JUNTA DE ANDALUCIA

CONSEJERIA DE EDUCACION
CEP Lora del Río

Squid, un proxy caché para GNU/Linux

Alberto Molina Coballes, José Domingo Muñoz Rodríguez y José Luis Rodríguez Rodríguez.

30 de abril de 2010

En este documento se describe la instalación y configuración del proxy-cache Squid para agilizar y a la vez controlar el acceso de una red local a Internet. Este documento forma parte del curso *Servicios en GNU/Linux. Portal Educativo*, organizado por el CEP de Lora del Río (Sevilla) en 2010.

Este trabajo es una obra derivada de la documentación del curso *Software Libre y Educacion: servicios de red, gestores de contenidos y seguridad* de José Angel Bernal, Fernando Gordillo, Hugo Santander y Paco Villegas.

Índice

1. Introducción

Un servidor proxy se utiliza para centralizar en él las comunicaciones de una red local con otras redes o equipos. Inicialmente la comunicación con proxy era una de las formas posibles de establecer comunicaciones con una red pública desde una red privada, aunque actualmente es más habitual hacer esto a través de NAT.

El funcionamiento de un servidor proxy es tal que los clientes de la red local en lugar de realizar las peticiones directamente a la red externa, se la solicitan al servidor proxy, éste establece la conexión con el equipo externo y después devuelve la petición al cliente inicial. Esto permite utilizar un proxy cuando se quiere:

- Controlar el acceso de los equipos de la red local a otras redes.
- Agilizar el acceso de la red local a otras redes, ya que habitualmente incluyen cachés que permiten reutilizar las consultas anteriores.

También es importante tener en cuenta que un servidor proxy funciona a nivel de aplicación, lo que implica que no existe un servidor proxy universal sino que cada proxy soporta una serie de protocolos. En el caso de Squid, estos son HTTP y FTP, aunque también es posible utilizarlo de forma limitada en otros protocolos como TLS, SSL, Gopher y HTTPS.

2. Conceptos sobre cachés

Los servidores que actúan de proxy-caché se pueden configurar de varias formas, la forma más simple es un solo servidor proxy-caché en la red[1] en el que todos los ordenadores pertenecientes a esa red accederán a este servidor, que será el que almacenará todos los datos. Cuando un usuario solicita al servidor una página, éste comprueba si fue actualizada desde que fue almacenada. Si tiene la versión actualizada ahorra al usuario final la descarga de la misma proporcionándosela directamente.

Otro método de configurar la salida a Internet de una red de ordenadores es creando una jerarquía de servidores proxy-caché. Los servidores en un nivel superior a un servidor son denominados padres (*parent*) y los que se encuentran al mismo nivel son hermanos o iguales (*sibblings*, *neighbor* o *peer*).

Cuando Squid obtiene una petición de un cliente, comprueba si el objeto solicitado está en el disco del servidor. Si está, comprueba que el objeto no ha caducado y procede a enviarlo al cliente. Si por el contrario, el objeto no está o ha caducado, comprueba que otras cachés (padres o hermanas) lo tengan, proceso que realiza enviando paquetes UDP a esas máquinas con la URL.

La otra caché comprueba, a continuación, si tiene dicho objeto en el disco duro y envía un mensaje indicando si lo posee o no. La máquina original espera las respuestas y después decide si debe obtener el objeto de la otra caché o debe ir directamente a por él.

Cuando existe una máquina hermana el servidor le solicitará la información que no tiene y en caso de no tenerla estos servidores accederá directamente al servidor web remoto. En caso que existan también servidores padre en la configuración, la información que no tengan los hermanos la solicitará al servidor padre, que a su vez la solicitará directamente al servidor web remoto.

[1] Éste será el caso que veamos en este curso.

3. Instalación

Para instalar Squid hay que utilizar simplemente:

```
avatar:~# aptitude install squid3
```

Los ficheros y directorios más importantes son:

- `/etc/squid3/`, donde se guardan los ficheros de configuración, fundamentalmente el fichero `squid.conf`.
- La documentación se encuentra la ubicación estándar `/usr/share/doc/squid3/`.
- `/var/spool/squid3/`, donde se almacenan las páginas *cacheadas*, es decir, las que se han traído de Internet y que se guardan mientras no caduquen para la próxima vez que las solicite alguien.
- `/var/log/squid3/`, donde se ubican los ficheros de registro de squid que son independientes del *syslog* del sistema.

