Universidad Tecnológica de El Salvador

INFORMÁTICA Y CIENCIAS APLICADAS

TÉCNICO EN INGENIERÍA DE REDES COMPUTACIONALES

TEMA:

"LABORACIÓN DE MANUAL DIDÁCTICO WED SOBRE LA CONFIGURACIÓN DE SERVIDOR PROXY Y FIREWALL BAJO EL SISTEMA OPERATIVO LINUX PARA EL APRENDIZAJE DE LOS ESTUDIANTES DE LA CARRERA TÉCNICA DE INGENIERÍA EN REDES COMPUTACIONALES DE LA UNIVERSIDAD TECNOLÓGICA DE EL SALVADOR"

TRABAJO DE GRADUACIÓN PRESENTADO POR:

JORGE ALBERTO REYES ORELLANA
LUIS FERNANDO AGUILAR PORTILLO
ALBA IVETTE ARAGON DE ULLOA

PARA OPTAR AL GRADO DE:

TÉCNICO EN INGENIERÍA DE REDES COMPUTACIONALES

SEPTIEMBRE, 2007

SAN SALVADOR, EL SALVADOR, CENTROAMÉRICA

AUTORIDADES UNIVERSITARIAS

LIC. JOSÉ MAURICIO LOUCEL

RECTOR

ING. NELSON ZÁRATE SÁNCHEZ

VICERRECTOR ACADÉMICO

ING. LORENA DUQUE DE RODRÍGUEZ

DECANA DE LA FACULTAD DE INGENIERÍA

JURADO EXAMINADOR

ING. JULIO CESAR MENENDEZ

Presidente

ING. VICENTE ANTONIO ZARCEÑO

Primer Vocal

LIC. ANTONIO HERRERA PALACIOS

Segundo Vocal

SEPTIEMBRE, 2007

SAN SALVADOR, EL SALVADOR, CENTROAMERICA

Universidad Tecnológica de El Salvador

ACTA DE EXAMEN PROFESIONAL

HABIÉNDOSE REUNIDO EL JURADO CALIFICADOR INTEGRADO POR:

ING. VICENTE ANTONIO ZARCEÑO JOCO, LIC. ANTONIO HERRERA PALACIOS, ING. JULIO CESAR MENENDEZ

A LAS 12:00 M. DEL DIA: MIERCOLES, 18 DE JULIO DEL DOS MIL SIETE.

Y LUEGO DE HABER DELIBERADO SOBRE EL EXAMEN PROFESIONAL DE LOS ALUMNOS:

1- LUIS FERNANDO AGUILAR PORTILLO	CARNET 26-3951-2004
2- JORGE ALBERTO REYES ORELLANA	CARNET 26-4138-2004
3- ALBA IVETTE ARAGON DE ULLOA	CARNET 26-4365-2004

QUIENES PRESENTARON DEFENSA DE SU TRABAJO DE GRADUACION TITULADO:

" ELABORACIÓN DE MANUAL DIDÁCTICO WED SOBRE LA CONFIGURACIÓN DE SERVIDOR PROXY Y FIREWALL BAJO EL SISTEMA OPERATIVO LINUX PARA EL APRENDIZAJE DE LOS ESTUDIANTES DE LA CARRERA DE TÉCNICOS EN INGENIERIA DE REDES COMPUTACIONALES."

PARA OPTAR AL GRADO DE:

" TÉCNICO EN INGENIERIA DE REDES COMPUTACIONALES"

Y DEL CUAL TAMBIEN EVALUARON LOS CONOCIMIENTOS RELACIONADOS CON EL TEMA DEL MISMO. POR LO QUE ESTE JURADO RESUELVE DECLARAR EL EXAMEN COMO:

APROBADO

YA QUE CUMPLE CON LOS REQUISITOS ESTABLECIDOS EN EL REGLAMENTO DE GRADUACION DE LA UNIVERSIDAD.

SAN SALVADOR, 18 DE JULIO DEL DOS MIL SIETE.

F. _____
PRIMER VOCAL
ING. VICENTE ANTONIO ZARCEÑO JACO

F. _____
SEGUNDO VOCAL
LIC. ANTONIO HERRERA PALACIOS

F. _____
PRESIDENTE
ING. JULIO CESAR MENENDEZ

AGRADECIMIENTOS

A DIOS TODOPODEROSO:

Por brindarme sabiduría y paciencia, para vencer todos los obstáculos que la vida me presenta

A MI MADRE:

Mercedes Luz Orellana Por darme la vida, educarme y guiarme para alcanzar mis metas en la vida, Gracias por siempre.

A NUESTRO ASESOR:

Ing. Vicente Antonio Zarceño, por guiarnos y darnos valiosos consejos en nuestro proyecto. Gracias

Jorge Alberto Reyes Orellana

AGRADECIMIENTOS

Quiero agradecer a mis padres Sandra Cecilia Portillo y Raúl Eduardo Benítez por darme el apoyo moral y económico, siempre seran mi apoyo y mi guía, gracias por brindarme su apoyo en los momentos buenos y difíciles de mi Vida. **"Gracias por todo"**

Agradezco también a **DIOS TODOPODEROSO**, por darme fortaleza y sabiduría, y por ayudarme a superar los obstáculos de la vida.
"Gracias Padre"

Quiero agradecer a nuestro asesor el Ing. Vicente Antonio Zarceño, ya que sin su guía y sus consejos no hubiera sido posible esté proyecto. **"Muchas Gracias"**

Luis Fernando Aguilar Portillo

AGRADECIMIENTO

A DIOS TODOPODEROSO
Por ayudarme y darme la fuerza y la capacidad de haber terminado mi carrera, por proveerme de lo necesario para poder salir adelante en está etapa de mi vida.
¡GRACIAS PADRE CREADOR!

A MI ESPOSO
Edgar Ernesto Ulloa por apoyarme y estar siempre a mi lado ayudándome en todo lo que necesito ya que sin su ayuda nunca lo hubiera podido lograr, muchas gracias por su amor y comprensión este triunfo es de los dos.

A MI HIJA
Isabel Noemí Ulloa ha sido mi mayor fuente de inspiración para terminar mi carrera y que todo lo que he logrado es para tu bienestar mi bebé.

A MI MADRE
Maria Luz Colocho Por enseñarme el buen camino, por darme todo su amor y enseñarme el amor a nuestro Dios y que siempre estuvo a mi lado apoyándome en mis decisiones.

A MIS COMPAÑEROS DE TESIS
Por la compresión, ayuda y paciencia que me tuvieron a la realización de esté documento.

A MI ASESOR
Ing. Vicente Antonio Zarceño por la paciencia y la ayuda que nos brindo en todo el proceso de la realización de la tesis muchas gracias.

Alba Ivette Aragón de Ulloa

INDICE

CAPITULO II: DOCUMENTACION TECNICA Y ECONOMICA

CAPITULO III: PROTOTIPO

Introducción

El presente trabajo aporta toda la información necesaria para crear un manual de seguridad sobre la configuración de un servidor Proxy y Firewall, (a nivel de software) en el ambiente Linux en el cual se explicarán las definiciones de dichos servicios, se mostrará la instalación y los parámetros a configurar, usando SQUID 3.05 para el servicio de Proxy e IPTABLES para el Firewall. Se presentará información acerca de la seguridad que debe existir en toda red utilizando un Firewall y por que surge la necesidad de usar un Proxy. También se mostrarán las ventajas y desventajas del Sistema Operativo Linux en comparación al sistema operativo Windows

CAPITULO I: MARCO TEÓRICO DE REFERENCIA

1.1 Situación Problemática

El estudio de esta investigación, se debe a la falta de material didáctico comprensible, actualizado y practico, para los estudiantes de la carrera de: Técnico en Ingeniería de Redes sobre la temática de seguridad de redes en relación a la configuración de servicios de Proxy y Firewall en el ambiente del sistema operativo Linux, en la Universidad Tecnológica de El Salvador.

Al mismo tiempo se pretende con el manual establecer restricciones y normas de seguridad a las redes, haciendo así una red más sólida y segura pues al encontrarse constantemente conectada a Internet y manipulada por diferentes alumnos quienes envían y reciben información en pequeñas unidades, que contienen la dirección de quien envía el mensaje y del receptor. Pero al igual que con los correos electrónicos, no siempre se recibe en el ordenador todos los contenidos que se desea, ya que a veces, algunos son completamente innecesarios o peligrosos para el rendimiento del equipo.

La proliferación de las líneas ADSL o la fibra óptica que han mejorado la velocidad de conexión a Internet y la posibilidad de visitar más sitios en menos tiempo, también ha hecho que aumente el riesgo de que algún intruso se apodere de nuestra información confidencial o puede dejar inutilizado nuestro ordenador en cualquier momento.

1.2 Enunciado del problema

¿En que medida la elaboración de un manual didáctico Web sobre la configuración de servidor Proxy y Firewall bajo el Sistema Operativo Linux ayudara al aprendizaje de los estudiantes de la Carrera Técnico de Ingeniería en Redes Computacionales de la Universidad Tecnológica de El Salvador?

1.3 Justificación

Esta investigación proporcionará la información necesaria para poder adoptar los conocimientos necesarios para poder configurar los servicios de Firewall y Proxy, y a la vez poder familiarizarse con el entorno del sistema operativo Linux.

Este proyecto permitirá que los estudiantes adquieran el conocimiento necesario sobre la administración que una red debe tener para que sea segura y eficiente. Ampliando los conocimientos para manipular y comprender las definiciones del sistema operativo Linux, específicamente los servicios de Firewall y Proxy de forma práctica.

1.4 Objetivos

1.4.1 Objetivo General

- Elaborar un manual de instalación y configuración de un servidor Proxy y Firewall bajo la plataforma Linux a los alumnos de la Universidad Tecnológica de El Salvador el cual contenga configuración de servidor Proxy y Firewall que le sirva de apoyo y guía a la asignatura de Sistemas Operativos de la carrera Técnico en Ingeniería de Redes Computacionales

1.4.2 Objetivos Específicos

- Presentar las ventajas y desventajas del sistema operativo Linux como plataforma para los servicios de Proxy y Firewall

- Proporcionar los parámetros básicos de las configuraciones de los servidores de Proxy y Firewall

- Realizar una recopilación teórica de material en la cual se reflejen todos los parámetros de seguridad necesarios para la configuración de un servidor Proxy y firewall para compartir el servicio de Internet y dar seguridad a una red.

1.5 Alcances

- Que la Universidad Tecnológica de El Salvador cuente con un material de apoyo relacionado con la seguridad de Redes en ambiente Linux para los estudiantes de la carrera Técnico en Ingeniería de Redes Computaciones, para su mayor conocimiento.

- Creación de un manual de referencia bibliográfica Web para consultas de los alumnos de la carrera Técnico en Ingeniería de Redes Computacionales orientado a la materia de Redes de la Universidad Tecnológica sobre el tema en discusión.

- Dejar configuraciones generales sobre los parámetros de un servidor Proxy y Firewall manejando un sistema operativo Linux

1.6 Delimitaciones

- **Delimitación Organizacional:** La delimitación Organizacional de dicho proyecto se realizará en la Universidad Tecnológica de El Salvador, Escuela de Informática.

- **Delimitación Geográfica:** La delimitación geográfica del proyecto sobre el Manual Web de Configuración de un Servidor Proxy y Firewall bajo el Sistema Operativo Red Hat Linux se realizará en la Universidad Tecnológica de El Salvador, Calle Arce No. 1120, San Salvador, El Salvador Edif. Francisco Morazán.

- **Delimitación Temporal:** El proyecto sobre el Manual Web de Configuración de un Servidor Proxy y Firewall en el Sistema Operativo Red Hat Linux se realizará en 3 meses que comprende del mes de Marzo 2007 a Junio 2007.

- **Delimitación Específica**: La delimitación específica del proyecto sobre un Manual Web de Configuración de un Servidor Proxy y Firewall se delimitará específicamente en la Facultad de Informática y Ciencias aplicadas Escuela de Informática, Cátedra de Sistemas Operativos.

1.7 Estudios de Factibilidad

Para poder implementar este proyecto se necesitan cubrir varios factores entre los cuales ya está cubierto el económico, ya que la Universidad Tecnológica tiene los requerimientos para poderlo llevar a cabo, ya que el laboratorio de redes de la universidad ya tiene instalado el Sistema Operativo Red Hat Linux, así como el factor técnico, ya que se cuenta con todas las herramientas necesarias para la buena implementación del proyecto.

Factibilidad Técnica

Es factible para la Universidad Tecnológica de El Salvador porque consta con las computadoras, el Sistema Operativo y herramientas necesarias para realizar la configuración de un servidor Proxy y Firewall.

Factibilidad Económica

Es factible económicamente para la Universidad Tecnológica de El Salvador debido a que la licencia del Sistema Operativo y los programas a utilizar son gratuitos y no se requiere de una gran inversión económica.

Factibilidad Operativa

Únicamente se necesitaría tener el manual de configuración para poder llevar acabo este proyecto ya que en él está lo necesario.

1.8 Linux

Linux es un sistema operativo gratuito y de libre distribución inspirado en el sistema Unix, escrito por Linus Torvalds con la ayuda de miles de programadores en Internet. Unix es un Sistema Operativo desarrollado en 1970, una de cuya mayor ventaja, es que fácilmente portable a diferentes tipos de ordenadores, por lo que existen versiones de Unix para casi todos los tipos de ordenadores, desde PC y Mac hasta estaciones de trabajo y superordenadores. Al contrario que otros Sistemas Operativos, como por ejemplo MacOS (Sistema operativo de los Apple Macintosh), Unix no está pensado para ser fácil de emplear, sino para ser sumamente flexible. Por lo tanto Linux no es en general tan sencillo de emplear como otros sistemas operativos, aunque, se están realizando grandes esfuerzos para facilitar su uso. Pese a la enorme flexibilidad de Linux y su gran estabilidad (y el bajo coste) han hecho de este sistema operativo una opción mas para tener en cuenta por aquellos usuarios que se dediquen a trabajar a través de redes, naveguen por Internet, o se dediquen a la programación. Además el futuro de Linux es brillante y cada vez más gente y más empresas (entre otras IBM, Intel, Corel) están apoyando este proyecto, con lo que el sistema será cada vez más sencillo de emplear y los programas serán cada vez mejores.

