PRÁCTICA. PROXY WEB: SQUID

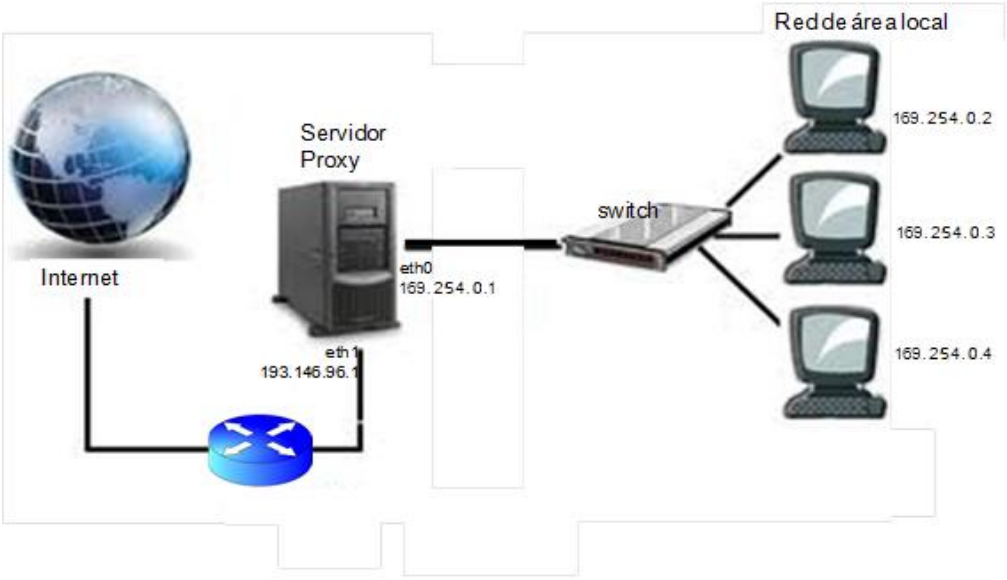
1. Requisitos iniciales 2
2. Introducción 2
3. Configuracción del escenario 3
4. Configuración básica del servidor Squid 6
   4.1. Parámetro http_port ¿Qué puerto utiliza Squid? 7
   4.2. Controles de acceso 7
   4.3. Reglas de Control de Acceso 9
   4.4. Ejemplos de Listas y Reglas de control de acceso 11
      4.4.1. Configuración Servidor 11
      4.4.2. Configuración Clientes 13
5. Acceso por Autenticación 17
6. Ficheros log 20
7. Proxy acelerado 21

1. Requisitos iniciales:

Debemos importar en virtualBox dos máquinas Centos. Una hará de servidor y la otra de cliente.

2. Introducción:

Vamos a configurar un servidor proxy que proporcione/controle el servicio de acceso a la Web desde la red de área local. Como se ve en la figura el servidor proxy tiene que tener dos tarjetas de red[1], una conecta a la red local y otra hacia Internet.



Las peticiones Web de los ordenadores de nuestra red local irán dirigidas al servidor proxy y será el proxy el que haga la petición de la página web al servidor Web destino. El servidor Web destino devolverá la página al proxy y el proxy se la enviará al equipo que le realizó la petición original.

Como se puede comprobar en la figura, desde Internet es como si en nuestra red sólo existiera el servidor Proxy, permaneciendo oculta la existencia del resto de ordenadores, esto es una medida de seguridad para nuestra red.

Además, desde el servidor proxy se puede controlar y restringir el tráfico Web de nuestra organización, lo que aportará una medida de seguridad adicional, ya que los usuarios no accederán a sitios indebidos y en muchas ocasiones peligrosos.

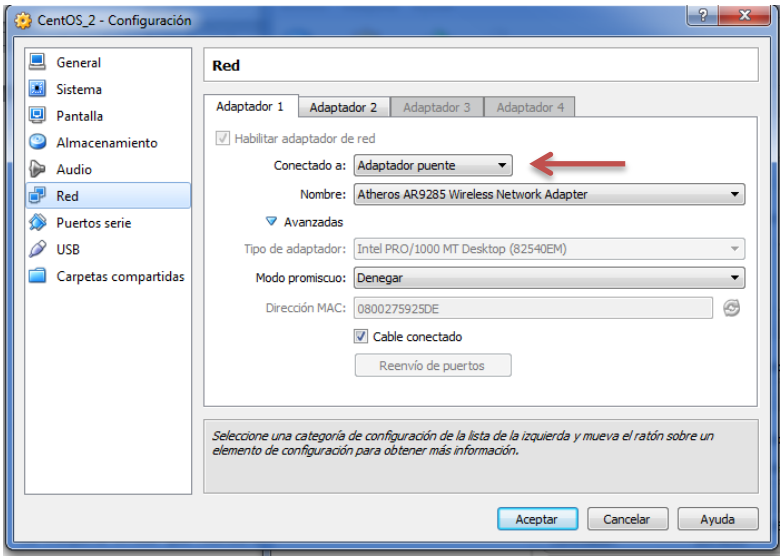
[1] En Linux podemos configurar con una sola tarjeta dos direcciones de red, simulando que tenemos dos tarjetas. Obviamente en un caso real esto no es aconsejable, hay que separar los dos tráficos.