Una vez instalado squid, podemos comprobar que está operativo comprobando las conexiones que están abiertas:

```
avatar:~# netstat -putan |grep squid
```

```
tcp      0    0 0.0.0.0:3128        0.0.0.0:*       LISTEN    2450/(squid)
udp      0    0 0.0.0.0:3130        0.0.0.0:*                 2450/(squid)
udp      0    0 0.0.0.0:56529       0.0.0.0:*                 2450/(squid)
```

Squid utiliza por defecto el puerto `3128/tcp` para comunicarse con los clientes, aunque es posible modificar este puerto y utilizar cualquier otro [2]. Para comunicarse con otros proxies de su jerarquía utiliza el protocolo ICP a través del puerto `3130/udp`, aunque en nuestro caso no utilizaremos más que un proxy y por tanto podemos ignorar las conexiones UDP.

4. Configuración elemental

El archivo de configuración que utiliza Squid es `/etc/squid3/squid.conf`. Este fichero es un tanto peculiar, ya que incluye muchos parámetros comentados que no se utilizan inicialmente y además incluye bastantes comentarios sobre la utilización y sintaxis de estos parámetros. En concreto, en la versión que estamos utilizando, el fichero de configuración de Squid consta de `4745` líneas (!).

Para tener una primera idea de algunos parámetros que se están utilizando de forma explícita, lo sacamos por la salida estándar, quitando las líneas que empiecen por un espacio en blanco (suponemos que son líneas en blanco) y por # (comentarios):

```
avatar:~# cat /etc/squid3/squid.conf |grep -v ^$|grep -v ^#
```

```
acl manager proto cache_object
acl localhost src 127.0.0.1/32
acl to_localhost dst 127.0.0.0/8
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
```

[2] El más frecuente en otros proxies es el `8080/tcp`

```
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access deny all
icp_access deny all
htcp_access deny all
http_port 3128
hierarchy_stoplist cgi-bin ?
access_log /var/log/squid3/access.log squid
refresh_pattern ^ftp:           1440    20%     10080
refresh_pattern ^gopher:        1440    0%      1440
refresh_pattern (cgi-bin|\?)    0       0%      0
refresh_pattern .               0       20%     4320
icp_port 3130
coredump_dir /var/spool/squid3
```

La mitad de las líneas del fichero de configuración son definiciones de listas de control de acceso (ACL) que veremos en la siguiente sección y el resto de directivas que aparecen no es necesario modificarlas inicialmente.

También hay que tener en cuenta que Squid asume valores por defecto para gran cantidad de parámetros, que son los que precede con una línea `#Default:`, inicialmente hay `238` parámetros con valores por defecto:

```
avatar:~# cat /etc/squid3/squid.conf |grep "#Default:"|wc -l
```

```
238
```

En resumen se puede decir que Squid es una aplicación que incluye una enorme cantidad de opciones y que permite ajustarla en detalle a las condiciones y necesidades de la red concreta en la que se encuentre. En un documento como éste no se pretende realizar un ajuste *fino* de la aplicación, sino esbozar las posibilidades que Squid ofrece modificando las directivas más importantes y mostrando algunos ejemplos del funcionamiento de las ACL.

4.1. Recursos dedicados a Squid

Squid no utiliza todos los recursos del equipo donde está instalado, sino que es necesario definir qué cantidad de memoria RAM y espacio en disco puede utilizar como máximo para la caché. Los parámetros por defecto son muy conservadores para evitar crear problemas iniciales en máquinas con pocos recursos, por lo que es lógico aumentarlos de acuerdo a los recursos disponibles.

Si quisiéramos asignar `128MB` de memoria RAM para la caché en RAM de Squid [3], deberíamos descomentar y modificar la siguiente línea:

[3] El valor por defecto es sólo 8MB

/etc/squid3/squid.conf

```
1566 cache_mem 128 MB
```

Si quisiéramos modificar el tamaño máximo de los objetos cacheados en RAM deberíamos modificar la línea:

/etc/squid3/squid.conf

```
1575 # maximum_object_size_in_memory 8 KB
```

El espacio en disco reservado para almacenar los distintos objetos que se piden a través del proxy se define con la directiva `cache_dir`. La ubicación, formato y tamaño de este espacio en disco está definido por:

/etc/squid3/squid.conf

```
1733 cache_dir ufs /var/spool/squid3 100 16 256
```

El formato genérico de esta directiva es:

```
cache_dir tipo directorio Mbytes L1 L2 [options]
```

- `tipo`. Tipo de sistema de almacenamiento a utilizar (ufs es el único que está definido por defecto en la instalación).
- `directorio`. Ruta del directorio que se va a utilizar para guardar los datos del caché.
- `Mbytes`. Cantidad de espacio en disco en Megabytes que se va a utilizar para el caché. Si queremos que utilice el disco entero es recomendable poner aquí un 20 % menos del tamaño.
- `L1`. Número de subdirectorios de primer nivel que serán creados bajo directorio.
- `L2`. Número de subdirectorios de segundo nivel que serán creados bajo cada subdirectorio de primer nivel.