1.8.1 ¿Qué son las distribuciones?

Una de los primeros conceptos que aparecen al iniciarse en Linux es el concepto de distribución.

Una distribución es un agrupamiento del núcleo del sistema operativo Linux (la parte desarrollada por Linus Torvalds) y otra serie de aplicaciones de uso general o no tan general. En principio las empresas que desarrollan las distribuciones de Linux están en su derecho al cobrar una cierta cantidad por el software que ofrecen, aunque en la mayor parte de las ocasiones se pueden conseguir estas distribuciones desde Internet, de revistas o de amigos, siendo todas estas formas gratuitas y legales.

Las distribuciones más conocidas son RedHat, Debian, Slackware, SuSE y Corel Linux, todas ellas incluyen el software más reciente y empleado lo cual incluye compiladores de C/C++, editores de texto, juegos, programas para el acceso a Internet, así como el entorno gráfico de Linux: X Windows.

A lo largo de este manual se considerará la distribución de Linux más extendida en la actualidad: RedHat 9.0, aunque la mayor parte de la información debe ser válida para el resto de las distribuciones, existen determinadas opciones que están sujetas a cambio como el sistema de instalación del sistema operativo.

1.8.2 Requisitos mínimos de Linux

La configuración mínima sobre la que Linux es capaz de funcionar es un procesador 386SX con 2Mb de memoria RAM y una disquetera de 1.44, aunque con estas características simplemente podremos arrancar el sistema y poco más. Una configuración más realista debería incluir 4Mb de RAM si no vamos a utilizar el entorno gráfico, 8Mb en caso contrario y un mínimo de 40MB de espacio en disco, aunque en las distribuciones actuales el espacio debería ser mayor de unos 300Mb. Cantidades ridículas si las comparamos con los dispositivos disponibles actualmente, pero que permiten reutilizar materiales más antiguos de los que se pueda disponer. Frente a estos requerimientos, se encuentran los sistemas Windows, con unos requerimientos desorbitados.

Linux soporta cualquier CPU compatible con los procesadores x86 de Intel, pudiéndose encontrar versiones que funcionan con procesadores 680x0 de Motorola utilizados en ordenadores Amiga y Atari. También son compatibles con Linux muchos ordenadores basados en Alpha, ciertas máquinas Sparc, las máquinas basadas en PowerPC como por ejemplo los ordenadores Macintosh de Apple, así como ARM, MIPS y algunos tipos de agendas electrónicas, teléfonos y consolas de juegos.

1.9 Windows o Linux, ventajas y desventajas

Cualquier sistema de computación requiere de un Sistema Operativo que sirva de intermediario entre el usuario y el hardware, que se ocupe de "arrancar" la computadora, que designe los recursos disponibles, que controle la seguridad, que ofrezca los entornos de trabajo y determine espacios en los discos.

Los sistemas existentes son varios y cada uno con sus características, ventajas y no tanto, ingresan en la lucha cotidiana por el dominio. Sin embargo, el enfrentamiento más fuerte está planteado entre dos ya eternos rivales: "Windows vs. Linux"

1.9.1 Razones para usar Linux:

Es más seguro

- Ya que la gran mayoría de los ataques de hackers son dirigidos a servidores Windows al igual que los virus los cuales se enfocan principalmente a servidores con éste sistema operativo.
- La plataforma Linux es más robusta lo cual hace más difícil que algún intruso pueda violar el sistema de seguridad de Linux.
- El manejo de la memoria de Linux evita que los errores de las aplicaciones detengan el núcleo de Linux.

Es más rápido

- Al tener una plataforma más estable, esto favorece el desempeño de aplicaciones de todo tipo tales como: bases de datos, aplicaciones XML, multimedia, etc.

- La eficiencia de su código fuente hace que la velocidad de las aplicaciones Linux sean superiores a las que corren sobre Windows lo cual se traduce en velocidad de su página.

- Linux permite navegar por Internet y conectar máquinas en red de manera natural (los protocolos TCP/IP o PPP por ejemplo, están incluidos como un módulo básico del núcleo)

- **Es más económico**
- Ya que requieren menor mantenimiento. En servidores Windows es más costoso debido a que es necesaria una frecuente atención y monitoreo contra ataques de virus, hackers y errores de código, instalación y actualización de parches y service packs.
- El software Linux así como también un sin número de aplicaciones son de código abierto (gratuitos).

- No requieren supervisión tan estrecha ni pagos de pólizas de mantenimiento necesarias para obtener los Service Packs.
- Puede funcionar en máquinas humildes: Linux puede correr servicios en un x86 a 200 MHz con calidad

- Linux ya no está restringido a personas con grandes conocimientos de informática: Los desarrolladores de Linux han hecho un gran esfuerzo por dotar al sistema de asistentes de configuración y ayuda, además de un sistema gráfico muy potente .Distribuciones Linux como Red Hat/Fedora tienen aplicaciones de configuración similares a las de Windows

- Linux es multitarea y multiusuario: Esta característica imprescindible está en Unix desde su concepción pero le llevó a Microsoft más de 20 años ofrecerlo en su sistema operativo de consumo

- Uno de los problemas de ambos sistemas es que Windows no puede escribir en particiones Linux o que desde Linux no se puede escribir.

1.9.2 Razones para usar Windows

Es más fácil

- Windows siendo el sistema operativo más comercial lo cual se refleja en la disponibilidad de aplicaciones, facilidad de mantenimiento así como soporte en el desarrollo de nuevas aplicaciones, puntos que pueden ser cruciales en la elección de servidores que corren aplicaciones Web.

- En la mayoría de distribuciones Linux hay que conocer nuestro Hardware a la hora de instalar
- Aplicaciones desarrolladas en menor tiempo
- Fruto de la inversión realizada por Microsoft y aunado a una comunidad de programadores cada vez más grande se ha logrado facilitar el desarrollo de aplicaciones y sistemas que corran sobre servidores Windows lo cual se ve reflejado en tiempos de desarrollo menores.
- La curva de aprendizaje en el sistema Windows es mucho menor.
- La instalación es muy sencilla y no requiere de mucha experiencia. Toda la información presentada al usuario es gráfica.

1.10 Proxy

Un Proxy (intermediario) es un programa o dispositivo que actúa como pasarela (gateway) entre redes, en el caso que nos ocupa entre la red local e Internet. Cada vez que alguno de los ordenadores de la red desea acceder a Internet realiza esta petición al Proxy. El Proxy utiliza la única conexión existente para enviar y recibir la información de Internet al ordenador que la solicita.

1.10.1 ¿Cómo funciona un Proxy?

Un Proxy permite a otros equipos conectarse a una red de forma indirecta a través de él. Cuando un equipo de la red desea acceder a una información o recurso, es realmente el Proxy quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. En unos casos esto se hace así porque no es posible la comunicación directa y en otros casos porque el Proxy añade una funcionalidad adicional, como puede ser la de mantener los resultados obtenidos (por eje.: una página Web) en una cache que permita acelerar sucesivas consultas coincidentes. Con esta denominación general de Proxy se agrupan diversas técnicas.

Desde el punto de vista del usuario de la red local, el sistema funciona como si tuviera realmente un acceso directo a Internet. El usuario accede inmediatamente desde su ordenador a una página Web o recibe su correo electrónico, sin siquiera saber que el Proxy existe.

En realidad, al abrir un programa como Internet Explorer o recoger el correo pendiente, la petición de servicio se realiza al Proxy, no al servidor de Internet. El Proxy es el encargado de redireccionar estas peticiones a la máquina correspondiente (el servidor de la página Web o el servidor de correo) y una vez recibida la información, de transmitirla al ordenador que la solicitó.

1.10.2 Servicios que ofrece un Proxy

El Proxy puede ofrecer a los ordenadores de la red local, todos los servicios disponibles en Internet incluyendo servicios avanzados de transmisión de audio y video

- Correo electrónico
- World Wide Web
- Transmisiones FTP
- Telnet
- News
- Soporte del protocolo UDP: aplicaciones de streaming del tipo de Real Audio
- IRC
- Socks
- DNS

1.10.3 Ventajas de un Proxy

- **Control**. Sólo el intermediario hace el trabajo real, por tanto se pueden limitar y restringir los derechos de los usuarios, y dar permisos sólo al Proxy.
- **Ahorro**. Sólo uno de los usuarios (el Proxy) ha de estar equipado para hacer el trabajo real.

- **Velocidad**. Si varios clientes van a pedir el mismo recurso, el Proxy puede hacer caché: guardar la respuesta de una petición para darla directamente cuando otro usuario la pida. Así no tiene que volver a contactar con el destino, y acaba más rápido.
- **Filtrado** de contenidos: El servidor Proxy puede hacer un filtrado de páginas o contenidos basándose en criterios de restricción establecidos por el administrador dependiendo valores y características de lo que no se permite, creando una restricción cuando sea necesario.
- **Modificación de contenidos**: Basándose en la misma función del filtrado, y llamado Privoxy, tiene el objetivo de proteger la privacidad en Internet, puede ser configurado para bloquear direcciones y Cookies por expresiones regulares y modifica en la petición el contenido.
- **Anonimato**. Si todos los usuarios se identifican como uno sólo, es difícil que el recurso accedido pueda diferenciarlos. Pero esto puede ser malo, por ejemplo cuando hay que hacer necesariamente la identificación.
- **Demanda a Usuarios**: Puede cubrir a un gran número de usuarios, para solicitar, a través de él, los contenidos Web.
- **Conexión automática (autodialing):** No es necesario que el ordenador que actúa como Proxy esté conectado permanentemente a Internet. Con esta función, cada vez que un usuario realiza una petición, el Proxy establece la conexión. Del mismo modo el proxy la desconecta cuando no hay ninguna petición, todo ello automáticamente.

1.10.4 Desventajas de un Proxy

- **Abuso**. Al estar dispuesto a recibir peticiones de muchos usuarios y responderlas, es posible que haga algún trabajo que no toque. Por tanto, ha de controlar quién tiene acceso y quién no a sus servicios, cosa que normalmente es muy difícil.

- **Carga**. Un Proxy ha de hacer el trabajo de muchos usuarios.

- **Intromisión**. Es un paso más entre origen y destino, y algunos usuarios pueden no querer pasar por el Proxy. Y menos si hace de caché y guarda copias de los datos, pues al almacenar las páginas y objetos que los usuarios solicitan puede suponer una violación de la intimidad para algunas personas.

- **Incoherencia**. Si hace de caché, es posible que se equivoque y dé una respuesta antigua cuando hay una más reciente en el recurso de destino. Las páginas mostradas pueden no estar actualizadas si éstas han sido modificadas desde la última carga que realizó el Proxy caché.

- **Irregularidad**. El hecho de que el Proxy represente a más de un usuario da problemas en muchos escenarios, en concreto los que presuponen una comunicación directa entre un emisor y un receptor (como TCP/IP).

1.11 Tipos de Proxy

1.11.1 Proxy de Web / Proxy cache de Web

Se trata de un Proxy para una aplicación específica: el acceso a la Web. Aparte de la utilidad general de un Proxy, proporciona una cache para las páginas web y los contenidos descargados, que es compartida por todos los equipos de la red, con la consiguiente mejora en los tiempos de acceso para consultas coincidentes. Al mismo tiempo libera la carga de los enlaces hacia Internet.

Funcionamiento

El cliente realiza una petición (p.e. mediante un navegador Web) de un recurso de Internet (una página Web o cualquier otro archivo) especificado por una URL.
Cuando el Proxy caché recibe la petición, busca la URL resultante en su caché local. Si la encuentra, devuelve el documento inmediatamente, si no es así, lo captura del servidor remoto, lo devuelve al que lo pidió y guarda una copia en su caché para futuras peticiones.

El caché utiliza normalmente un algoritmo para determinar cuándo un documento está obsoleto y debe ser eliminado de la caché, dependiendo de su antigüedad, tamaño e histórico de acceso. Dos de esos algoritmos básicos son el LRU (el usado menos

recientemente, en inglés "Least Recently Used") y el LFU (el usado menos frecuentemente, "Least Frequently Used").

Los proxies Web también pueden filtrar el contenido de las páginas Web servidas. Algunas aplicaciones que intentan bloquear contenido Web ofensivo están implementadas como proxies Web. Otros tipos de Proxy cambian el formato de las páginas Web para un propósito o una audiencia específicos, por ejemplo, mostrar una página en un teléfono móvil o una PDA. Algunos operadores de red también tienen proxies para interceptar virus y otros contenidos hostiles servidos por páginas remotas.

Ejemplo:

Un cliente de un ISP manda una petición a Google la petición llega en un inicio al servidor Proxy que tiene este ISP, no va directamente a la dirección IP del dominio de Google. Esta página concreta suele ser muy solicitada por un alto porcentaje de usuarios, por lo tanto el ISP la retiene en su Proxy por un cierto tiempo y crea una respuesta mucho menor en tiempo. Cuando el usuario crea una búsqueda el servidor Proxy ya no es utilizado y la red envía su petición y el cliente recibe su respuesta ahora sí desde Google.

1.11.2 Proxies transparentes

Muchas organizaciones (incluyendo empresas, colegios y familias) usan los proxies para reforzar las políticas de uso de la red o para proporcionar seguridad y servicios de caché. Normalmente, un Proxy Web o NAT no es transparente a la aplicación cliente: debe ser configurada para usar el Proxy, manualmente. Por lo tanto, el usuario puede evadir el Proxy cambiando simplemente la configuración.

Un Proxy transparente combina un servidor Proxy con NAT 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. Este es el tipo de Proxy que utilizan los proveedores de servicios de Internet (ISP). En España, la compañía más expandida en cuanto a ADSL se refiere, ISP Telefónica, dejó de utilizar Proxy transparente con sus clientes a partir de Febrero de 2006.