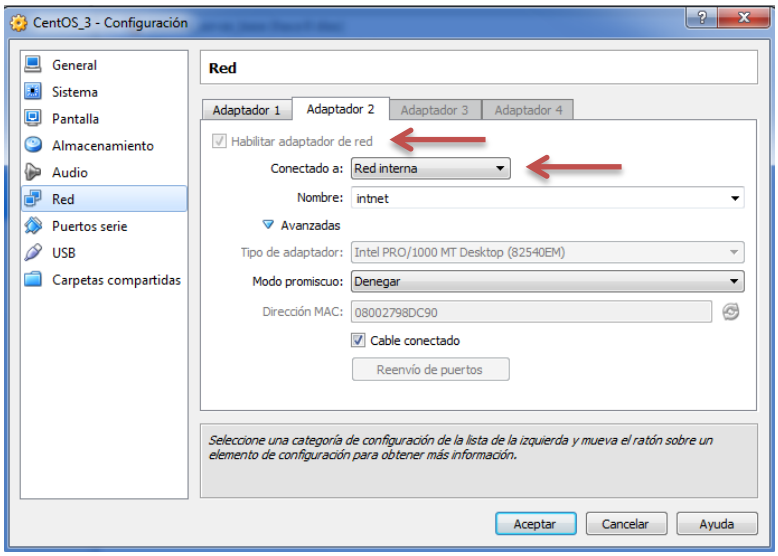
Además, si el servidor hace sólo de proxy para el servicio Web desde la red de área local no se podrá establecer ningún otro tipo de comunicación con el exterior: telnet, smtp (correo electrónico), Messenger, etc.

3. Configuracción del escenario:

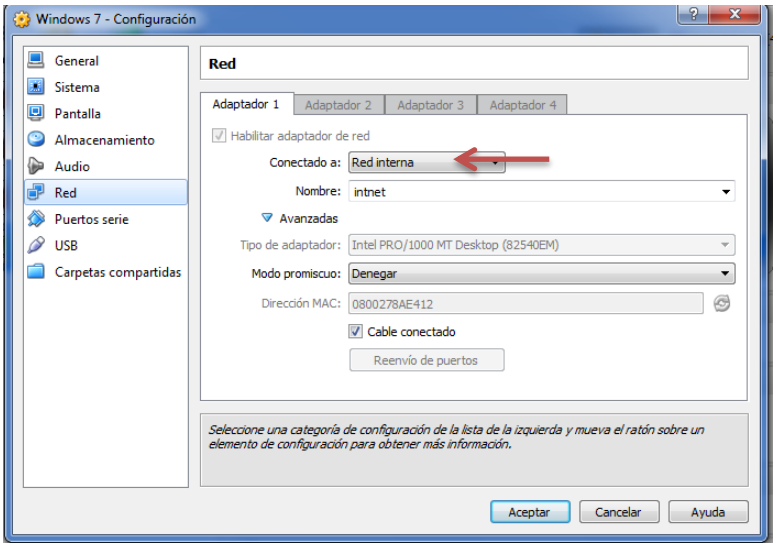
Debemos construir un mapa de red similar al esquema del apartado anterior, vamos a realizarlo creando una red interna dentro de virtual box que no pueda acceder directamente a Internet y donde estarán tanto el servidor como el cliente.

La máquina virtual que realizará la función de servidor proxy, un CentOS, debe tener dos adaptadores de red, uno para acceder a Internet (Adaptador puente) y el otro para la red interna:





La máquina virtual cliente que va a acceder a Internet a través del proxy, debe tener solo un adaptador de red como red interna, lo que le impedirá acceder directamente a Internet:



Como en la red interna no hemos configurado un servidor DHCP que asigne IPs a los "equipos", en las máquinas Windows se asigna una IP aleatoria de 169.254.0.0/16 (169.254.0.1 – 169.254.255.254), y en CentOS no se asignará ninguna IP, por lo que debemos asignarla nosotros manualmente tanto si es el servidor o el cliente.

Usando el comando:

`#ifconfig`

Nos mostrará los diferentes adaptadores de red y sus configuraciones:

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:C6:70:C0
          inet addr:192.168.1.18  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec6:70c0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:59 errors:0 dropped:0 overruns:0 frame:0
          TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15999 (15.6 KiB)  TX bytes:6697 (6.5 KiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:98:DC:90
          inet6 addr: fe80::a00:27ff:fe98:dc90/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2736 (2.6 KiB)  TX bytes:1836 (1.7 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:480 (480.0 b)  TX bytes:480 (480.0 b)
```

Como podemos ver, este es el servidor Proxy ya que tiene dos adaptadores de red (a parte del *Local Loopback*, adaptador que te permite acceder a tu propio ordenador), eth0 sería el adaptador puente que ya tiene asignada una dirección IP por el servidor DHCP del router (y que sería el equivalente a la IP 193.146.96.1 del esquema de red del apartado anterior) y eth1 que no tiene asignada una dirección IP. Para asignarle una

dirección IP a eth1 ejecutamos el siguiente comando para abrir su fichero de configuración:

#gedit /etc/sysconfig/network-scripts/ifcfg-eth1

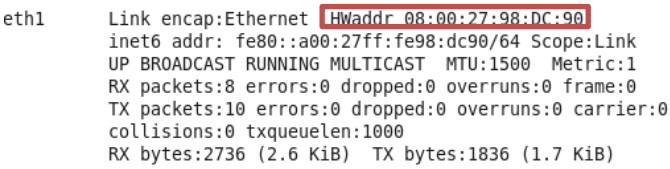
Nota: tener en cuenta que si el nombre del adaptador de red es otro, por ejemplo ethX, tenéis que abrir su correspondiente fichero de configuración ifcfg-ethX .

Se nos abrirá el fichero de configuración del adaptador eth1, el cual lo más seguro es que esté vacío, tenemos que dejar las siguientes líneas:

```
DEVICE=”eth1”
HWADDR=”XX:XX:XX:XX:XX:XX”
IPADDR=”X.X.X.X”
ONBOOT=”yes”
```

En el equipo servidor vamos a usar la IP 169.254.0.1, en el campo IPADDR, porque se suele usar esta primera IP del rango de IPs como la IP de los servidores. Si es el cliente cualquiera en el rango de 169.254.0.2 – 169.254.255.254.

HWADDR es la dirección mac del adaptador de red, la podemos saber mediante el comando ifconfig:



Una vez terminemos, guardamos los cambios y ejecutamos los siguientes comandos:

#ifconfig eth1 down

#ifconfig eth1 up

ó

#service network restart

Podemos comprobar que la configuración ha sido establecida correctamente mediante el comando ifconfig.

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:C6:70:C0
          inet addr:192.168.1.18  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec6:70c0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:153 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28193 (27.5 KiB)  TX bytes:8739 (8.5 KiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:98:DC:90
          inet addr:169.254.0.1  Bcast:169.254.255.255  Mask:255.255.0.0
          inet6 addr: fe80::a00:27ff:fe98:dc90/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:56 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5472 (5.3 KiB)  TX bytes:10448 (10.2 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:67 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6708 (6.5 KiB)  TX bytes:6708 (6.5 KiB)
```

IMPORTANTE: Si no le ponemos HWADDR a la configuración, esta configuración se aplicará a todos los adaptadores de red.

No olvidar configurar la tarjeta de red tanto en el servidor como en el cliente.

4. Configuración básica del servidor Squid:

Para realizar el caso práctico vamos a utilizar SQUID que es el software para servidor Proxy Web más popular y extendido entre los sistemas operativos Linux (también hay versión para Windows). Es muy confiable, robusto y versátil. Es software libre y gratuito y viene incluido en la mayoría de las distribuciones de Linux.

Funcionalidad de Squid

Entre otras cosas, Squid puede hacer de proxy y caché con los protocolos HTTP, FTP, GOPHER y WAIS, Proxy de SSL, caché transparente, WWCP, aceleración HTTP, caché de consultas DNS y otras muchas más como filtración de contenido y control de acceso por IP y por usuario.

Squid no puede funcionar como proxy para servicios como SMTP, POP3, TELNET, SSH, etc. Si se requiere hacer proxy para cualquier cosa distinta a HTTP, HTTPS, FTP, GOPHER y WAIS se requerirá implementar enmascaramiento de IP a través de NAT

(Network Address Translation) o instalar otros proxies que soporten estas aplicaciones.

El uso más extendido de Squid es como caché de páginas Web. Cachear una página Web significa guardar su contenido en el servidor Squid la primera vez que se la pide un cliente. La segunda vez que se la soliciten, Squid comprueba si esa página ha cambiado, si no lo ha hecho Squid envía al cliente la página que tiene almacenada en su servidor. Esto multiplicado por miles de accesos a páginas Web y cientos de usuarios dentro de una organización, reduce drásticamente el ancho de banda consumido. Del mismo modo es utilizado por los proveedores de servicio de Internet para reducir el ancho de banda consumido por sus clientes. De hecho, cuando las cachés no funcionan bien hay muchas quejas por parte de los usuarios de Internet ya que no están viendo la última versión de determinados sitios Web.

En este curso vamos a centrar nuestra atención en el uso de Squid como proxy, ya que lo que nos interesa en estos momentos es la seguridad de nuestra red.

En http://www.squid-cache.org existe abundante documentación sobre Squid, una sección de Faq, y está disponible el programa.

Instalación

Para poder saber la versión de squid que tenemos ejecutamos el comando:

```
[root@localhost ~]# rpm -q squid
squid-3.1.10-9.el6_3.i686
```

Si no está instalado debemos instalarlo con el comando:

```
# yum install squid
```

Squid proporciona un ejemplo de fichero de configuración /usr/share/doc/squid-*/squid.conf.documented que contiene mucha información sobre como configurar Squid, contiene más de 5600 líneas de comentarios y ejemplos.

El fichero de configuración de Squid es /etc/squid/squid.conf. Que, como podéis comprobar, trae una serie de directivas por defecto. Dejarlas tal cual vienen, en principio están preparadas para que Squid funcione directamente.