Puesto que el contenido de este directorio va a cambiar con frecuencia, es recomendable ubicarlo colocarlo en una partición separada por varias razones:

- La caché podría sobrepasar al resto del sistema de archivos o de la partición que comparte con otros procesos.
- Cuanto más cambie un sistema de archivos, mayores son también las posibilidades de que se encuentre dañado. Mantener la caché en una partición limita la parte de su sistema completo de archivos que resulta dañado.

Dependiendo del tipo de tráfico de la red, es habitual definir un tamaño máximo para los archivos que se van a cachear, ya que archivos excesivamente grandes que no se vayan a descargar muchas veces, llenan la caché y no resultan útiles. Si quisiéramos establecer un tamaño máximo de 32 MB para los archivos de la caché de disco, tendríamos que descomentar y modificar la línea:

/etc/squid3/squid.conf

```
1772 maximum_object_size 32768 KB
```

4.2. Control de acceso

Una de las funciones principales de Squid es controlar el acceso de los equipos de la red local a otras redes y es posible realizar esto a través de listas de control de acceso o ACL. También es posible utilizar programas auxiliares como Dansguardian o Squidguard para estas funciones, aunque en este documento explicaremos la forma de hacerlo directamente con Squid.

El procedimiento que se sigue es definir las distintas ACL y posteriormente se permite o deniega el acceso a una determinada función de la caché[4]. La opción de configuración encargada es normalmente `http_access`, que permite o deniega al cliente el acceso a Squid. Es muy importante tener en cuenta que Squid lee las directivas de arriba a abajo para determinar qué regla aplicar.

El formato general de la directiva `acl` es:

```
acl nombreACL tipoACL cadena ...
acl nombreACL tipoACL "fichero"
```

donde:

- `nombreACL` es el nombre que corresponde a esta definición ACL.
- `tipoACL` es el tipo de elemento contenido en esta definición.
- `cadena` o `fichero` es el argumento apropiado al `tipoACL`, pudiendo haber más de un argumento en la lista.

Vamos a analizar las ACL inicialmente definidas:

/etc/squid3/squid.conf (FILTRADO)

```
1  acl manager proto cache_object
2  acl localhost src 127.0.0.1/32
3  acl to_localhost dst 127.0.0.0/8
4  acl SSL_ports port 443
5  acl Safe_ports port 80          # http
6  acl Safe_ports port 21          # ftp
7  acl Safe_ports port 443         # https
8  acl Safe_ports port 70          # gopher
9  acl Safe_ports port 210         # wais
10 acl Safe_ports port 1025-65535  # unregistered ports
11 acl Safe_ports port 280         # http-mgmt
12 acl Safe_ports port 488         # gss-http
13 acl Safe_ports port 591         # filemaker
14 acl Safe_ports port 777         # multiling http
15 acl CONNECT method CONNECT
```

Donde podemos ver que se puede definir una ACL múltiple mediante varias líneas con la misma ACL, como es el caso de *Safe_ports*. Las directivas *acl* simplemente sirven para hacer definiciones, para decidir si se permite o deniega un determinado proceso, se utilizan las siguientes líneas:

/etc/squid3/squid.conf (FILTRADO)

```
1  http_access allow manager localhost
2  http_access deny manager
```

[4] Una descripción más completa de las ACL de squid se puede encontrar en http://wiki.squid-cache.org/SquidFaq/SquidAcl

```
3 http_access deny !Safe_ports
4 http_access deny CONNECT !SSL_ports
5 http_access allow localhost
6 http_access deny all
```

1. Permite acceso al protocolo *cache_object* sólo desde la dirección IP `127.0.0.1`.
2. Deniega el acceso al protocolo *cache_object* en cualquier otra situación.
3. No permite conexiones a puertos no definidos en la acl *Safe_ports*.
4. No permite conexiones directas (sin cachear), salvo a los puertos definidos en la acl *SSL_ports*.
5. Permite acceder a squid desde la dirección IP `127.0.0.1`.
6. No permite acceso a squid en cualquier otra situación[5].