1.11.3 Reverse Proxy

Es un servidor Proxy instalado en el domicilio de uno o más servidores Web. Todo el tráfico entrante de Internet y con el destino de uno de esos servidores Web pasa a través del servidor Proxy. Hay varias razones para instalar un "reverse Proxy":

- **Seguridad:** el servidor Proxy es una capa adicional de defensa y por lo tanto protege los servidores Web.

- **Cifrado / Aceleración SSL**: cuando se crea un sitio Web seguro, habitualmente el cifrado SSL no la hace el mismo servidor Web, sino que es realizada por el "reverse Proxy", el cual está equipado con un hardware de aceleración SSL (Security Sockets Layer).

- **Distribución de Carga**: el "reverse Proxy" puede distribuir la carga entre varios servidores Web. En ese caso, el "reverse Proxy" puede necesitar reescribir las URL de cada página Web (traducción de la URL externa a la URL interna correspondiente, según en qué servidor se encuentre la información solicitada)

- **Caché de contenido estático**: Un "reverse Proxy" puede descargar los servidores Web almacenando contenido estático como imágenes y otro contenido gráfico.

1.11.4 Proxy NAT (Network Address Translation)

Otro mecanismo para hacer de intermediario en una red es el NAT. La traducción de direcciones de red (NAT, Network Address Translation) también es conocida como enmascaramiento de IPs. Es una técnica mediante la cual las direcciones fuente o destino de los paquetes IP son re-escritas, sustituidas por otras (de ahí el "enmascaramiento").

Esto es lo que ocurre cuando varios usuarios comparten una única conexión a Internet. Se dispone de una única dirección IP pública, que tiene que ser compartida. Dentro de la red de área local (LAN) los equipos emplean direcciones IP reservadas para uso privado y será el Proxy el encargado de traducir las direcciones privadas a esa única dirección pública para realizar las peticiones, así como de distribuir las páginas recibidas a aquel usuario interno que la solicitó.

Esta situación es muy común en empresas y domicilios con varios ordenadores en red y un acceso externo a Internet. El acceso a Internet mediante NAT proporciona una cierta seguridad, puesto que en realidad no hay conexión directa entre el exterior y la red privada, y así nuestros equipos no están expuestos a ataques directos desde el exterior.
Mediante NAT también se puede permitir un acceso limitado desde el exterior, y hacer que las peticiones que llegan al Proxy sean dirigidas a una máquina concreta que haya sido determinada para tal fin en el propio Proxy.

La función de NAT reside en los Cortafuegos (informática), y resulta muy cómoda porque no necesita de ninguna configuración especial en los equipos de la red privada que pueden acceder a través de él como si fuera un mero encaminador.

1.11.5 Proxy de Aplicación

Con un servidor Proxy de aplicación el proceso se automatiza. Cuando el usuario se comunica con el mundo exterior el cliente le envía al usuario primero al servidor Proxy. El servidor Proxy establece la comunicación con el servidor que ha solicitado (el mundo exterior) y le devuelve los datos.

1.11.6 Proxy De Socks

Un servidor Proxy SOCKS se parece bastante a un panel de conmutación. Tan sólo establece la conexión entre su sistema y otro sistema externo. La mayoría de los servidores SOCKS presentan el inconveniente de que sólo trabajan con conexiones del tipo TCP y como cortafuegos no suministran autenticación para los usuarios. Sin embargo, su ventaja es que registran los sitios a los que cada usuario se ha conectado.

Beneficios

- Acceso transparente a Internet, por medio de cualquier programa, y a cualquier servicio (Netscape, Opera, Internet Explorer, CuteFTP...).
- Mayor velocidad en la navegación: aquellas páginas que hayan sido visitadas serán guardadas en el servidor para que no haya que solicitarlas de Internet salvo que hayan cambiado.

- Optimización de uno o varios accesos ADSL que disponga.
- Posibilidad de un control absoluto de los accesos a Internet, por fecha, hora, lugar, e incluso persona.
- Capacidad de control de Páginas prohibidas

1.12 Squid

Squid es el software para servidor Proxy más popular y extendido entre los Sistemas Operativos basados sobre UNIX. Es muy confiable, robusto y versátil. Al ser software libre, además de estar disponible el código fuente, está libre del pago de costosas licencias por uso o con restricción a un uso con determinado número de usuarios.

Actualmente *Squid* es capaz de hacer Proxy-caché de los protocolos HTTP, FTP, GOPHER, SSL y WAIS. No soporta POP, NNTP, RealAudio y otros. En realidad, aunque es posible, no estamos utilizando Squid en Valliniello para que la red interna acceda a Internet, sino únicamente para acelerar el acceso a la WWW.

El servidor Proxy *Squid* guarda los datos cacheados en la memoria RAM, realiza caché de consultas DNS, y no guarda en el cache las peticiones que son rechazadas.

Squid también soporta 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, lo que es muy útil, por ejemplo en un centro educativo

para permitir/denegar acceso al servidor a diferentes grupos de usuarios. Al utilizar el protocolo de cache de Internet, squid puede ahorrar un considerable ancho de banda. Mejorando la velocidad de acceso a Internet en estos protocolos.

Este software consta de un programa servidor principal llamado squid, un programa búsqueda de nombres en el DNS llamado dnsserver, un programa para recuperar ficheros vía FTP, llamado ftpget, y algunas herramientas de mantenimiento y programas cliente. Cuando arranca, levanta un numero configurable de procesos dnsserver, cada uno de los cuales realiza una búsqueda en el sistema DNS, lo que reduce el tiempo que la caché espera por la resolución de nombres.

Squid es el resultado del esfuerzo de numerosas personas individuales de Internet y al igual que el sistema operativo *Linux* es gratis y tiene el código fuente disponible para poder modificarlo según las necesidades.

Software Requerido

Squid-3.05.STABLE1

Hardware Requerido

Requerimientos mínimos:

Servidor a 250 Mhz, 256 MB RAM, 8 GB disco duro.

1.13 Firewall

Un firewall es un dispositivo o un software que funciona como barrera defensiva entre redes, permitiendo o denegando las transmisiones de una red a la otra. Un uso típico es situarlo entre una red local e Internet, como dispositivo de seguridad para evitar que los intrusos puedan acceder a información confidencial.

Para permitir o denegar una comunicación el Firewal examina el tipo de servicio al que corresponde, como pueden ser el Web, el correo o el IRC. Dependiendo del servicio decide si lo permite o no. Además, examina si la comunicación es entrante o saliente y dependiendo de su dirección puede validarla o no.

De este modo, un firewall puede abrir desde una red local hacia Internet servicios de Web, correo y FTP, pero no a IRC que puede ser innecesario para nuestro trabajo.

Actualmente, podemos encontrar dos tipos de Firewall: de software y de hardware. Estos últimos son los más caros, pero también los mejores. Ambos, se basan en el filtrado de información.

1.13.1 Firewall de software

Estos programas son los más comunes en los hogares, ya que a parte de resultar mucho más económicos que el hardware, su instalación y actualización es más sencilla. Eso sí, presentan algunos problemas inherentes a su condición: consumen recursos del

ordenador, algunas veces no se ejecutan correctamente o pueden ocasionar errores de compatibilidad con otro software instalado.

Actualmente, los sistemas operativos más modernos como Windows XP y Linux integran soluciones básicas de firewall, en algunos casos, como en el software libre, son muy potentes y flexibles, pero requieren de un conocimiento en redes y puertos necesarios para las aplicaciones.

1.13.2 Seguridad apoyada en hardware

Los firewall de hardware se utilizan más en empresas y grandes corporaciones. Normalmente son dispositivos que se colocan entre el router y la conexión telefónica. Como ventajas, podemos destacar, que al ser independientes del PC, no es necesario configurarlos cada vez que reinstalamos el sistema operativo, y no consumen recursos del sistema. Su mayor inconveniente es el mantenimiento, ya que son difíciles de actualizar y de configurar correctamente

Para evitar en lo posible los intrusos o ataques, surgieron los firewall, una especie de murallas defensivas contra los “cracker” que actúan en la Red.

Podemos definir como ataques, todas aquellas acciones que suponen una violación de la seguridad de nuestro sistema, confidencialidad, integridad o disponibilidad.

Estas acciones se pueden clasificar de modo genérico según los efectos causados, como:

- Interrupción: cuando un recurso del sistema es destruido o se vuelve no disponible.
- Intercepción: una entidad no autorizada consigue acceso a un recurso.
- Modificación alguien no autorizado consigue acceso a una información y es capaz de manipularla.
- Fabricación: cuando se insertan objetos falsificados en el sistema.

  También se pueden ordenar por modalidades de ataque según la forma de actuar:

- Escaneo de puertos: esta técnica consiste en buscar puertos abiertos, y fijarse en los que puedan ser receptivos o de utilidad.
- Ataques de autentificación: cuando un atacante suplanta a una persona con autorización.
- Explotación de errores: suceden en el momento que se encuentran agujeros de seguridad en los sistemas operativos, protocolos de red o aplicaciones.
- Ataques de denegación de servicio (DOS): consiste en saturar un servidor con pedidos falsos hasta dejarlo fuera de servicio.

1.14 Iptables

IPtables es un sistema de firewall vinculado al kernel de Linux que se ha extendido enormemente a partir del kernel 2.4 de este sistema operativo. Al igual que el anterior sistema ipchains, un firewall de iptables no es como un servidor que se inicia o se detiene o que se pueda caer por un error de programación (esto es una pequeña mentira, ha tenido alguna vulnerabilidad que permite DoS, pero nunca tendrá tanto peligro como las aplicaciones que escuchan en determinado puerto TCP): iptables esta integrado con el kernel, es parte del sistema operativo. ¿Cómo se pone en marcha? Realmente lo que se hace es aplicar reglas. Para ellos se ejecuta el comando iptables, con el que se añade, borra, o crean reglas. Por ello un firewall de iptables no es sino un simple script de shell en el que se van ejecutando las reglas de firewall.

Iptables es una forma de indicarle al kernel algunas cosas que debe hacer con cada paquete, esto se hace en base a las características de un paquete en particular. Los paquetes de red tienen muchas características, algunas pueden ser los valores que tienen en sus encabezados (a donde se dirigen, de donde vienen, números de puertos, etc., etc.), otra puede ser el contenido de dicho paquete (la parte de datos), y existen otras características que no tienen que ver con un paquete en particular sino con una sumatoria de ellos. La idea es lograr identificar un paquete y hacer algo con el mismo.

Iptables es un conjunto de comandos que permiten decirle al kernel qué hacer con ciertos paquetes que cumplan con ciertas características'.

Ahora bien, las acciones que se puede indicar con iptables no son tan amplias, con iptables se va a poder decirle al kernel que acepte un paquete (lo deje pasar), lo deniegue (lo descarte), lo rechace, lo marque o lo modifique. Hay ciertas cosas que no se va a poder indicarle, por ejemplo que lo ``reenvié" por alguna ruta estática TCP/IP definida. Con lo cual, con iptables se va a poder filtrar o modificar paquetes, nada más.

En iptables, se tienen varios tipos de reglas: reglas de filtrado, reglas de NAT, reglas de mangle (para manipular paquetes)...se concentrará en las reglas de filtrado, para poder filtrar tráfico y implementar nuestro firewall .Estas reglas se definen en la tabla filter (que es la tabla por defecto de iptables).

CAPITULO II: DOCUMENTACION TECNICA Y ECONOMICA

2.1 Proyecto temático

Diseñar un manual didáctico Web en la Universidad Tecnológica de El Salvador el cual servirá de apoyo en la Facultad de Informática y Ciencias Aplicadas, en la cátedra de redes. El proyecto permitirá conocer las instalaciones y configuraciones de los servicios de Firewall y Proxy en el sistema operativo Linux, definiciones de dichos servicios, sus ventajas y desventajas de Linux sobre Windows.

El proyecto comprende los siguientes servicios:

- Servicios de Firewall

En este lo que se hará es convocar el servicio con el comando IPtables para poner en marcha el firewall y seguidamente modificar su configuración

- Servicios de Proxy

  Se requerirá descargar el software Squid 3.05 para poner en marcha el servidor Proxy después de su descarga se descomprime y se compila, después de estar instalado se procede a modificar los parámetros

2.2 Documentación técnica

2.2.1 Red Hat Linux

Red Hat es una distribución Linux creada por Red Hat, que fue una de las más populares en los entornos de usuarios domésticos.

Es una de las distribuciones Linux de "mediana edad". La versión 1.0 fue presentada el 3 de noviembre de 1994. No es tan antigua como la distribución Slackware, pero ciertamente es más antigua que muchas otras. Fue la primera distribución que usó RPM como su formato de paquete, y en un cierto plazo ha servido como el punto de partida para varias otras distribuciones, tales como la orientada hacia PCs de escritorio Mandrake Linux (originalmente Red Hat Linux con KDE), Yellow Dog Linux, la cual se inició desde Red Hat Linux con soporte para PowerPC, y ASPLinux (Red Hat Linux con mejor soporte para caracteres no-Latinos).

Desde el 2003, Red Hat ha desplazado su enfoque hacia el mercado de los negocios con la distribución Red Hat Enterprise Linux y la versión no comercial Fedora Core. Red Hat Linux 9, la versión final, llegó oficialmente al final de su vida útil el 30 de abril de 2004, aunque el proyecto Fedora Legacy continuó publicando actualizaciones, hasta ser abandonado dicho proyecto a finales de 2006.

2.2.2 Características especiales

Red Hat es instalado con un ambiente gráfico llamado *Anaconda*, diseñado para su fácil uso por novatos. También incorpora una herramienta llamada *Lokkit* para configurar las capacidades de Cortafuegos.

Al igual que en el Red Hat Linux 8.0, fue habilitado como el sistema de codificación de tipografías para el sistema. Esto tiene poco efecto en usuarios angloparlantes, pero cuando se usa la parte superior del juego de caracteres ISO 8859-1, éstos se codifican de manera radicalmente diferente. Esto puede ser visto, por ejemplo, por usuarios que hablan francés o sueco como algo agresivo, pues sus antiguos sistemas de archivo lucen muy diferentes y pueden ser luego inutilizables. Puede deshacerse este cambio quitando la parte ".UTF-8" de la configuración de lenguaje.