Configuración básica del servidor Squid.

4.1. Parámetro http_port ¿Qué puerto utiliza Squid?

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

Para especificar varios puertos:

```
# Squid normally listens to port 3128
http_port 3128

http_port 8080
```

Si desea incrementar la seguridad, puede vincularse el servicio a una IP. Si lo vinculamos a la IP privada del servido, sólo se pueda acceder al servicio desde la red local. Considerando nuestro esquema de red inicial, el servidor utilizado posee una IP 169.254.0.1, por lo que puede hacerse lo siguiente:

```
# Squid normally listens to port 3128
http_port 169.254.0.1:3128

http_port 169.254.0.1:8080
```

4.2. Controles de acceso:

Se puede incrementar la seguridad controlando el acceso que Squid proporciona a la red de área local, por ejemplo permitiendo acceder sólo desde un conjunto determinado de IPs, o a unos usuarios, o dependiendo del día y hora. También se pueden restringir los sitios a los que se accede, por ejemplo denegando la descarga de mp3, videos, etc. que suele ser un tráfico no deseado en las empresas y organismos.

El control se establece mediante las Listas de Control de Acceso, ACL, que definan una red o bien ciertas maquinas en particular. A cada lista se le asignará una Regla de Control de Acceso que permitirá o denegará el acceso a Squid. Procedamos a entender cómo definir unas y otras.

Listas de control de acceso:

Una lista de control de acceso tiene la siguiente sintaxis:

> acl [nombre de la lista] *type* [equipos o redes incluidos en la lista]

```
#
# Recommended minimum configuration:
#
acl manager proto cache_object
acl localhost src 127.0.0.1/32 ::1
acl to_localhost dst 127.0.0.0/8 0.0.0.0/32 ::1

# Example rule allowing access from your local networks.
# Adapt to list your (internal) IP networks from where browsing
# should be allowed
acl localnet src 10.0.0.0/8     # RFC1918 possible internal network
acl localnet src 172.16.0.0/12  # RFC1918 possible internal network
acl localnet src 192.168.0.0/16 # RFC1918 possible internal network
acl localnet src fc00::/7       # RFC 4193 local private network range
acl localnet src fe80::/10      # RFC 4291 link-local (directly plugged) machines
```

Por defecto, todos los dispositivos tienen denegado el acceso a los servicios de Squid (excepto localhost). Si se desea establecer una lista de control de acceso que defina a toda la red local, se utiliza la IP que corresponde a la red y la máscara de la subred. Por ejemplo, para nuestra red donde las máquinas tienen direcciones IP 169.254.n.n con máscara de subred 255.255.0.0, podemos utilizar lo siguiente:

```
acl miredlocal src 169.254.0.0/16
```

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

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

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

```
169.254.1.100
169.254.1.2
169.254.51.3
169.254.46.15
169.254.12.16
169.254.0.20
169.254.243.40
```

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*.

4.3 Reglas de Control de Acceso:

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

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

# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
http_access allow localnet
http_access allow localhost

# And finally deny all other access to this proxy
http_access deny all
```

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 *excepción*. Pueden definirse, por ejemplo, dos listas de control de acceso, una denominada *lista1* y otra denominada *lista2*. La siguiente regla 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.

Importante: El orden de las opciones en http_access es importante. Squid empieza por la primera y las procesa hacia abajo, deteniéndose en la primera opción de http_access con una entrada ACL que se cumpla, por eso es importante situar nuestras reglas de acceso antes de la regla "http_access deny all" existente en el fichero de configuración.

Opciones de ACL para Squid, el nombre de la lista de acceso es una expresión arbitraria. El método de actuación puede ser (lo que señalamos como type en la definición de una ACL):

src	Dirección de origen. La expresión coincidente puede ser una dirección IP, o un conjunto de direcciones (siempre en formato CIDR), nunca un host.

dst	Dirección de destino, con el mismo formato que la anterior
myip	Se evalúa en función de la dirección ip por la que le llegue al squid la petición. Se trata de su dirección IP.
srcdomain	Evalúa el nombre del host del que procede la petición.
dstdomain	Evalúa el destino de la petición a partir de su host.
srcdom_regex y dstdom_regex	Evalúa los nombres de dominio de origen o destino, pudiéndose utilizar expresiones regulares POSIX para encontrar coincidencias.
time	Evalúa en función de tiempo
url_regex	Regla muy útil. Evalúa toda la dirección que el cliente solicite, empezando desde http://.. incluido. Se utilizarán expresiones regulares.
urlpath_regex	Igual que la anterior, sólo que se empieza a evaluar a partir del primer / que se encuentre por detrás del nombre de dominio.
method	Método con el que se trate de obtener el contenido. Los más usuales son GET, POST, CONNECT y, en algunos casos, PUT.
maxconn	Esto se utiliza para detectar cuándo un cliente ha alcanzado un máximo de conexiones, que especificaremos en su expresión coincidente con un número entero positivo.
rep_mime_type	Se utiliza para evaluar en función del tipo de dato que el cliente solicite, pudiendo ser algo como application/x-javascript o application/x-msdos-program. Consultar /etc/mime.types para ver una lista.

Algunas de las acciones posibles son (se indican en las reglas de acceso):

http_access	Controlar si lo que coincide con la ACL tiene acceso al proxy
no_cache	En este caso, deny sirve para impedir que se cachee lo que coincida con el ACL y allow para forzarlo. Es útil utilizar una ACL de tipo urlpath_regex aquí. Ejemplo: acl NOCACHE urlpath_regex -i cgi-bin \? \.gz$ \.bz2$ no_cache deny NOCACHÉ Esta regla impide el cacheo de páginas con contenido dinámico y de archivos con extensiones (.gz, .bz2) que no merece la pena cachéar. Es además insensible a mayúsculas, por si cae algún *.ISO por ejemplo.

4.3. Ejemplos de Listas y Reglas de control de acceso:

4.3.1. Configuración Servidor:

Una vez visto el funcionamiento de la Listas y las Regla de Control de Acceso, procederemos a utilizarlas para nuestra configuración.

Accedemos a nuestra máquina virtual CentOS en la que hemos instalado el servidor Squid y hemos configurado las dos tarjetas de red. En nuestro ejemplo la red local es 169.254.0.0/16.

Si se desea definir una lista de control de acceso que abarque toda la red local, utilizamos la siguiente línea en la sección de Listas de Control de Acceso:

```
acl todalared src 169.254.0.0/16
```

Por lo que las Listas de Control de Acceso serán las siguientes (si eliminamos los acl que vienen por defecto hay que eliminar las referencias a estos en todo el fichero, sino dará errores al iniciar el servicio, ES MEJOR NO ELIMINARLAS). Habiendo hecho lo anterior, la sección de listas de control de acceso debe quedar más o menos del siguiente modo:

```
# Recommended minimum configuration:

acl all src 0.0.0.0/0.0.0.0

acl localhost src 127.0.0.1/32

acl todalared src 169.254.0.0/16
```

A continuación procedemos a aplicar la regla de control de acceso (en el apartado de "INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS"):

```
http_access allow todalared
```

Habiendo hecho lo anterior, la zona de reglas de control de acceso debería quedar más o menos de este modo:

```
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 169.254.0.0/16. Esto significa que cualquier máquina desde 169.254.0.1 hasta 169.254.255.254 podrá acceder a los servicios de Squid.

La regla **http_access deny all** deniega cualquier otro acceso al servidor que no se haya permitido anteriormente a esta sentencia.

Iniciando, reiniciando y añadiendo el servicio al arranque del sistema

Una vez terminada la configuración, arrancamos el servicio ejecutando el siguiente comando:

```
# service squid start
```

Si necesitamos reiniciar para probar cambios hechos en la configuración, hay que ejecutar:

```
# service squid restart
```

Pero es más recomendable recargar el fichero de configuración si el servicio está encendido, ya que aparte de tardar en parar el servicio, puede ser crucial en entornos donde la parada del servidor pueda cortar las comunicaciones existentes en ese momento):

```
# service squid reload
```

Si se desea que squid se inicie de manera automática la próxima vez que inicie el sistema, hay que ejecutar:

```
# /sbin/chkconfig squid on
```

IMPORTANTE: Desactivar el cortafuegos y SELinux en el servidor y en el cliente.

4.3.2. Configuración Clientes:

Para que los equipos de nuestra red local (ya sean Linux o Windows) accedan a la Web a través del proxy, hay que configurar los datos de nuestro servidor proxy en el navegador Web del cliente. En concreto habrá que indicar la dirección IP del servidor Proxy y el número de puerto en el que escucha las peticiones de servicio Web, recordar que por defecto es el 3128.

Por lo tanto, hay que configurar los datos del servidor proxy en el navegador Web que utilicemos en los clientes, vamos a ver como se hace en el Mozilla Firefox y el Internet Explorer.

Desde virtualBox arrancamos el Centos que hemos configurado dentro de nuestra red interna y que tendrá una IP dentro del rango 169.254.0.0/16. Sólo tendremos que configurar el navegador Web que utilicemos, en el caso de Centos lo más habitual es el Firefox. Ponemos la configuración del explorer por si alguien prefiere utilizar un Windows como cliente.

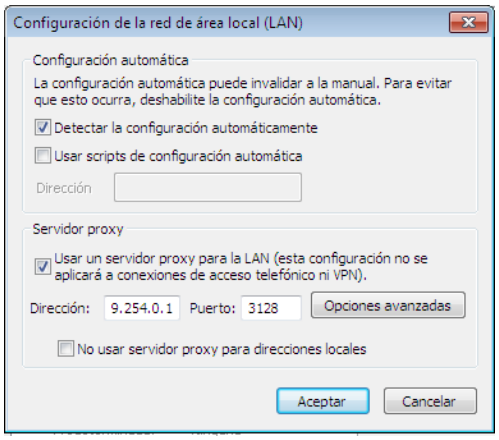
Configuración de Firefox para usar un proxy Web:

Abrimos el navegador Firefox, menú "Editar->Preferencias". En la ventana emergente seleccionamos el botón "Avanzado" y dentro de esta interfaz la pestaña "Red". Pulsamos sobre el botón "Configuración" del apartado "Configurar cómo Firefox se conecta a la red".