4.2.1. Permitir acceso a nuestra red local

Aunque Squid está instalado, las ACL definidas inicialmente no permiten conexiones de ningún equipo distinto de *localhost*. Para permitir acceder a los equipos de nuestra red es necesario crear una ACL con las características de nuestra red y anteponerla a la línea `http_access deny all`. Para ello podemos utilizar la ACL *localnet* definida aunque inicialmente comentada:

/etc/squid3/squid.conf

```
588 #acl localnet src 10.0.0.0/8     # RFC1918 possible internal network
589 #acl localnet src 172.16.0.0/12  # RFC1918 possible internal network
590 #acl localnet src 192.168.0.0/16 # RFC1918 possible internal network
```

en nuestro caso podríamos dejar la línea:

/etc/squid3/squid.conf

```
590 acl localnet src 192.168.2.0/24  # RFC1918 possible internal network
```

Y posteriormente permitir acceso a *localnet* con la directiva:

/etc/squid3/squid.conf

```
646 http_access allow localnet
```

4.2.2. Restricciones de acceso

Una vez que podemos acceder a otras redes desde la red local utilizando el proxy, podemos establecer restricciones a nuestros usuarios en función de la dirección o dominio destino, la hora, el día, el tipo de fichero solicitado, etc. Para que las acl que definamos no produzcan ningún conflicto con las predefinidas de Squid, es necesario colocarlas tras la línea:

/etc/squid3/squid.conf

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

A continuación mostraremos algunos ejemplos con las ACL más significativas.

[5] Formalmente al puerto 3128/tcp

Dominio origen Permite especificar un FQHN de un equipo concreto o el dominio de nuestra red local, su sintaxis es:

```
acl nombreACL srcdomain nombreDominio
```

Dominio destino Permite especificar un FQHN de un equipo concreto o un dominio externo, su sintaxis es:

```
acl nombreACL dstdomain nombreDominio
```

Expresión regular que concuerda con el nombre del cliente Permite especificar una expresión regular para el nombre de los clientes, su sintaxis es:

```
acl nombreACL srcdom_regex [-i] expresión
```

Expresión regular que concuerda con el nombre del servidor Permite especificar una expresión regular para el nombre de dominio destino, su sintaxis es:

```
acl nombreACL dstdom_regex [-i] expresión
```

Control por día y hora Permite controlar el acceso en función de la hora o el día, su sintaxis es:

```
acl nombreACL time [día] [h1:m1-h2:m2]
```

Expresión regular que concuerda con la URL Permite establecer reglas en función de expresiones regulares que aparezca en la URL de la petición, su sintaxis es:

```
acl aclname url_regex [-i] ^http://expresión
```

Expresión regular que concuerda con la ruta de la URL Permite establecer reglas en función de expresiones regulares ignorando el protocolo y el nombre del servidor, en ejemplo para definir ficheros gif sería:

```
acl aclname urlpath_regex [-i] \.gif$
```

Una vez definidas las ACL entra en juego la directiva `http_access`, que se utiliza para permitir o denegar el acceso de una determinada ACL.

Importante: Orden de las reglas. Hay que tener en cuenta que la lectura se realiza de arriba a abajo y que Squid deja de leer líneas de `http_access` en cuanto que encuentra una que aplicar. Supongamos el siguiente orden en las reglas que tenemos aplicadas:

```
http_access deny all
http_access allow localnet
```

En este caso, la segunda línea no se leerá nunca, ya que la anterior se aplica a todas las direcciones IP origen, incluyendo las definidas para la red local.

Las ACL son especialmente útiles cuando queremos prohibir el acceso a una lista de sitios inapropiados (enlaces a páginas web con contenido pornográfico, violento, descargas masivas, etc.). Squid no está optimizado para gestionar una larga lista de sitios, pero puede gestionar un número concreto de sitios sin problemas.


```
...
acl porno dstdom_regex \.playboy.com \.sex.com \.sevillafc.es # :-P
...
http_access deny porno
http_access allow localnet
http_access deny all
```

En este caso, las direcciones que serán consideradas como inadecuadas son las que tienen los dominios `playboy.com` o `sex.com`. Estas URL tendrán filtrado el acceso y no será posible acceder a ellas, tal como indica la directiva `http_access deny porno`. Si se piden otras URL, Squid pasará a evaluar la siguientes directivas, por lo que si el cliente se conecta dentro del rango permitido se cursará la petición. De lo contrario, la petición será rechazada.