La versión 8.0 fue además la primera en incluir el entorno de escritorio gráfico Bluecurve.

Red Hat Linux carece de muchas características debido a posibles problemas de copyright y patentes. Por ejemplo, el soporte al formato MP3 está desactivado tanto en Rhythmbox como en XMMS; en su lugar, Red Hat recomienda usar Ogg Vorbis, que no tiene patentes. Sin embargo, el soporte para MP3 puede ser instalado luego, aunque se requiere el pago de regalías en los Estados Unidos. El soporte al formato NTFS también está ausente, pero también puede ser instalado libremente.

Red Hat Linux

Desarrollador Red Hat

Familia de S.O. Linux

Modelo de desarrollo Open source **Núcleo** Linux

Tipo de núcleo Monolítico

Licencia GPL **Última versión estable** 9 / 31 de marzo de 2003

Estado actual Sin desarrollo

Sitio web www.redhat.com

2.3 Squid

Squid es un popular programa de software libre que implementa un servidor Proxy y un *demon* para Web caché, publicado bajo licencia GPL. Tiene una amplia variedad de utilidades, desde acelerar un Servidor Web. Guardando en caché peticiones repetidas, hasta caché de Web, DNS y otras búsquedas para un grupo de gente que comparte recursos de la red, además de añadir seguridad filtrando el tráfico. Está especialmente diseñado para ejecutarse bajo entornos Unix-like.

Squid ha sido desarrollado durante muchos años y se le considera muy completo y robusto. Soporta muchos protocolos, aunque se usa principalmente para HTTP y FTP. Se añade soporte también a TLS, SSL, Internet Gopher y HTTPS.

2.3.1 Características de Squid

1. Proxy y Caché de HTTP, FTP, y otras URLs
2. Proxy para SSL
3. Jerarquías de Caché
4. ICP, HTCP, CARP, Caché Digests
5. Caché transparente
6. WCCP (Squid v2.3 y superior)
7. Control de acceso
8. Aceleración de servidores HTTP
9. SNMP
10. Caché de resolución DNS.

2.3.2 Compatibilidad

Squid puede ejecutarse en los siguientes Sistemas Operativos:

- Linux
- FreeBSD
- OpenBSD
- NetBSD
- BSDI
- Mac OS X
- OSF and Digital Unix

- IRIX
- SunOS/Solaris
- NeXTStep
- SCO Unix
- AIX
- HP-UX
- Las versiones más recientes de Squid también compilan en Windows en sus versiones NT/2000/XP/2003 usando los paquetes Cygwin/GnuWin32.

2.4 Iptables

Iptables es el nombre de la herramienta de espacio de usuario mediante la cual el administrador puede definir políticas de filtrado del tráfico que circula por la red. El nombre *iptables* se utiliza frecuentemente de forma errónea para referirse a toda la infraestructura ofrecida por el proyecto Netfilter. Sin embargo, el proyecto ofrece otros subsistemas independientes de iptables tales como el *connection tracking system* o sistema de seguimiento de conexiones, o *queue*, que permite encolar paquetes para que sean tratados desde espacio de usuario. *iptables* es un software disponible en prácticamente todas las distribuciones de Linux actuales.

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

El paquete de los ip6tables también incluye ip6tables**.** ip6tables se utiliza para configurar el filtro del paquete IPv6.

Dependencias

Los iptables requieren un núcleo que ofrezca el filtro del paquete de los ip_tables. Esto incluye todo el 2.4.x y el núcleo 2.6.x lanza.

2.4.1 Características principales de Iptables

- Enumerando el contenido del paquete filtra el ruleset
- La adición/el quitar/que se modifica gobierna en el ruleset del filtro del paquete
- El enumerar/que pone a cero los contadores de la por-regla del paquete filtra el ruleset.

2.5 Evaluación técnica y económica

2.5.1 Evaluación técnica

caracteristicas	Windows	Linux
Estabilidad	85%	95%
Seguridad	80%	95%
Eficiencia	85%	90%
Eficaz	90%	90%
costos	40%	100%
Robustes	80%	95%
Consumo de recursos	90%	50%
Licenciamiento	50%	100%

2.5.2 Evaluación Económica

Linux fue escogido por obtener mayor rendimiento en todas las áreas, demostrando que es un sistema operativo más seguro, estable, rápido, eficaz y eficiente entre otros, y sobre todo una de sus características más conocidas es que es de código abierto (open-course) y es completamente gratuito

En este cuadro se hará mención de la inversión que se utilizará en los recursos a utilizar en este proyecto.

Insumos tecnológicos	Licencia
Licencia de RedHat Linux	Gratuita
Software de Proxy (Squid)	Gratuita

Materiales de Oficina	Precio
Cartucho negro para impresor	$ 14.00
Resma de papel Bond	$ 6.50
Lapiceros	$ 1.00
Cuadernos	$ 2.00

Servicios Gráficos	
Fotocopias	$ 20.00
anillados	$ 7.00
Empastado	$ 60.00

total	$ 110.50

2.6 Tecnología y recursos seleccionados

2.6.1 Listado de tecnología y recursos seleccionados

- RedHat 9.0
- Squid 3.05
- Iptables

2.6.2 Justificación de tecnología y recursos seleccionados

- **RedHat 9.0 Linux**

Ya que este sistema operativo es gratuito y de libre distribución, es una plataforma segura, estable, es muy reconocido para servicios de redes, no requiere de grandes recursos de Hardware.

- **Squid 3.05**

El software para servidor Proxy, es muy confiable, robusto y versátil, es un software gratuito y de fácil configuración.

- **Iptables**

Un sistema de firewall, no es un servidor que lo iniciamos o detenemos o que se pueda caer por un error de programación, iptables está integrado con el kernel, es parte del sistema operativo, lo que se hace es aplicar reglas, estas reglas se definen en la tabla filter.

2.6.3 CRONOGRAMA DE ACTIVIDADES A DESARROLLAR

ID	Fases	Marzo			Abril		Mayo		Junio		Julio			Agosto		Sept
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
1	Recolección de información															
2	Selección de material															
3	Tratamiento de la información															
4	Redacción preliminar															
5	Revisión, critica y correcciones															
6	Entrega del primer avance															
7	Corrección del primer informe															
8	Recolección de información técnica															
9	Redacción preliminar, segundo avance															
10	Revisión, critica y correcciones															
11	Entrega del segundo avance															
12	Corrección del segundo avance															
13	Preparación del prototipo del proyecto															
14	Preparación de defensa															
15	Defensa															
16	Corrección del trabajo después de la defensa															
17	Presentación de trabajo impreso															

CAPITULO III: PROTOTIPO

3.1 Configuración servidor Proxy (Compilación e instalación de Squid)

1. Se descomprime el archivo:

% tar zxvf squid-3.05-src.tar.gz

2. se ingresa al directorio creado con el paso anterior:

% cd squid-3.05

3. Se configura y compila squid:

% ./configure --prefix=/usr/local/squid

% make all

El parámetro *--prefix=/usr/local/squid* en ./configure solo indica el directorio donde será instalado squid (/usr/local/squid).

4. Se instala squid (con permisos de root):

% make install

3.2 Configuración SQUID.

Squid utiliza el fichero de configuración localizado en */etc/squid/squid.conf*, y se podrá trabajar sobre este utilizando un editor de texto. Los parámetros a configurar:

- http_port
- cache_dir
- Al menos una Lista de Control de Acceso

- Al menos una Regla de Control de Acceso
- httpd_accel_host
- httpd_accel_port
- httpd_accel_with_proxy

3.2.1 Parámetro http_port

Las asignaciones hechas por IANA (Internet Assigned Numbers Authority) y continuadas por la ICANN (Internet Corporation for Assigned Names and Numbers) desde el 21 de marzo de 2001, los Puertos Registrados (rango desde 1024 hasta 49151) recomendados para Servidores Intermediarios (Proxies) pueden ser el 3128 y 8080 a través de TCP.

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.

En el caso de un Servidor Intermediario (Proxy) Transparente, regularmente se utilizará el puerto 80 o el 8000 y se valdrá del re-direccionamiento de peticiones de modo tal que no habrá necesidad alguna de modificar la configuración de los clientes HTTP para utilizar el Servidor Intermediario (Proxy). Bastará con utilizar como puerta de enlace al servidor

Hoy en día puede no ser del todo práctico el utilizar un Servidor Intermediario (Proxy) Transparente, a menos que se trate de un servicio de 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 Intermediario (Proxy) con restricciones por clave de acceso, lo cual no puede hacerse con un Servidor Intermediario (Proxy) Transparente, debido a que se requiere un diálogo de nombre de usuario y clave de acceso.

Regularmente algunos programas utilizados comúnmente por los usuarios suelen traer de modo predefinido el puerto 8080 (servicio de cacheo WWW) para utilizarse al configurar que Servidor Intermediario (Proxy) utilizar. Si queremos aprovechar esto en 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:

```
# Default: http_port 3128
http_port 3128
http_port 8080
```

Si se desea incrementar la seguridad, puede vincularse el servicio a una IP que solo se pueda acceder desde la red local. Considerando que el servidor utilizado posee una IP 192.168.1.254, puede hacerse lo siguiente:

```
# Default: http_port 3128
http_port 192.168.1.254:3128
http_port 192.168.1.254:8080
```

3.2.2 Parámetro cache_mem.

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

- Objetos en tránsito.
- Objetos frecuentemente utilizados (Hot).
- Objetos negativamente almacenados en el caché.

Los datos de estos objetos se almacenan en bloques de 4 Kb. 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.

De modo predefinido se establecen 8 MB. 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 128 MB de RAM, establezca 16 MB como valor para este parámetro:

```
cache_mem 16 MB
```

3.2.3 Parámetro cache_dir:

Este parámetro se utiliza para establecer que tamaño se desea que tenga el caché en el disco duro para Squid. De modo predefinido Squid utilizará un caché de 100 MB, 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 se desee. Mientras más grande sea el caché, más objetos se 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 directorios subordinados con 256 niveles cada uno. No modifique esto números, no hay necesidad de hacerlo.

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

3.2.4 Vida en el caché

Para configurar el tiempo que pueden permanecer los objetos almacenados en el caché. De modo general, si definimos un tiempo de permanencia demasiado bajo, estaremos desaprovechando una de las principales ventajas del uso de servidor Proxy, mientras que si se establece un periodo demasiado alto, también se satura innecesariamente la capacidad de almacenaje.

Parece una decisión razonable, en la mayoría de casos, fijar un mes de vida para los objetos del caché. Esto se logra con la instrucción:

reference_age 1 month

3.2.5 Parámetro ftp_user.

Al acceder a un servidor FTP de manera anónima, de modo predefinido Squid enviará como clave de acceso 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 clave de acceso, puede especificarse la dirección de correo electrónico que uno considere pertinente.

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

3.2.6 Controles de acceso.

Es necesario establecer Listas de Control de Acceso que definan una red o bien ciertas máquinas en particular. A cada lista se le asignará una Regla de Control de Acceso que permitirá o denegará el acceso a Squid.

3.2.7 Listas de control de acceso.

Una lista de control de acceso se establece con la siguiente sintaxis:

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

Si se desea establecer una lista de control de acceso que abarque a toda la red local, basta definir la IP correspondiente a la red y la máscara de la sub-red. Por ejemplo, si se tiene una red donde las máquinas tienen direcciones IP 192.168.1.n con máscara de sub-red 255.255.255.0, podemos utilizar lo siguiente:

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

También puede definirse una Lista de Control de Acceso especificando un fichero localizado en cualquier parte del disco duro, y la cual contiene una lista de direcciones IP. Ejemplo:

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

El fichero /etc/squid/permitidos contendría algo como siguiente:

```
192.168.1.2
192.168.1.3
192.168.1.16
192.168.1.40
```

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.

3.2.8 Reglas de Control de Acceso.

Estas definen si se permite o no el acceso hacia 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 consideramos 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 no. 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 un gran grupo de 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.

3.2.9 Aplicando listas y reglas de control de acceso.

Se dispone de una red 192.168.1.0/255.255.255.0, se desea definir toda la red local, se utilizara la siguiente línea en la sección de Listas de Control de Acceso:

```
acl todalared src 192.168.1.0/255.255.255.0
```

Habiendo hecho lo anterior, la sección de listas de control de acceso debe quedar del siguiente modo:

Listas de Control de Acceso: definición de una red local completa

```
#
# 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 todalared src 192.168.1.0/255.255.255.0
```

A continuación se procederá a aplicar la regla de control de acceso:

```
http_access allow todalared
```

La zona de reglas de control de acceso quedara de este modo:

Reglas de control de acceso: Acceso a una Lista de Control de Acceso.

```
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM
YOUR CLIENTS
#
http_access allow localhost
http_access allow todalared
http_access deny all
```

La regla http_access allow todalared permite el acceso a Squid a la Lista de Control de Acceso denominada todalared, la cual está conformada por 192.168.1.0/255.255.255.0. Esto significa que cualquier máquina desde 192.168.1.1 hasta 192.168.1.254 podrá acceder a Squid.

Ejemplo 2

Si se quiere permitir el acceso a Squid a ciertas direcciones IP de la red local, se debe crear un fichero que contenga dicha lista. Genere el fichero /etc/squid/listas/redlocal, dentro del cual se incluirán solo aquellas direcciones IP que desea confirmen la Lista de Control de acceso.

```
192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.15
192.168.1.16
192.168.1.20
192.168.1.40
```

Se le denominara a esta lista de control de acceso como redlocal:

```
acl redlocal src "/etc/squid/listas/redlocal"
```

Habiendo hecho lo anterior, la sección de listas de control de acceso quedara del siguiente modo:

Listas de Control de Acceso: definición de una red local completa

```
#
# 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 "/etc/squid/listas/redlocal"
```

Explicación de la regla de control de acceso:

```
http_access allow redlocal
```

La zona de reglas de control de acceso quedara de este modo:

Reglas de control de acceso: Acceso a una Lista de Control de Acceso.

```
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM
YOUR CLIENTS
#
http_access allow localhost
http_access allow redlocal
http_access deny all
```

La regla http_access allow redlocal permite el acceso a Squid a la Lista de Control de Acceso denominada redlocal, la cual está conformada por las direcciones IP

especificadas en el fichero /etc/squid/listas/redlocal. Esto significa que cualquier máquina no incluida en /etc/squid/listas/redlocal no tendrá acceso a Squid.