Donde se marca la opción "Configuración manual del proxy" y se introducen los datos como aparece en la captura, IP del servidor Proxy y número de puerto. Pulsamos aceptar. De esta manera la próxima vez que solicitemos una página Web desde Firefox, éste contactará con el servidor proxy para obtener su contenido.

Configuración de Internet Explorer para usar un proxy Web:

Abrimos el navegador Explorer, menú "Herramientas->Opciones de Internet". En la ventana emergente seleccionamos la pestaña "Conexiones" y pulsamos el botón "Configuración de LAN". Marcamos la opción "Usar un servidor proxy para la LAN" e introducir la dirección (169.254.0.1) y el puerto del proxy (3128).



Ejercicio: Ahora comprobamos que podemos acceder a Internet con la configuración establecida anteriormente.

```
acl todalared src 169.254.0.1/16

http_access allow todalared
```

NOTA: recordar recargar el fichero de configuración del servidor Proxy para que lea la nueva configuración.

Ejercicio: Crear una lista de nombre “nopermitidos” que incluya la IP de vuestro cliente y comprobar que con esta regla de acceso no podríais acceder a la Web.

```
acl nopermitidos src 169.254.X.X/16

http_access allow todalared  !nopermitidos
```

<u>Restricción de acceso a contenidos por extensión:</u>

Denegar el acceso a ciertos tipos de extensiones de fichero permite hacer un uso más racional del ancho de banda del que se dispone. El funcionamiento es verdaderamente simple, y consiste en denegar el acceso a ciertos tipos de extensiones que coincidan con lo establecido en una *Lista de Control de Acceso*.

Lo primero será generar una lista la cual contendrá direcciones Web y palabras usualmente utilizadas en nombres de ciertos dominios. Ejemplos:

```
\.avi$
\.mp4$
\.mp3$
\.mp4$
\.mpg$
```

```
\.mpeg$
\.mov$
\.ra$
\.ram$
\.rm$
\.rpm$
\.vob$
\.wma$
\.wmv$
\.wav$
\.doc$
\.xls$
\.mbd$
\.ppt$
\.pps$
\.ace$
\.bat$
\.exe$
\.lnk$
\.pif$
\.scr$
\.sys$
\.zip$
\.rar$
```

Esta lista, la cual deberá ser completada con todas las extensiones de fichero que el administrador considere pertinentes, la guardaremos como */etc/squid/listaextensiones*.

Parámetros en /etc/squid/squid.conf

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

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

A continuación modificaremos la regla de control de acceso existente agregando con un símbolo de *!* que denegará el acceso a la *Lista de Control de Acceso* denominada *listaextensiones*:

```
http_access allow redlocal !listaextensiones
```

La regla anterior permite el acceso a la *Lista de Control de Acceso* denominada *redlocal*, pero le niega el acceso a todo lo que coincida con lo especificado en la *Lista de Control de Acceso* denominada *listaextensiones*.

Ejercicio: Comprobar que no podéis descargar ficheros con las extensiones indicadas en el archivo listaextensiones.

Restricción de acceso por horarios:

Denegar el acceso en ciertos horarios permite hacer un uso más racional del ancho de banda del que se dispone y también es una medida de seguridad ya qué sólo se podrá acceder al servicio cuando lo determine el administrador. El funcionamiento es, de nuevo, simple, y consiste en denegar el acceso en horarios y días de la semana.

La sintaxis para crear *Listas de control de acceso* que definan horarios es la siguiente:

```
acl [nombre del horario] time [días de la semana] hh:mm-hh:mm
```

Los días de la semana se definen con letras, las cuales corresponden a la primera letra del nombre en inglés, de modo que se utilizarán del siguiente modo:

S - Domingo
M - Lunes
T - Mastes
W - Miercoles
H - Jueves
F - Viernes
A - Sábado
Ejemplo:

```
acl semana time MTWHF 09:00-21:00
```

Esta regla define a la lista *semana*, la cual comprende un horario de 09:00 a 21:00 horas desde el lunes hasta el viernes.

Este tipo de listas se aplican en las *Reglas de Control de Acceso* con una mecánica similar a la siguiente: se permite o deniega el acceso en el horario definido en la *Lista de Control de Acceso* denominada *X* para las entidades definidas en la *Lista de Control de Acceso* denominada *Y*.

```
http_access [allow | deny] [nombre del horario] [lista de entidades]
```

Ejemplo: Se quiere establecer que los miembros de la *Lista de Control de Acceso* denominada *clasematutina* tengan permitido acceder hacia Internet en un horario que denominaremos como *matutino*, y que comprende de lunes a viernes de 09:00 a 15:00 horas.

La definición para el horario sería:

```
acl clasematutina src 169.254.0.1/16
acl matutino time MTWHF 09:00-15:00
```

La definición de la *Regla de Control de Acceso* sería:

```
http_access allow matutino clasematutina
```

Lo anterior, en resumen, significa que quienes conformen *clasematutina* podrán acceder a Internet de Lunes a Viernes de 09:00-15:00 horas.

