$I -A FORWARD -p tcp -d 10.0.0.0/8 --sport 53 -j ACCEPT

$I -A FORWARD -p udp -d 10.0.0.0/8 --sport 53 -j ACCEPT

$I -A FORWARD -p tcp -d 10.0.0.0/8 -s 10.0.0.2 -j ACCEPT

$I -A FORWARD -p tcp -d 10.0.0.0/8 -j REJECT

3.4.2 APENDICE DOS: Listas y Reglas de Control de Acceso para Squid

ARCHIVO vi /etc/squid/squid.conf

Listas de control de acceso por defecto:

acl all src 0.0.0.0/0.0.0.0

acl localhost src 127.0.0.1/255.255.255.255

Listas que definen conjuntos de maquinas

acl redlocal src 10.0.0.0/255.0.0.0

Listas que definen palabras contenidas en un URL

acl sitiosdenegados url_regex "/etc/squid/sitiosdenegados"

acl inocentes url_regex "/etc/squid/sitios-inocentes"

acl listaextensiones urlpath_regex “/etc/squid/listaextensiones”

Regla de Control de Acceso

http_access allow redlocal !sitiosdenegados

http_access allow all inocentes

http_access allow redlocal !listaextensiones

Archivos necesarios en el fichero /etc/squid/

Contenido en /etc/squid/sitiosdenegados

www.chicas.com

www.sexandsex.com

porn

girl

celebrit

extasis

drug

playboy

adult
xxx
napster

Contenido en /etc/squid/sitios-inocentes
www.sexualidadjovel.cl
www.sexualidad.com

Contenido en /etc/squid/listaextensiones
/.avi$
/.mp4$
/.mp3$
/.mpg$
/.mpeg$
/.mov$
/.ra$
/.ram$
/.rm$
/.rpm$
/.vob$
/.wma$
/.wmv$
/.wav$

3.4.3 APENDICE TRES: Configuración Delay Pools

Archivo /etc/squid/squid.conf

```
#puertos donde escuchan squid
http_port 8080
icp_port 3130

#memoria para squid
cache_mem 16 MB

#espacio en disco
cache_dir ufs /cache 250 16 256

#listas de control
acl red_local url_regex –i 10.0.0.0/8
acl extensiones url_regex -i ftp .exe .mp3 .vqf .tar.gz .gz .rpm .zip .rar .avi
.mpeg .ram .rm .iso .raw .wav .mov
#limitar el ancho de banda
acl dia time 09:00-23:59

#configuración de dos delay pools
delay_pools 2
#primer delay pools
delay_class 1 2
delay_parameters 1 -1/-1 -1/-1
delay_access 1 allow redlocal
#segundo delay pools
delay_class 2 2
delay_parameters 2 5000/150000 5000/120000
delay_access 2 allow dia
delay_access 2 deny !dia
delay_access 2 allow extensions
```

CONCLUSIONES

- Un firewall es la solución más efectiva que sirve entre la comunicación de dos redes cumpliendo determinadas trayectorias de seguridad, decidiendo si un paquete pasa, se modifica, se convierte o se descarta.
- Si es bien cierto que hay muchas soluciones para la seguridad, los firewalls son flexibles, seguros, y fáciles de manejar para el administrador
- Squid es una herramienta de alto desempeño que se ha venido desarrollando desde hace varios años y es hoy en día un muy popular y ampliamente utilizado entre los sistemas operativos como GNU/Linux y derivados de Unix®.
- Squid es muy confiable, robusto y versátil y se distribuye bajo los términos de la Licencia Pública General GNU (GNU/GPL). Siendo sustento lógico libre, está disponible el código fuente para quien así lo requiera.
- En mi conclusión personal, argumento que estas dos herramientas son fáciles de entender y manejar, y son útiles para el control en la seguridad de una organización.
- Los Delay Pools son un método para mantener a raya a los usuarios problemáticos más que para optimizar el uso del canal. Los Delay Pools se implementan cuando el ancho de banda es un recurso escaso, costoso y hay usuarios abusivos.

BIBLIOGRAFIA

- NEGROPONTE, Nicholas. "El mundo digital", año de publicación 1980.
- BARRIOS DUEÑAS, José. "Implementación de servidores con GNU/Linux, edición de Abril del 2007.
- Squid: The Definitive Guide By Duane Wessels. Publisher : O'Reilly. ISBN : 0-596-00162-2.
- Enrutamiento avanzado y control de tráfico en Linux / Linux Advanced Routing & Traffic Control HOWTO.
- http://www.redes-linux/index-php
- http://es.wikipedia.org/wiki/cortafuegos.htm
- http://es.wikipedia.org/wiki/capas-tcp-ip.htm
- http://www.wikilearning.com/monografia/netfilter_iptables_ii_parte/9945
- http://es.wikipedia.org/w/index.php?title=Iptables&redirect=yes
- http://www.wikilearning.com/monografia/netfilter_iptables/9944
- http://www.wikilearning.com/monografia/netfilter_iptables/9945
- http://redavalon.blogspot.com/2007/06/firewall-con-iptables.html
- http://www.netfilter.org/documentation/FAQ/netfilter-faq.html
- http://mmc.igeofcu.unam.mx/LuCAS/Manuales-LuCAS/blfs-es/blfs-es-5.0/postlfs/firewall.html#postlfs-security-fw-kernel
- http://squid.visolve.com/squid/index.htm.
- http://linuxfocus.org/Castellano/March2002/article235.shtml
- http://www.linuxparatodos.net/
- http://www.squid-cache.org/why
- http://es.wikipedia.org/wiki/squid.htm
- http://dns.bdat.net/documentos/squid/t1.html
- http://www.tufuncion.com/
- http://www.squid-cache.org/Intro/