3.2.10 Controles de acceso (ACL)

Squid es capaz de permitir o denegar accesos de navegadores a ciertas páginas, no permitir que los navegadores les hagan peticiones de ciertas páginas.. Utiliza listas de control de acceso ACL:

3.2.11 Parámetro HTTP_ACCESS

Con este parámetro especificamos qué navegadores (u otros Proxys) se podrán conectar a nuestro Proxy para realizar peticiones HTTP:

Ejemplo:

acl ourallowedhosts src 196.4.160.0/255.255.255.0

acl all src 0.0.0.0/0.0.0.0

http_access allow ourallowedhosts

http_access deny all

Con esta serie de "instrucciones" se definen primero las listas de control de acceso (acl …) y segundo se permite acceso (http_access) a los hosts 196.4.160.* y luego denegamos acceso a todos los demás. Es decir sólo pueden usar nuestro Proxy unos determinados IP.

También se puede usar para denegar acceso desde nuestro Proxy a "ciertas" páginas Web. Si no quisiéramos que los navegadores que se conectan a nuestro Proxy lo usen para conectarse a esas páginas:

acl adult dstdomain .playboy.com .sex.com

acl ourallowedhosts src 196.4.160.0/255.255.255.0

acl all src 0.0.0.0/0.0.0.0

http_access deny adult

http_access allow ourallowedhosts

http_access deny all

De esta forma nuestro proxy jamás solicitará páginas a dominios como playboy.com o sex.com, en su lugar mostrará un mensaje de error.

Nota: Hay que tener cuidado con el orden, si pusiéramos primero el "deny all" las demás líneas de "http_access" no tendrían efecto puesto que ya hemos denegado a todo el mundo (en cuanto una regla encaja dejar de mirar). Primero permitimos acceso y luego denegamos al resto.

3.2.12 Parámetro chache_mgr.

De modo predefinido, si algo ocurre con el caché, como por ejemplo que muera 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 JReyes@midominio.net
```

3.2.13 Parámetro *cache_peer*: caches padres y hermanos.

El parámetro *cache_peer* se utiliza para especificar otros Servidores Intermediarios (Proxies) con caché en una jerarquía como padres o como hermanos. Es decir, definir si hay un Servidor Intermediario (Proxy) adelante o en paralelo. La sintaxis básica es la siguiente:

```
cache_peer servidor tipo http_port icp_port opciones
```

Ejemplo: Si el caché va a estar trabajando detrás de otro servidor cache, es decir un caché padre, y considerando que el caché padre tiene una IP 192.168.1.1, escuchando peticiones HTTP en el puerto 8080 y peticiones ICP en puerto 3130 (puerto utilizado de modo predefinido por Squid), especificando que no se almacenen en caché los objetos que ya están presentes en el caché del Servidor Intermediario (Proxy) padre, utilice la siguiente línea:

```
cache_peer 192.168.1.1 parent 8080 3130 proxy-only
```

Cuando se trabaja en redes muy grandes donde existen varios Servidores Intermediarios (Proxy) haciendo caché de contenido de Internet, es una buena idea hacer trabajar todos los caché entre si. Configurar caches vecinos como sibling (hermanos) tiene como beneficio el que se consultarán estos caches localizados en la red local antes de acceder

hacia Internet y consumir ancho de banda para acceder hacia un objeto que ya podría estar presente en otro caché vecino.

Ejemplo:

Si el caché va a estar trabajando en paralelo junto con otros caches, es decir caches hermanos, y considerando los caches tienen IP 10.1.0.1, 10.2.0.1 y 10.3.0.1, todos escuchando peticiones HTTP en el puerto 8080 y peticiones ICP en puerto 3130, especificando que no se almacenen en caché los objetos que ya están presentes en los caches hermanos, utilice las siguientes líneas:

```
cache_peer 10.1.0.1 sibling 8080 3130 proxy-only
cache_peer 10.2.0.1 sibling 8080 3130 proxy-only
cache_peer 10.3.0.1 sibling 8080 3130 proxy-only
```

Pueden hacerse combinaciones que de manera tal que se podrían tener caches padres y hermanos trabajando en conjunto en una red local. Ejemplo:

```
cache_peer 10.0.0.1 parent 8080 3130 proxy-only
cache_peer 10.1.0.1 sibling 8080 3130 proxy-only
cache_peer 10.2.0.1 sibling 8080 3130 proxy-only
cache_peer 10.3.0.1 sibling 8080 3130 proxy-only
```

3.2.14 Caché con aceleración.

Cuando un usuario hace petición hacia un objeto en Internet, este es almacenado en el caché de Squid. 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 de Squid y además optimiza enormemente la utilización del ancho de banda.

La configuración de Squid como Servidor Intermediario (Proxy) Transparente solo requiere complementarse utilizando una regla de iptables que se encargará de re-direccionar peticiones haciéndolas pasar por el puerto 8080. La regla de iptables necesaria se describe más adelante en este documento.

3.2.15 Proxy Acelerado: Opciones para Servidor Intermediario (Proxy) en modo convencional.

En la sección HTTPD-ACCELERATOR OPTIONS deben habilitarse los siguientes parámetros:

```
httpd_accel_host virtual
httpd_accel_port 0
httpd_accel_with_proxy on
```

Proxy Acelerado: Opciones para Servidor Intermediario (Proxy) Transparente. Si se trata de un Servidor Intermediario (Proxy) transparente, deben utilizarse las siguientes

opciones, considerando que se hará uso del caché de un servidor HTTP (Apache) como auxiliar:

```
# Debe especificarse la IP de cualquier servidor HTTP en la
# red local o bien el valor virtual
httpd_accel_host 192.168.1.254
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Proxy Acelerado: Opciones para Servidor Intermediario (Proxy) Transparente para redes con Internet Exlorer 5.5 y versiones anteriores.

Si va a utilizar Internet Explorer 5.5 y versiones anteriores con un Servidor Intermediario (Proxy) transparente, es importante recuerde que dichas versiones tiene un pésimo soporte con los Servidores Intermediarios (Proxies) transparentes imposibilitando por completo la capacidad de refrescar contenido. Si se utiliza el parámetro ie_refresh con valor on puede hacer que se verifique en los servidores de origen para nuevo contenido para todas las peticiones IMS-REFRESH provenientes de Internet Explorer 5.5 y versiones anteriores.

```
# Debe especificarse la IP de cualquier servidor HTTP en la
# red local
httpd_accel_host 192.168.1.254
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
Ie_refresh on
```

Lo más conveniente es actualizar hacia Internet Explorer 6.x o definitivamente optar por otras alternativas. Mozilla es en un conjunto de aplicaciones para Internet, o bien Firefox, que es probablemente el mejor navegador que existe en el mercado. Firefox es un navegador muy ligero y que cumple con los estándares, y está disponible para Windows, Linux, Mac OS X y otros sistemas operativos.

3.2.16 Estableciendo el idioma de los mensajes mostrados por Squid hacia el usuario.

Squid incluye traducción a distintos idiomas de las distintas páginas de error e informativas que son desplegadas en un momento dado durante su operación. Dichas traducciones se pueden encontrar en /usr/share/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/share/squid/errors/Spanish en lugar de hacerlo hacia /usr/share/squid/errors/English.

Elimine primero el enlace simbólico actual:

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

Coloque un nuevo enlace simbólico apuntando hacia el directorio con los ficheros correspondientes a los errores traducidos al español.

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

Nota: Este enlace simbólico debe verificarse, y regenerarse de ser necesario, cada vez que se actualizado Squid ya sea a través de yum, up2date o manualmente con el mandato rpm.

3.2.17 Iniciando, reiniciando y añadiendo el servicio al arranque del sistema.

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

```
service squid Stara
```

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

```
service squid restart
```

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

```
chkconfig squid on
```