Ejercicio: Poner una restricción horaria para que en este momento no tengáis acceso al servidor Web, comprobar que no accedéis.

Ejercicio: Poner una restricción horaria para que en este momento si tengáis acceso al servidor Web, comprobar que accedéis.

5. Acceso por Autenticación:

Es muy útil el poder establecer un sistema de autenticación para poder acceder hacia Internet, pues esto permite controlar quienes sí y quienes no accederán a Internet sin importar desde que máquina de la red local lo hagan. De este modo podemos tener un doble control, primero por dirección IP y segundo por nombre de usuario y clave de acceso.

Para tal fin nos valdremos de un programa externo para autenticar, como es *ncsa_auth*, de la NCSA (National Center for Supercomputing Applications), y que ya viene incluido como parte del paquete principal de Squid en la mayoría de las distribuciones actuales.

1. Creación del fichero de claves de acceso.

Se requerirá la creación previa de un fichero que contendrá los nombres de usuarios y sus correspondientes claves de acceso (cifradas). El fichero puede localizarse en cualquier lugar del sistema, con la única condición que sea asequible para el usuario squid.

Debe procederse a crear un fichero /etc/squid/claves:

```
#touch /etc/squid/claves
```

Salvo que vaya a utilizarse un guión a través del servidor web para administrar las claves de acceso, como medida de seguridad, este fichero debe hacerse leíble y escribible solo para el usuario squid:

```
#chmod 600 /etc/squid/claves
#chown squid:squid /etc/squid/claves
```

2. A continuación deberemos dar de alta las cuentas que sean necesarias, utilizando el comando *htpasswd -mismo que viene incluido en el paquete httpd-2.0.x-*. Ejemplo:

```
#htpasswd /etc/squid/claves adrian
```

Lo anterior solicitará teclear una nueva clave de acceso para el usuario *adrian* y confirmarla tecleando ésta de nuevo. Repita con el resto de las cuentas que requiera dar de alta.

Todas las cuentas que se den de alta de este modo son independientes a las ya existentes en el sistema. Al dar de alta una cuenta o cambiar una clave de acceso lo estará haciendo **EXCLUSIVAMENTE** para el acceso al servidor Proxy. Las cuentas son independientes a las que se tengan existentes en el sistema como serían shell, correo y Samba.

3. Parámetros en /etc/squid/squid.conf

Lo primero será especificar que programa de autenticación se utilizará. Localice la sección que corresponde a la etiqueta auth_param basic program, si no se encuentra colocar al principio del documento para evitarse problemas. Por defecto no está especificado programa alguno. Considerando que ncsa_auth se localiza en /usr/lib/squid/ncsa_auth, procederemos a añadir el siguiente parámetro:

```
auth_param basic program /usr/lib/squid/ncsa_auth   /etc/squid/claves
```

/usr/lib/squid/ncsa_auth corresponde a la localización de el programa para autenticar y /etc/squid/claves al fichero que contiene las cuentas y sus claves de acceso.

El siguiente paso corresponde a la definición de una Lista de Control de Acceso. Especificaremos una denominada password la cual se configurará para utilizar

obligatoriamente la autenticación para poder acceder a Squid. Debe localizarse la sección de Listas de Control de Acceso y añadirse la siguiente línea:

```
acl password proxy_auth REQUIRED
```

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

```
acl localhost src 127.0.0.1/32
acl todalared src 169.254.0.0/16
acl password proxy_auth REQUIRED
```

Procedemos entonces a modificar la regla de control de accesos que ya teníamos para permitir el acceso a Internet. Donde antes teníamos lo siguiente:

```
http_access allow todalared
```

Le añadimos password, la definición de la *Lista de Control de Acceso* que requiere utilizar clave de acceso, a nuestra regla actual, de modo que quede como mostramos a continuación:

```
http_access allow todalared password
```

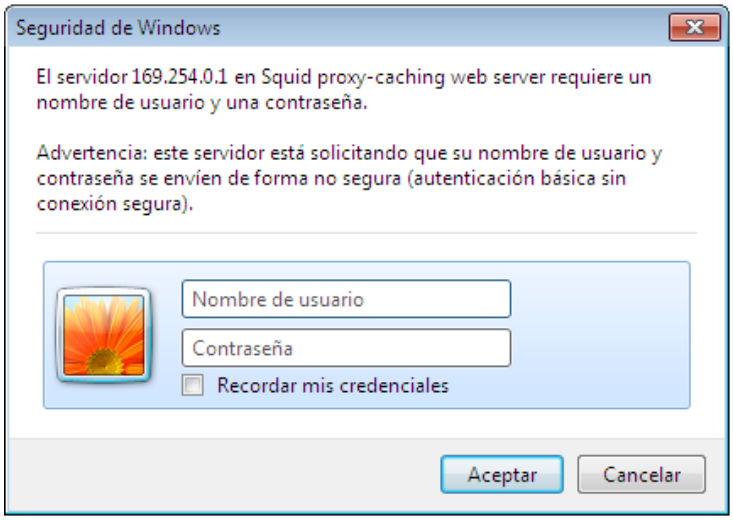
Habiendo hecho lo anterior, la zona de reglas de control de acceso debería quedar más o menos de este modo:

```
http_access allow localhost
http_access allow todalared password
http_access deny all
```

Tras guardar, recargamos el fichero:

```
# service squid reload
```

Y comprobamos que al acceder desde los clientes nos solicita un usuario y contraseña para poder continuar:



6. Ficheros log:

Squid genera varios ficheros de log:

- **/var/log/squid/access.log**: Aquí se guardan las peticiones que se le hacen al proxy, podemos saber cuánta gente usa el proxy, qué IPs están accediendo, qué páginas están solicitando...