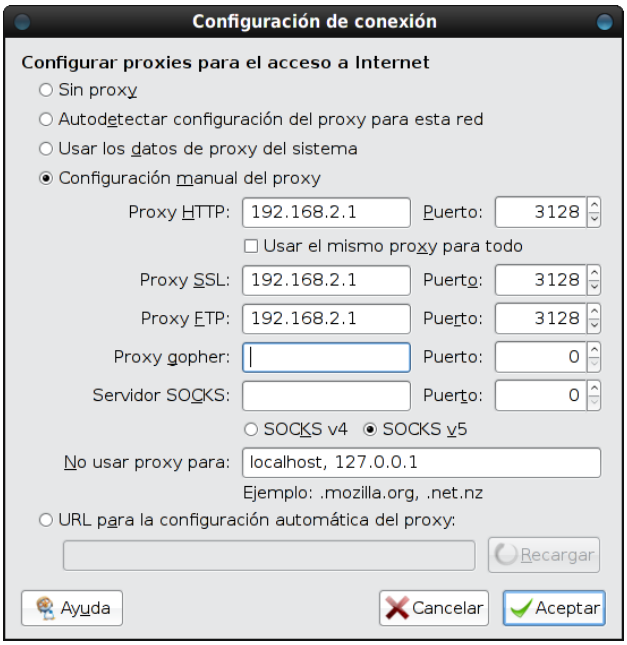
La limitación que tiene restringir acceso a diferentes dominios de la manera anterior es que cada vez que añadimos un dominio, hay que reiniciar Squid. Para evitar esto puede utilizarse un fichero donde ir añadiendo los dominios prohibidos, mediante la directiva:

```
acl prohibido dstdom_regex "/etc/squid3/dominios_prohibidos.txt"
```

Y deberíamos crear el correspondiente fichero con una línea por dominio.

5. Configuración de los clientes

El cliente de la red local lo deberemos configurar para que utilice el proxy para acceder a otras redes, por ejemplo Internet. En el caso del navegador Mozilla Firefox, seleccionamos `Editar` → `Preferencias` → `Avanzado` → `Red`→ `Configuración ...` → `Configuración Manual del proxy` y en los distintos protocolos ponemos la dirección del servidor proxy y el puerto `3128`, podemos especificarlo para todos los protocolos menos en el socks.



Veremos que nuestros accesos a Internet son mucho más rápidos y el control que podemos llegar a tener es muy importante.

Si trabajamos en modo consola y deseamos definir la variable de entorno[6] `http_proxy`, podemos usar:

```
avatar:~$ export http_proxy="http://192.168.2.1:3128"
```

Es útil, por ejemplo, para actualizar un sistema com `apt-get` que accede a Internet a través de un proxy, aunque en esos casos también es frecuente utilizar un proxy específico como *approx*.

[6] Si añadimos la variable a algún script de arranque se tomará como valor por defecto. Desde GNOME o KDE también podemos configurarla.

6. Autenticación simple de usuarios

Existe la posibilidad de utilizar el proxy sólo por usuarios que se autentiquen en el mismo, de manera que sea posible aplicar las reglas del proxy en función del usuario que lo utilice. Squid soporta diferentes mecanismos de autenticación como LDAP, Kerberos, etc., pero como la explicación de estos métodos excede el ámbito de este documento, mostraremos sólo la autenticación utilizando lo que se denomina autenticación básica. Para configurar Squid de este modo hay que utilizar la siguiente directiva:

```
auth_param basic program /usr/lib/squid3/ncsa_auth /etc/squid3/passwd
```

Donde especificamos que el programa `ncsa_auth` realizará la autenticación y el fichero donde se almacenará la información de autenticación [7].

Ahora habrá que crear una ACL para los usuarios que utilizan autenticación simple:

```
acl pass_web proxy_auth REQUIRED
```

Y para aplicarla podríamos poner:

```
http_access allow pass_web
http_access allow hostpermitidos
http_access deny all
```

7. Proxy transparente

En contraposición a la navegación a través de un proxy con autenticación que hemos visto anteriormente, existe la posibilidad de configurar Squid para que todos los usuarios de una red naveguen a través del proxy sin necesidad de tener que configurar cada una de sus aplicaciones, esto es lo que se denomina *proxy transparente*. Es más, se hace para que naveguen a través del proxy sin saberlo o sin quererlo, asumiendo todas las restricciones que éste imponga.

En primer lugar hay que utilizar la siguiente línea de iptables:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT \
--to-port 3128
```

En el caso de que el proxy se encuentre en la misma máquina que el cortafuegos, en caso de que sea en otra máquina (por ejemplo con IP 192.168.2.44), tendríamos que poner:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT \
--to 192.168.2.44:3128
```

Y modificar la línea `http_port 3128`, de manera que quede:

```
http_port 3128 transparent
```

En versiones anteriores a la 2.6, era necesario añadir varias directivas del tipo `httpd_accel`, pero actualmente se consigue el funcionamiento en modo transparente con esta simple modificación.

[7] Squid no genera las entradas de este fichero, para ello deberemos utilizar un programa externo como htpasswd incluido en el paquete apache2-utils