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

3.2.18 Depuración de errores

Cualquier error al inicio de Squid solo significa que hubo errores de sintaxis, errores de dedo o bien se 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
```

3.3 Configuración servidor firewall

3.3.1 Sintaxis

- *-A* «cadena»

Agrega una regla al final de la lista de reglas de la la cadena especificada

- *-D* «cadena» número_de_regla

Elimina una regla de la cadena especificada simplemente indicando el número de la regla.

- *-d {dirección/máscara}*

Indica que el paquete va destinado a la dirección

- *-dport «puerto»*

Indica el número el puerto de destino

- *-E vieja-cadena nueva-cadena*

Cambia cadena por una nueva

- *-F «cadena»*

Borra todas las reglas de la cadena especificada o todas las reglas si no se especifica ninguna cadena.

- *-f «cadena»*

Indica que la regla se aplicará a la segunda y sucesivas partes de un paquete fragmentado.

- -h

Describe la sintaxis de los comandos

- -I «cadena»

Inserta una o más reglas al comienzo de la cadena especificada

- -i «interfase»

Indica la interfase de entrada por donde se recibirá el paquete (en la especificación de interfases se puede utilizar el símbolo ``+" como metacaracter, por ejemplo ``eth+" que contempla eth0, eth1, eth2, etc.) se usa con reglas INPUT y FORWARD

- -j «acción o cadena»

Indica que acción se tomará cuando se encuentre una coincidencia con dicha regla.

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.

- -L «cadena»

Permite ver las reglas de la cadena especificada. Si no se indica ninguna cadena, mostrará las reglas de todas las cadenas.

- --line-numbers

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

- -m «módulo»

Indica que se va a utilizar un módulo en particular (si bien esto no hace a la especificación del paquete, se debe indicar para que el iptables entienda qué funcionalidad extra deberá utilizar con el fin de ``matchear'' el paquete)

La explícita: -m {tcp,udp,icmp,mac,mark,multiport,owner,limit,tos,ttl,...} o directamente usando la opción implícita y que el propio iptables se encargue de cargar las extensiones, como por ejemplo especificar directamente: "-p tcp" que provocará que se cargue la extensión tcp o lo que sería equivalente: "-m tcp -p tcp"

- -N cadena

Permite crear una cadena creada por el usuario.

- -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).

- -nL

Lista todas las reglas de la tabla indicada con -t «tabla»

- -o «interfase»

Indica la interfase por donde saldrá el paquete. (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 éste nombre será matcheada. Si esta opción se omite, se matcheará todo nombre de interfaz. Se usa con reglas FORWARD y OUTPUT.

- -P cadena política

Aplica una determinada política por defecto a la cadena especificada como primer argumento. Posibles políticas son ACCEPT, DROP, QUEUE y RETURN.

- -p «protocolo»

Indica el protocolo del paquete (los más comunes son ``tcp", ``udp", ``icmp", pero hay muchos tipos de protocolos soportados)

- -R «cadena» «pos»

Mover una regla a otra posición dentro de una cadena.

- -s «dirección/máscara »

Indica la dirección de origen del paquete la dirección puede ser indicada como un host tipo "instable.org" o como una dirección IP en formato numérico. La máscara se puede indicar de la manera tradicional (p.ej 192.168.0.1/255.255.255.0) o empleando la notación moderna (p. ej. 192.168.0.1/24) el 24 representa los 24 primeros bits de la máscara cerrados si se pusiera 32 sería una máscara totalmente cerrada, con 16 se obtiene los 16 primeros cerrados, etc. si se le agrega un ! se niega la opción

- [!] --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.

- -t «tabla»

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

- --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.

- -V

Muestra version de iptables

- -v

Muestran todos los detalles de las reglas, tales como los contadores de paquetes y bytes, las comparaciones TOS, y las interfaces. En cualquier otro caso, se omitirán estos valores.

- -X cadena

Elimina la cadena especificada creada por el usuario o todas las cadenas creadas por el usuario si no se especifica.

- -Z cadena

Pone a cero todos los contadores de la cadena o de todas si no se especifica nada.

3.3.2 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 (aceptar, descartar, encolar o retornar, respectivamente). 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 (tal como ocurre con una llamada a una subrutina en un lenguaje de programación). Si el paquete consigue atravesar la cadena definida por el usuario sin que ninguna de las reglas de esa cadena actúe sobre él, el procesamiento del paquete continúa donde había quedado en la cadena actual. Éstos llamados entre cadenas se pueden anidar hasta cualquier nivel deseado.

Existen los siguientes destinos ya incorporados:

- **ACCEPT** (aceptar)

Éste destino hace que netfilter acepte el paquete. El significado de esto depende de cuál sea la cadena realizando esta aceptación. De hecho, 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)

Éste destino hace que netfilter descarte el paquete sin ningún otro tipo de procesamiento. Él paquete simplemente desaparece sin indicación de que fue descartado al ser entregado a la terminal de envio 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)

Éste 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, éste 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 defecto de la cadena en cuestión (ACCEPT, DROP o similar).Hay muchos destinos de extensión disponibles. Algunos 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 TPC denegadas, un icmp-port-unreachable para sesiones UDP denegadas o un icmp-protocol-unreachable para paquetes no TCP y no 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 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 orígen del paquete sean reescritos para 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**

Esta 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 matchea 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.

3.4 Tablas

Iptables consiste básicamente de tres tablas ya incorporadas, cada una de las cuales contiene ciertas cadenas predefinidas. Es posible crear nuevas tablas mediante módulos de extensión. Él 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.

3.4.1 Tabla filter:

Usado para implementar el firewall. Aquí se produce el filtrado de paquetes (es decir, de bloquear o permitir que un paquete continúe su camino) Esta tabla es la responsable de todos los paquetes que pasan a través de la tabla de filtros. Las reglas INPUT y OUTPUT, se usan para filtrar los paquetes que van dirigidos a nuestra maquina FORWARD se usa para filtrar paquetes que van a otras redes o maquinas.

Contiene las siguientes cadenas predefinidas y cualquier paquete pasará por una de ellas:

Filter table (Tabla de filtros)

- INPUT

Todo el tráfico entrante: todos los paquetes destinados a este sistema atraviesan esta cadena (y por esto se la llama algunas veces LOCAL_INPUT o ENTRADA_LOCAL)

- OUTPUT

Todo el tráfico saliente Todos los paquetes creados por éste sistema atraviesan esta cadena (a la que también se la conoce como LOCAL_OUTPUT o SALIDA_LOCAL)

- FORWARD

Para enrutar tráfico a través de nuestro ordenador hacia otro ordenador. Se supone que éste tráfico no es para nosotros (Cadena de REDIRECCIÓN) Todos los paquetes que meramente pasan por este sistema (para ser ruteados) recorren esta cadena

3.4.2 NAT table (Tabla de traducción de direcciones de red)

Para alterar el estado de un paquete, 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. Ésta tabla es la responsable de configurar las reglas de reescritura de direcciones o de puertos de los paquetes. El primer paquete en cualquier conexión pasa a través de esta tabla; estas son reglas que se aplican ANTES que las filter, se usan para rederigir puertos, direcciones y/ o hacer cambios en las direcciones

Contiene las siguientes cadenas redefinidas:

- **PREROUTING** (Cadena de PRERUTEO)

Alteran el tráfico así como llegue a nosotros, los paquetes entrantes pasan a través de esta cadena antes de que se consulte la tabla de ruteo local, principalmente para DNAT (destination-NAT o traducción de direcciones de red de destino)

- **POSTROUTING**

Alterar paquetes generados localmente antes de enrutarlos Los paquetes salientes pasan por esta cadena después de haberse tomado la decisión del ruteo, principalmente para SNAT (source-NAT o traducción de direcciones de red de origen)

- **OUTPUT**

Altera paquetes justo antes de que salgan, Permite hacer un DNAT limitado en paquetes generados localmente

3.4.3 Tabla MANGLE:

Existen otras reglas que van por encima de las NAT, son reglas para modificar los paquetes. 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 (Cadena de PRERUTEO)**

Todos los paquetes que logran entrar a este sistema, antes de que el ruteo decida si el paquete debe ser reenviado (cadena de REENVÍO) o si tiene destino local (cadena de ENTRADA)

- INPUT chain (Cadena de ENTRADA) — Todos los paquetes destinados para este sistema pasan a través de esta cadena

- FORWARD chain (Cadena de REDIRECCIÓN) — Todos los paquetes que meramente pasan por este sistema pasan a través de esta cadena

- OUTPUT chain (Cadena de SALIDA) — Todos los paquetes creados en este sistema pasan a través de esta cadena

- POSTROUTING chain (Cadena de POSRUTEO) — Todos los paquetes que abandonan este sistema pasan a través de esta cadena

Además de las cadenas ya incorporadas, el usuario puede crear todas las cadenas definidas por el usuario que quiera dentro de cada tabla, las cuales permiten agrupar las reglas en forma lógica.

Cada cadena contiene una lista de reglas. Cuando un paquete se envía a una cadena, se lo compara, en orden, contra cada regla en la cadena. La regla especifica qué propiedades debe tener el paquete para que la regla lo matchee, como número de puerto o dirección IP. Si la regla no lo matchea, el procesamiento continúa con la regla siguiente. Si la regla, por el contrario, matchea el paquete, las instrucciones de destino de las reglas se siguen (y cualquier otro procesamiento de la cadena normalmente se

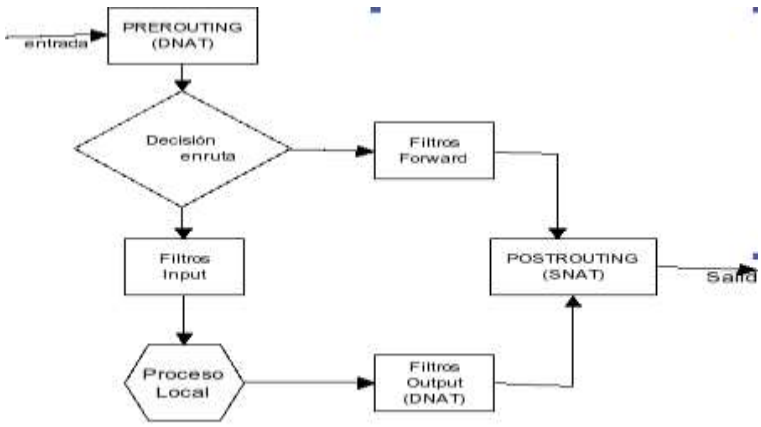
aborta). Algunas propiedades de los paquetes solo pueden examinarse en ciertas cadenas (por ejemplo, la interfaz de red de salida no es válida en la cadena de ENTRADA).

Algunos destinos solo pueden usarse en ciertas cadenas y/o en ciertas tablas (por ejemplo, el destino SNAT solo puede usarse en la cadena de POSRUTEO de la tabla de traducción de direcciones de red).

3.5 Esquemas de filtrado

Es importante saber cual es el circuito exacto que siguen los paquetes en la máquina. Cada paquete que llega a nuestro cortafuegos sigue un determinado camino donde se le aplican una determinadas reglas de filtrado. Si el paquete no ha verificado ninguna de las reglas de aceptar o denegar entonces también deberemos tomar una decisión final con

ese paquete.



3.6 Ejemplos de comandos

Cerrar conexiones entrantes desde eth0 y hacia el puerto (local) 80 (HTTP)

iptables -A INPUT -p tcp -i eth0 --dport 80 -j DROP

Enmascarar por las conexiones procedentes de la red 10.0.0.0 como si lo hicieran desde la ip configurada en la interfaz eth0

iptables -t nat -A POSTROUTING -o eth0 -s 10.0.0.0/255.255.255.0 -j MASQUERADE

Redireccionar al puerto 3128 (proxy) todos los paquetes que entran por eth1 y con destino puerto 80 (HTTP), de esta manera conseguimos un proxy transparente.

iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port 3128

Detener todas las conexiones entrantes desde la interfaz eth0 menos la conexiones al servicio ssh. La primera regla deja pasar los paquetes al 22 y la segunda cierra todo lo demas.

#iptables -A INPUT -p tcp -i eth0 -m state --state NEW,ESTABLISHED,RELATED --dport 22 -j ACCEPT

 # Iptables -A INPUT -p all -i eth0 -m state --state NEW,INVALID -j DROP

Deshabilitar los paquetes ICMP entrantes de tipo echo (8) para el firewall (regla INPUT) y la red protegida (regla FORWARD).

 # iptables -A INPUT -i eth0 -p icmp --icmp-type 8 -j DROP

 # iptables -A FORWARD -i eth0 -p icmp --icmp-type 8 -j DROP

Especificar Puerto

 # iptables -A INPUT -p tcp --dport 1:1024 -j DROP

#iptables -A INPUT -p tcp -s 127.0.0.1 --dport 80 -j ACCEPT

#iptables -A INPUT -p udp -s 0/0 --sport 137 -j DROP

Especificar fragmentos

iptables -A OUTPUT -f -d 192.168.1.1 -j DROP

Especificar direcciones Ip de orígen y destino:

iptables -A INPUT -s 1.2.3.4/32 -d 5.6.7.8/32 -j DROP

Especificar Interface:

 # iptables -A INPUT -i eth0 -p tcp -s 5.6.7.8/32 -d 1.2.3.4/32 -j DROP

 # iptables -A OUTPUT -o ppp0 -p tcp -s 5.6.7.8/32 -d 1.2.3.4/32 -j DROP

Filtrado de puertos

iptables -I INPUT -p tcp --dport 22 -j DROP

Especificar una Inversión:

 # iptables -A INPUT -s 5.6.7.8/32 -d 1.2.3.4/32 -j DROP

Evitar ataques por denegación de servicio (DoS) con una tasa más rápida para incrementar la respuesta.

Protección contra Syn-flood (inundación mediante Syn):

 # iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT

Furtivo buscando puertos (port scanner):

 # iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT

Evitar Ping de la muerte:

 # iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j DROP

Aceptar conexiones al puerto 80 (www) en la tarjeta eth0

iptables -A INPUT -i eth0 -s 0.0.0.0/0 -p TCP --dport www -j ACCEPT

Compartir mi conexión a internet:

#iptables -t nat -I POSTROUTING -s 192.168.0.0/24 -o ppp0 -j MASQUERADE

La regla anterior dice: todo paquete cuyo origen (source, de ahi viene el "-s" de la linea de comandos) sea la red 192.168.0.0/24 y que salga por la interface de red ppp0 sea afectado por la operación (target) MASQUERADE. Esta operación hace que se realice

NAT con la dirección IP pública que tenemos en ese momento en ppp0 (se lo utiliza para conexiones con direccion IP dinámica).

#iptables -t filter -I FORWARD -s 192.168.0.10 -j DROP

Explicación: Bloqueo todos los paquetes (DROP) que provengan de la máquina 192.168.0.10, los cuales vayan a ser FORWARDdeados (o sea, cuyo destino no sea el linux).

#iptables -t filter -I INPUT -s 192.168.0.11 -j DROP

Explicación: Bloqueo todos los paquetes (DROP) que provengan de la máquina 192.168.0.11, los cuales tengan como destino el linux (por eso el INPUT)

Creación de una cadena que bloquee las conexiones nuevas, excepto si vienen de dentro.

 # iptables -N micadena

 # iptables -A micadena -m state --state ESTABLISHED, RELATED -j ACCEPT

 # iptables -A micadena -m state --state NEW -i ! ppp0 -j ACCEPT

 # iptables -A micadena -j DROP

Saltar a esa cadena desde las cadenas INPUT y FORWARD.

 # iptables -A INPUT -j micadena

 # iptables -A FORWARD -j micadena

Nota: Todo aquello que le anteceda el signo "!" significa NO o NOT.

 # iptables -A FORWARD -s 1.2.3.4/32 -d !10.0.0.0/8 -j micadena

###----- definición de politicas por defecto-------###

iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -p FORWARD ACCEPT

#Aplicando reglas en INPUT

iptables -A INPUT -p tcp --dport 1:1024 -j DROP

iptables -A INPUT -p udp --dport 1:1024 -j DROP

iptables -A INPUT -p tcp -s 10.0.0.1 -j ACCEPT

iptables -A INPUT -p udp -s 10.0.0.1 -j ACCEPT

iptables -A INPUT -p all -s 0.0.0.0/0 -j ACCEPT

iptables -A INPUT -p all -m state --state NEW,INVALID -j DROP

#Aplicando reglas en OUTPUT

iptables -A OUTPUT -p all -s 10.0.0.1 -j ACCEPT

iptables -A OUTPUT -p all -s 0.0.0.0/0 -j ACCEPT

#Aplicando reglas en FORWARD

iptables -A FORWARD -p all -m state --state NEW,INVALID -j DROP

Este ejemplo, solo confiamos en una máquina local (El router), todo lo demás está cerrado, y además se rechaza las conexiones no solicitadas,

Mezclar nat y filtrado de paquetes

Enmascarar ppp0

iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

Prohibir paquetes NEW e INVALID que venga de o sean redirigidos

por ppp0

iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP

iptables -A FORWARD -i ppp0 -m state --state NEW,INVALID -j DROP

Activar el reenvío IP (IP Forwarding)

echo 1 > /proc/sys/net/ipv4/ip_forward

 # iptables -t nat -A POSTROUTING -s 1.2.3.4/32 -j SNAT --to 200.3.4.5

 # iptables -t nat -A PREROUTING -p tcp -d 200.3.4.5 --dport 80 -j DNAT --to 1.2.3.4:80

SCRIPT DE EJEMPLO

iptables –F

iptables –X

iptables –Z

iptables -t nat –F

###----- **Aquí se va a definir la politica por defecto**-------###

iptables -P INPUT ACCEPT

iptables -P OUTPUT ACCEPT

iptables -P FORWARD ACCEPT

iptables -t nat -P PREROUTING ACCEPT

iptables -t nat -P POSTROUTING ACCEPT

------Comienzo de las reglas---------###

###-------- A localhost le dejamos todo --------###

/sbin/iptables -A INPUT -i lo -j ACCEPT

###------Con esto evitamos responder a un ping-----###

echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all

###------Con esto evitamos un tipo de ataque por filtrado de bytes---###

iptables -A INPUT -i ppp0 -f -m length --length 0:40 -j DROP

###------Activamos el servidor solo para localhost----#

iptables -A INPUT -p tcp -s ! 127.0.0.1 --dport 80 -j DROP

###-----Ahora cerramos todo tanto por tcp como udp, desde el 1 al 1024---###

iptables -A INPUT -p tcp --dport 1:1024

iptables -A INPUT -p udp --dport 1:1024

Con esto terminado, es un script muy simple pero práctico e ilustrativo de que con muy pocas líneas se puede construir un firewall basado en iptables, además se puede ver el principio de flexibilidad que tiene iptables, esto sería el típico firewall de ejemplo de uso de las funciones más básicas.

Por otro lado tenemos la opción de cerrar todo menos lo necesario, este es el axioma en los administradores.

Borrar reglas

Se puede hacer de dos maneras. Primero, como sabemos que es la única regla en la cadena, podemos usar un borrado por número:

 # iptables -D INPUT 1

Para borrar la regla número uno de la cadena INPUT.

La segunda manera es repetir la orden -A, pero cambiando -A por -D. Es útil cuando se tiene una compleja cadena de reglas y no se quiere estar contándolas para averiguar que es la regla 37 la que se quiere eliminar. En este caso, se usaría:

 # iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP

Proteger máquina con Internet

Se va a proteger con su propio firewall. Se crea un script de shell en el que se van aplicando las reglas.

echo -n **Aplicando Reglas de Firewall**...

FLUSH de reglas

iptables –F

iptables –X

iptables –Z

iptables -t nat –F

Establecemos politica por defecto

i*ptables -P INPUT ACCEPT*

iptables -P OUTPUT ACCEPT

iptables -P FORWARD ACCEPT

iptables -t nat -P PREROUTING ACCEPT

iptables -t nat -P POSTROUTING ACCEPT

Empezamos a filtrar

 # El localhost se deja (en conexiones locales)

iptables -A INPUT -i lo -j ACCEPT

A la IP de esa maquina le dejamos todo

iptables -A INPUT -s 192.168.1.1 -j ACCEPT

 # A un nodo se le deja entrar al servidor de internet

iptables -A INPUT -s 192.168.1.3 -p tcp --dport 80 -j ACCEPT

 # A otro nodo se le deja usar el ssh

iptables -A INPUT -s 192.168.1.5 -p tcp --dport 22 -j ACCEPT

 # El puerto 443 de www debe estar abierto, es un servidor web seguo.

iptables -A INPUT -p tcp --dport 443 -j ACCEPT

 # El puerto 80 de www debe estar cerado, es un servidor web seguo.

iptables -A INPUT -p tcp --dport 80 -j DROP

 # Y el resto, se serrara

iptables -A INPUT -p tcp --dport 20:21 -j DROP

iptables -A INPUT -p tcp --dport 3306 -j DROP

iptables -A INPUT -p tcp --dport 22 -j DROP

iptables -A INPUT -p tcp --dport 10000 -j DROP

echo "

OK . Verifique que lo que se aplica con: iptables -L -n"

Ejemplos

Se quiere denegar todos los paquetes que vengan de internet y permitir que los equipos de la LAN (192.168.1.0/24) salgan a internet se utilizará el siguiente conjunto de reglas:

 # iptables -t filter -A INPUT -i lo -j ACCEPT

 # iptables -t filter -A INPUT -m state --state

RELATED,ESTABLISHED -j ACCEPT

 # iptables -t filter -A INPUT -s 192.168.1.0/24

-m state --state NEW -j ACCEPT

 # iptables -t filter -P INPUT DROP

 # iptables -t filter -A FORWARD -m state --state

RELATED,ESTABLISHED -j ACCEPT

 # iptables -t filter -A FORWARD -s 192.168.1.0/24

-m state --state NEW -j ACCEPT

 # iptables -t filter -P FORWARD DROP

 # iptables -t nat -A POSTROUTING -s 192.168.1.0/24

-d \! 192.168.1.0/24 -j MASQUERADE

Este conjunto de reglas básicamente deniega toda entrada externa y permite que los paquetes provenientes de la red local pasen a través del firewall e incluso esten dirigidos al firewall mismo. Paso a describir en detalle qué hace cada regla:

Esta regla acepta todos los paquetes que entren por la interfase de red 'lo' (la interfase de red utilizada para comunicación interna del mismo equipo: localhost). Esta ubicada en la cadena de INPUT ya que son paquetes que únicamente pueden estar dirigidos al firewall mismo.

Esta regla permite que todos los paquetes que estén en estado ``relacionado" o ``establecido" y que tengan como destino final el mismo firewall se acepten. El estado del paquete es determinado por el módulo ``state", cuando se utiliza este módulo el kernel va manteniendo una tabla de conexiones que se han establecido en algún sentido (por ejemplo un ftp ejecutado en el firewall para bajar algo de internet). Este módulo es muy útil para identificar los paquetes ``de vuelta" (o sea paquetes que vienen como respuesta a otros originalmente enviados por nosotros).

Esta regla permite que todos los paquetes de inicialización de conexión provenientes de la red 192.168.1.0/24 sean aceptados. El hecho de utilizar de nuevo el módulo de estado permite que el kernel sólo acepte paquetes que considere como ``nuevos", hay paquetes que se catalogan como inválidos, éstos últimos no van a ser aceptados por más que vengan de la red interna.

Ésta regla pone la política de la cadena en DROP. La política de la cadena es la última en ejecutarse (luego de que se evaluaron todas las reglas de la cadena), si un paquete llega a esta instancia entonces va a ser descartado.

Esta regla hace lo mismo que la regla ``2" pero con la diferencia que deja pasar los paquetes en estado ``relacionado" o ``establecido" que tienen como destino final una red detrás del firewall (en este caso seria la red 192.168.1.0/24).

Al igual que la regla ``3", esta regla acepta los paquetes que son enviados desde la red local hacia otra red (todos los paquetes que no van al firewall). Básicamente todos los paquetes que salen de las estaciones de trabajo de la red interna y que van a internet.

Esta regla pone como política de la regla que descarte todos los paquetes que lleguen a esta instancia.

Esta regla se ubica en la tabla de ``nat" ya que se utiliza para cambiar el encabezado de todos los paquetes que provengan de la red 192.168.1.0/24 y vayan a una red que no sea

192.168.1.0/24 (o sea internet en éste ejemplo). Se utiliza MASQUERADE para que el kernel ponga en el campo ``origen del paquete" la dirección pública del mismo firewall (la que tiene en ese momento, es indispensable en firewalls que tienen una IP pública y dinámica).

Algunos detalles interesantes del ejemplo anterior. La regla número uno de la cadena de FORWARD es la que más se ejecuta ya que por lo general hay muchos más paquetes considerados como ``relacionados" o ``establecidos" a los que se consideran ``nuevos" (estos últimos son sólo aquellos que se envían al inicio de una conexión). Por lo tanto se logra mejor performance si se coloca esta regla al inicio de la cadena, evitando que el kernel verifique las subsiguientes reglas de la cadena. La utilización del estado NEW es útil para evitar que paquetes en estado INVALID sean aceptados.

Permitiendo el acceso a ciertos puertos

Se tiene un equipo que tiene una dirección fija de internet y se quiere que este mismo equipo dé servicios (HTTP, HTTPS y SSH). Esto por supuesto no es un esquema recomendado, porque siempre se recomienda separar el firewall del resto de los equipos que brindan algún tipo de servicio, pero dado que hay tan solo un equipo para dar todos los servicios no nos queda otra alternativa.
Los protocolos HTTP, HTTPS y SSH trabajan con TCP únicamente y están ligados a los puertos 80, 443 y 22 respectivamente.

Siguiendo con el ejemplo anterior, este equipo también protege una red LAN (10.1.1.0/24) y es utilizado como puerta de enlace a internet.

 # iptables -t filter -A INPUT -i lo -j ACCEPT

 # iptables -t filter -A INPUT -m state --state

RELATED,ESTABLISHED -j ACCEPT

 # iptables -t filter -A INPUT -d 200.32.106.148 -p tcp

--dport 22 -m state --state NEW -j ACCEPT

 # iptables -t filter -A INPUT -d 200.32.106.148 -p tcp

--dport 80 -m state --state NEW -j ACCEPT

 # iptables -t filter -A INPUT -d 200.32.106.148 -p tcp

--dport 443 -m state --state NEW -j ACCEPT

 # iptables -t filter -A INPUT -s 10.1.1.0/24 -m state

--state NEW -j ACCEPT

 # iptables -t filter -P INPUT DROP

 # iptables -t filter -A FORWARD -m state --state

RELATED,ESTABLISHED -j ACCEPT

 # iptables -t filter -A FORWARD -s 10.1.1.0/24 -m state

--state NEW -j ACCEPT

 # iptables -t filter -P FORWARD DROP

 # iptables -t nat -A POSTROUTING -s 10.1.1.0/24

-d \! 10.1.1.0/24 -j SNAT --to-source 200.32.106.148

Las reglas nuevas son las que abren los puertos 22, 80 y 443 (reglas 3, 4 y 5), para realizar esto se utilizaron las opciones '-p', que permite especificar un protocolo (generalmente se utiliza TCP, UDP o ICMP, aunque se aceptan todos los protocolos que están definidos en /etc/protocols), también se utilizó la opción '-dport «nro. de puerto»' (puerto destino del paquete) que esta directamente relacionada con el argumento de la opción '-p'. Habiendo puesto como argumento el protocolo 'tcp', utilizar la opción '-dport' tiene sentido ya que el protocolo 'tcp' es un protocolo que maneja puertos. Si se hubiese utilizado un protocolo como 'icmp' o 'esp', el comando hubiera dado un error. La otra opción nueva que se utilizó fue '-d «dirección»' que indica la dirección destino a la cual esta destinado el paquete.