- **/var/log/squid/cache.log**: Aquí se guardan los errores, mensajes de inicio...

- **/var/log/squid/store.log**: Aquí se guarda lo que pasa con la cache, que páginas (objetos) se añaden, cuales se quitan...

Existen herramientas que nos ayudan en el análisis de lo que ocurre en Squid, por ejemplo CALAMARIS. También existe una MIB que define objetos sobre el funcionamiento de SQUID, si configuramos SNMP con Squid podemos recoger desde cualquier gestor la información de Squid. Por ejemplo con MRTG podríamos sacar gráficas sobre los datos de Squid a partir de los datos recogidos por SNMP. Como veis, las herramientas que hemos ido viendo a lo largo del curso confluyen en muchas ocasiones para utilizarlas conjuntamente.

Configuración Calamaris:

1. Descargar desde http://calamaris.cord.de/ la última versión del programa.

2. Movemos el fichero al directorio en que queremos instalarlo y tecleamos: `#tar -zxvf calamaris-X.tar.gz`

3. Accedemos al directorio y ejecutamos:
   #cat /var/log/squid/Access.log | ./calamaris

```
Proxy-Report

Report period: 25.Nov 12 10:23:41 - 25.Nov 12 10:37:08
Generated at:  26.Nov 12 13:15:27

# Summary
lines parsed:              9
invalid lines:             0
parse time (sec):          1

# Incoming requests by method
method                          request      %    Byte       %   sec  kB/sec
-------------------------------- --------- ------ -------- ------ ---- -------
GET                                     9 100.00    74526 100.00    0   92.58
-------------------------------- --------- ------ -------- ------ ---- -------
Sum                                     9 100.00    74526 100.00    0   92.58

# Incoming UDP-requests by status
no matching requests

# Incoming TCP-requests by status
status                          request      %    Byte       %   sec  kB/sec
-------------------------------- --------- ------ -------- ------ ---- -------
HIT                                     0   0.00        0   0.00    0    0.00
MISS                                    4  44.44    53132  71.29    0   74.55
ERROR                                   5  55.56    21394  28.71    0  231.88
-------------------------------- --------- ------ -------- ------ ---- -------
Sum                                     9 100.00    74526 100.00    0   92.58

# Outgoing requests by status
status                          request      %    Byte       %   sec  kB/sec
-------------------------------- --------- ------ -------- ------ ---- -------
DIRECT Fetch from Source                4 100.00    53132 100.00    0   74.55
SIBLING                                 0   0.00        0   0.00    0    0.00
PARENT                                  0   0.00        0   0.00    0    0.00
-------------------------------- --------- ------ -------- ------ ---- -------
Sum                                     4 100.00    53132 100.00    0   74.55

# Outgoing requests by destination
neighbor type                   request      %    Byte       %   sec  kB/sec
-------------------------------- --------- ------ -------- ------ ---- -------
DIRECT                                  4 100.00    53132 100.00    0   74.55
-------------------------------- --------- ------ -------- ------ ---- -------
Sum                                     4 100.00    53132 100.00    0   74.55


Calamaris $Revision: 2.59 $
Copyright (C) 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004 Cord Beermann.
Calamaris comes with ABSOLUTELY NO WARRANTY. It is free software, and you are
welcome to redistribute it under certain conditions. See source for details.
Calamaris-Homepage: http://Calamaris.Cord.de/
```

Por último queremos comentaros otro posible uso del Proxy Squid, aunque no vamos a entrar en su configuración (conceptualmente es similar a la vista hasta ahora).

7. Proxy acelerado:

Hasta ahora hemos trabajado con el concepto de proxy normal que proporciona acceso a Internet a una red privada, además con la ventaja de que utiliza la caché para que el acceso sea más rápido y se consuma menos ancho de banda.

Otra opción de utilización de Squid es como proxy acelerado, en el que se invierte el rol, es decir: si tu red local tiene un servidor Web, en vez que desde Internet se puedan conectar directamente al servidor lo que se hace es montar un proxy acelerado, de ahí lo de invertir el rol, ahora hace de proxy para tu servidor web, los clientes que accedan a tu web, en vez de ir directamente, llegaran hasta tu proxy y éste hará las peticiones al servidor Web interno.

Con este concepto la principal ventaja es la seguridad, ya que únicamente tu proxy accede al servidor web, luego también puedes cachear dependiendo si el contenido de tu sitio lo permite o no, lo que revierte en velocidad, ya que liberas al servidor Web de procesos, (proxy acelerado).