Por otro lado la regla nro. ``11'' introduce un cambio a la utilizada en el ejemplo anterior, en vez de utilizar el MASQUERADE se hace un NAT de la dirección origen del paquete en la cadena POSTROUTING de la tabla de 'nat'. Esta es la forma de fijar la dirección de origen a todos los paquetes que salgan a internet, es básicamente lo que se hizo con MASQUERADE, con la diferencia que acá se especifica una dirección y no se toma la dirección pública que tiene el equipo. Este parámetro es sumamente útil cuando tenemos más de una dirección de IP válida para utilizar.

3.7 Haciendo una DMZ

Se denomina DMZ host al ordenador que, situado en la DMZ, está expuesto a los riesgos de acceso desde Internet. Es por ello un ordenador de sacrificio, pues en caso de ataque está más expuesto a riesgos.

Normalmente el DMZ host está separado de Internet a través de un router o mejor un cortafuegos. Es aconsejable que en el cortafuegos se abran al exterior únicamente los puertos de los servicios que se pretende ofrecer con el DMZ host.

Como tercer y último esquema de firewall vamos a armar uno que tenga tres placas de red (en vez de las clásicas dos placas). Una de las placas va a ser la placa a internet, la segunda va a estar conectada a un hub separado donde vamos a poner todos los equipos semi publicos y por último la tercer placa de red va a estar conectada a nuestra LAN.

Aún teniendo un único número de IP público se puede armar una DMZ, lo que se hace es redireccionar los paquetes a diferentes equipos internos dependiendo el puerto destino de los mismos. Por ejemplo, si quiero tener un servidor de correo, otro de web y uno de acceso por ssh, voy a reenviar todos los paquetes que vengan al puerto 25 (smtp) a un equipo, todos los que vayan al 80 (web) a mi servidor web y por último todos los paquetes que vengan al puerto 22 van a ir a parar a mi servidor de acceso.

Estas serian las reglas a implementar (nota: el servidor sigue siendo la unica puerta de enlace de la red a internet):

 # iptables -t filter -A INPUT -i lo -j ACCEPT

 # iptables -t filter -A INPUT -m state --state

RELATED,ESTABLISHED -j ACCEPT

 # iptables -t filter -A INPUT -s 192.168.1.0/24

-m state --state NEW -j ACCEPT

 # iptables -t filter -P INPUT DROP

 # iptables -t filter -A FORWARD -m state

--state RELATED,ESTABLISHED -j ACCEPT

 # iptables -t filter -A FORWARD -m multiport

-p tcp --dport 22,25,80 -d 192.168.22.0/28 -m state

--state NEW -j ACCEPT

 # iptables -t filter -A FORWARD -s 192.168.22.0/24

-d 192.168.1.0/24 -j REJECT

 # iptables -t filter -A FORWARD -s 192.168.1.0/24

-m state --state NEW -j ACCEPT

 # iptables -t filter -P FORWARD DROP

 # iptables -t nat -A PREROUTING -d 200.32.106.148

-p tcp --dport 22 -j DNAT --to-destination 192.168.22.1:22

 # iptables -t nat -A PREROUTING -d 200.32.106.148

-p tcp --dport 25 -j DNAT --to-destination 192.168.22.2:25

 # iptables -t nat -A PREROUTING -d 200.32.106.148

-p tcp --dport 80 -j DNAT --to-destination 192.168.22.3:80

 # iptables -t nat -A POSTROUTING -s 192.168.1.0/24

-d \! 192.168.0.0/16 -j SNAT --to-source 200.32.106.148

Los cambios interesantes son:

La regla ``6" introduce el concepto de la utilización de varios módulos en una sola regla. Con iptables podemos definir varias veces el flag '-m', de hecho en esta regla se utiliza el módulo 'multiport' para poder definir en una única regla que se verifiquen todos los paquetes que tienen como puerto destino el puerto 22, el 25 o el 80. Además, utilizando el módulo de estado, que el paquete este declarado como un paquete ``nuevo". Por último se utilice otro parámetro nuevo para mostrar su uso, en la dirección destino del paquete puse el prefix '/28', este prefix establece que el rango de IPs destino es desde la IP 192.168.22.0 hasta la 192.168.22.15, con lo cual todos los equipos que ponga en la DMZ deberán estar en este rango (o agrandar el prefix por supuesto). Con esto quiero demostrar que el '/XX' que se pone detrás de los números de IP de redes indican justamente el rango de IPs que maneja esa red, no indica la máscara de la red (la red en la DMZ podría ser clase 'C' tranquilamente y eso es '/24').

La regla ``7" es una regla de esas que uno pone ``por las dudas", si bien no hay ninguna regla que habilite el paso entre la red (DMZ) 192.168.22.0/24 y la red (LAN)

192.168.1.0/24, uno pone esta regla que rechaza todo tipo de conexión que provenga desde la red DMZ hacia la red LAN (dicen que lo que abunda no daña ...).

El esquema inverso (desde la LAN hacia la DMZ) sí esta permitido en la regla ``8", y todos los paquetes ``de vuelta" también van a estar permitidos por la regla ``5".

Las reglas ``10", ``11" y ``12" trabajan en PREROUTING ya que es el único lugar donde se puede hacer un NAT de la dirección destino del paquete. Aqui se especifica que si un paquete va destinado a la dirección 200.32.106.148, puerto 22, éste sea reescrito para que vaya dirigido al IP 192.168.22.1, puerto 22 (podríamos hasta cambiar el puerto destino si quisiéramos). Lo mismo se hace con los puertos 25 y 80 en las reglas ``11" y ``12". Por último la regla ``13" tiene un pequeño cambio al segundo ejemplo, el prefix de la red destino es más grande y abarca una clase 'B' de la red 192.168.X.X.

Conclusiones

- Como conclusión para el presente trabajo se tiene que la elaboración del manual didáctico Web será de apoyo para la configuración de firewall, Proxy y esquemas en la red, para los estudiantes de la carrera Técnico en Ingeniería de Redes Computacionales de la Universidad Tecnológica de El Salvador en las materias de Sistemas Operativos.

  Así como conocer las ventajas y desventajas del Sistema Operativo Linux como plataforma para los servicios de Proxy y Firewall.

- Con este manual se han determinado los parámetros de configuración de los servidores firewall y Proxy.

Recomendaciones

- Para la mayor comprensión de este manual de configuración se requiere que los estudiantes de la carrera Técnico en Ingeniería en Redes Computacionales tenga un conocimiento previo del Sistema Operativo Linux para mayor comprensión del tema.

- Al igual se requiere que el estudiante tenga conocimiento de normativas de seguridad para el sistema operativo Linux.

- Leer, comprender y practicar los distintos comandos utilizados en dicho sistema operativo y en el cual se da una previa explicación de ellos en este manual.

Bibliografía

Wikipidia. [en línea]. «iptables »20 marzo 2007.

<http://es.wikipedia.org/wiki/Netfilter/iptables> [consulta: 30 marzo 2007]

Netfilter. [en línea]. « packet filtering» 15 enero 2007.

<http://www.netfilter.org/documentation/HOWTO/es/packet-filtering-HOWTO-7.html>

[consulta: 30 marzo 2007]

Kaipy.rastafuri. [en línea]. « squid» 20 diciembre 2006.

<http://kaipy.rastafurbi.org/redes/squid.pdf> [consulta: 30 marzo 2007]

Wikipidia. [en línea] «squid» 20 febrero 2007. <http://es.wikipedia.org/wiki/Squid>

[consulta: 30 marzo 2007]

Squid-cache. [en línea]. «squid» 15 marzo 2007. < http://www.squid-cache.org >

[consulta: 30 marzo 2007]

Visolve. [en línea]. «*Squid - A Quick Start Guide*» 10 febrero 2007

<http://www.visolve.com/squid/sqguide.php> [consulta: 30 marzo 2007]

ANEXOS

I Oferta técnica

San Salvador, 22 de mayo de 2007

Director de Escuela de Informática

Presente

Por medio de la presente, hacemos de su conocimiento, que estamos brindando esta oferta técnica donde se detallará el tipo de software que se utilizará en el laboratorio de redes del edificio Francisco Morazán de la Universidad Tecnológica de el Salvador, para llevar a cabo las prácticas para las configuraciones del servidor Proxy con el programa de Squid 3.05 y firewall con Iptables, bajo el sistema operativo Linux, para el aprendizaje de los estudiantes de la carrera técnica de ingeniería en redes computacionales

Atentamente,

Jorge Alberto Reyes Luís Fernando Aguilar Alba Ivette Aragón

II Oferta económica

San Salvador, 22 de mayo de 2007

Director de Escuela de Informática

Presente

Por medio de la presente, hacemos de su conocimiento que estamos brindando esta oferta económica donde se detallarán los programas a utilizar para el manual, que se utilizará en el laboratorio de redes del edificio Francisco Morazán, de la Universidad Tecnológica de el Salvador, para llevar a cabo las prácticas para las configuraciones del servidor Proxy con el software Squid 3.05 y firewall con Iptables bajo el sistema operativo Linux para el aprendizaje de los estudiantes de la carrera técnica de ingeniería en redes computacionales

Atentamente,

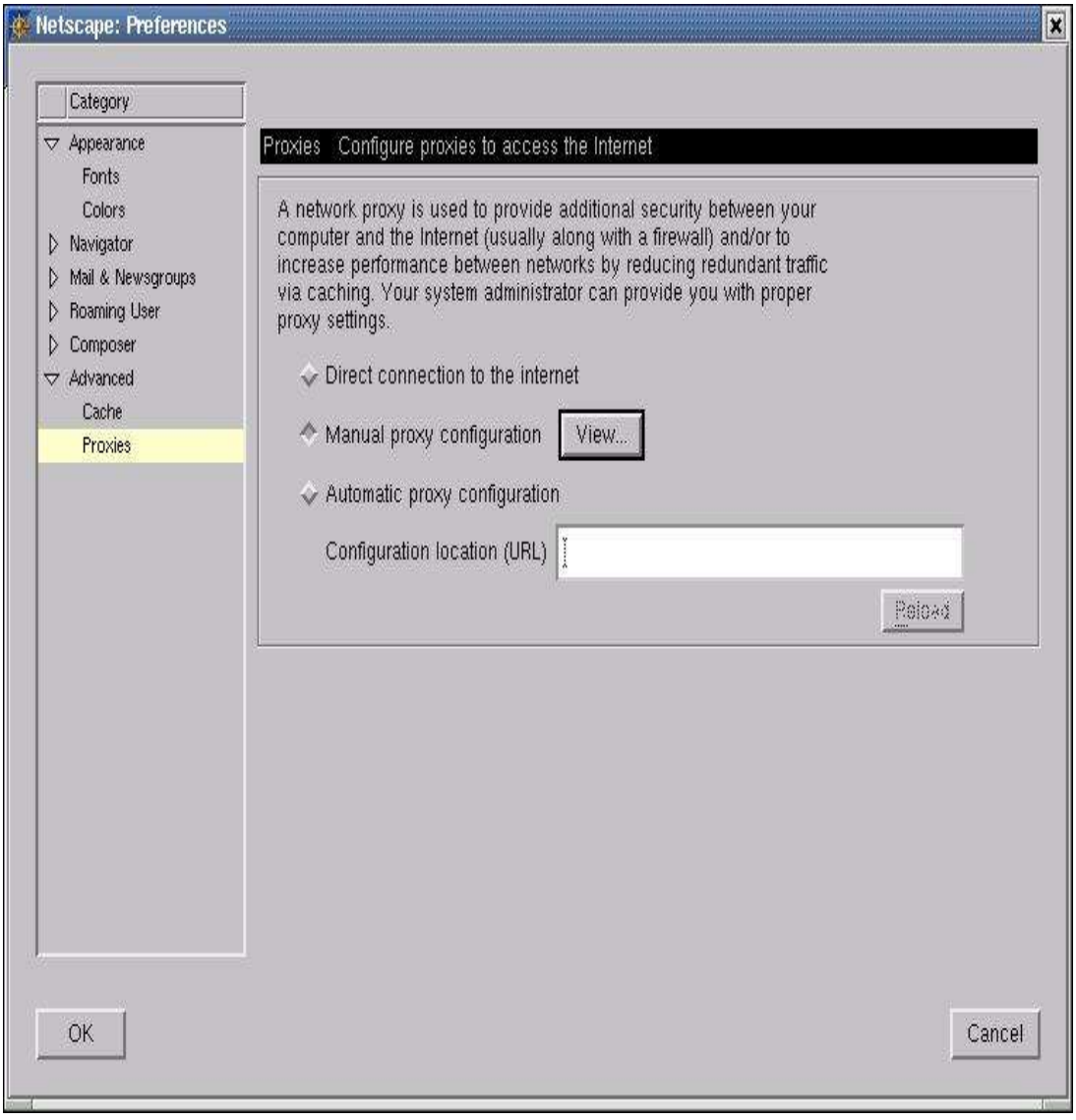
Jorge Alberto Reyes Luís Fernando Aguilar Alba Ivette Aragón

III Configuración de los clientes

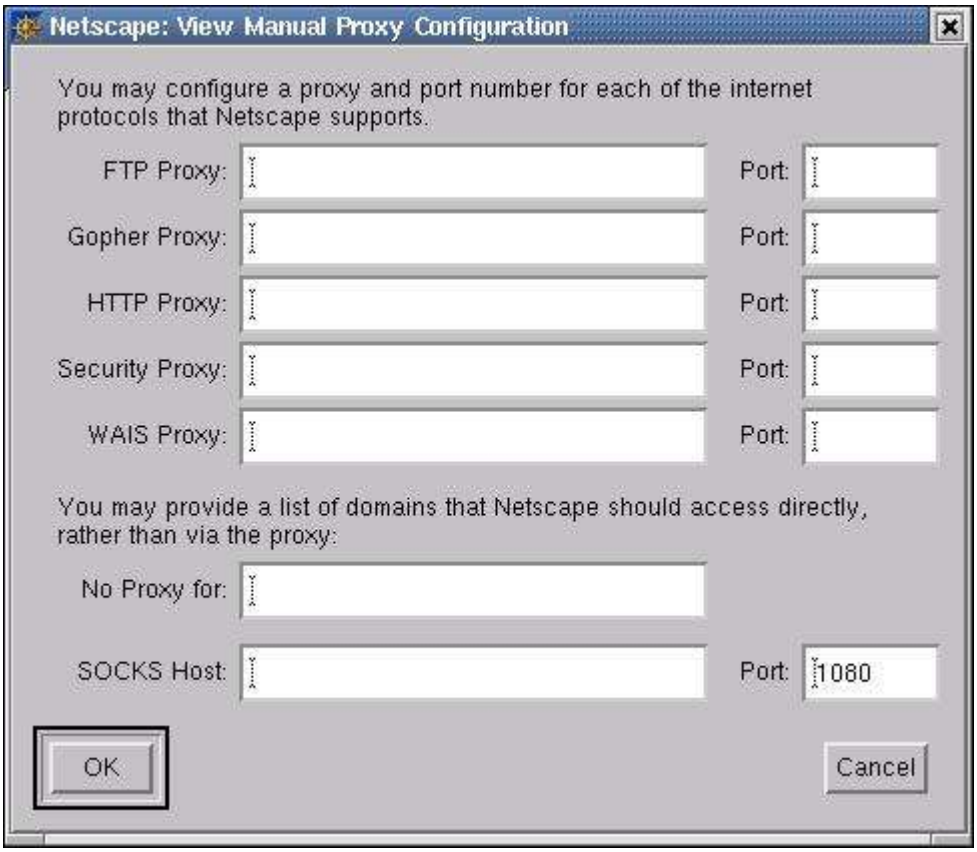
La configuración de los clientes consiste en indicar al navegador que utilice un Proxy para conectarse a Internet. Veamos un ejemplo con un navegador popular:

Netscape Navigator:

Seleccione Preferences del menú Edit. En la página de proxies de la sección Advanced, seleccione Manual proxy configuration, y haga click sobre el botón View.



Para cada protocolo que squid soporta (por defecto HTTP, FTP y gopher), teclee la IP o el nombre del servidor que aloja squid, junto al puerto correspondiente (por defecto el 3128).



De manera similar se deben configurar otras aplicaciones clientes que se desea que utilizaren la cache.

IV Comandos básicos Linux

Los comandos son esencialmente los mismos que cualquier sistema UNIX. En las tablas 3.1 y 3.2 se tiene la lista de comandos mas frecuentes. En la tabla 3.3 se tiene una lista de equivalencias entre comandos Unix/Linux y comandos DOS.

Comando/Sintaxis	Descripción	Ejemplos
cat *fich1* [...*fichN*]	Concatena y muestra un archivos	cat /etc/passwd
	Archivos	cat dict1 dict2 dict
cd [*dir*]	Cambia de directorio	cd /tmp
chmod *permisos fich*	Cambia los permisos de un archivo	chmod +x miscript
chown *usuario:grupo fich*	Cambia el dueño un archivo	chown nobody miscript
cp *fich1...fichN dir*	Copia archivos	cp foo foo.backup
diff [*-e*]*arch1 arch2*	Encuentra diferencia entre archivos	diff foo.c newfoo.c
du [*-sabr*] *fich*	Reporta el tamaño del directorio	du -s /home/
file *arch*	Muestra el tipo de un archivo	file arc_desconocido

find *dir test acción*	Encuentra archivos.	find . -name ``.bak" -print
grep [-*cilnv*] *expr archivos*	Busca patrones en archivos	grep mike /etc/passwd
head -*count fich*	Muestra el inicio de un archivo	head prog1.c
mkdir *dir*	Crea un directorio.	mkdir temp
mv *fich1 ...fichN dir*	Mueve un archivo(s) a un directorio	mv a.out prog1
mv *fich1 fich2*	Renombra un archivo.	mv .c prog_dir
less / more *fich(s)*	Visualiza página a página un archivo.	more muy_largo.c
	less acepta comandos vi.	less muy_largo.c
ln [-*s*] *fich acceso*	Crea un acceso directo a un archivo	ln -s /users/mike/.profile .
Ls	Lista el contenido del directorio	ls -l /usr/bin
Pwd	Muestra la ruta del directorio actual	Pwd
rm *fich*	Borra un fichero.	rm foo.c

rm -r *dir*	Borra un todo un directorio	rm -rf prog_dir
rmdir *dir*	Borra un directorio vacío	rmdir prog_dir
tail *-count fich*	Muestra el final de un archivo	tail prog1.c
vi *fich*	Edita un archivo.	vi .profile

Comandos Linux/Unix de manipulación de archivos y directorios tbl_comm_archivos

Comando/Sintaxis	Descripción	Ejemplos
at [*-lr*] *hora* [*fecha*]	Ejecuta un comando mas tarde	at 6pm Friday miscript
cal [[*mes*] *año*]	Muestra un calendario del mes/año	cal 1 2025
date [*mmddhhmm*] [*+form*]	Muestra la hora y la fecha	date
echo *string*	Escribe mensaje en la salida estándar	echo ``Hola mundo''
finger *usuario*	Muestra información general sobre	finger nn@maquina.aca.com.co
	un usuario en la red	
id	Número ID de un usuario	id usuario

kill [-*señal*] *PID*	Matar un proceso	kill 1234
man *comando*	Ayuda del comando especificado	man gcc
		man -k printer
passwd	Cambia la contraseña.	passwd
ps [*axiu*]	Muestra información sobre los procesos	ps -ux
	que se están ejecutando en el sistema	ps -ef
who / rwho	Muestra información de los usuarios	who
	Conectados al sistema.	

Comandos Linux/Unix más frecuentes tbl_comm_frec

Linux	DOS	Significado
cat	type	Ver contenido de un archivo.
cd, chdir	cd, chdir	Cambio el directorio en curso.
chmod	attrib	Cambia los atributos.

clear	cls	Borra la pantalla.
ls	dir	Ver contenido de directorio.
mkdir	md, mkdir	Creación de subdirectorio.
more	more	Muestra un archivo pantalla por pantalla.
mv	move	Mover un archivo o directorio.
rmdir	rd, rmdir	Eliminación de subdirectorio.
rm -r	deltree	Eliminación de subdirectorio y todo su contenido.

Equivalencia de comandos Linux/Unix y DOS tbl_equiv_comandos