Servidor Proxy

El proxy Squid es un servidor web proxy-caché con licencia GPL cuyo objetivo es funcionar como proxy de la red y también como zona caché para almacenar páginas web, entre otros.
La página oficial de SQUID es http://www.squid-cache.org/.
¿Qué es un servidor proxy-caché?
Es un servidor situado entre la máquina del usuario y otra red (a menudo Internet) que actúa como protección separando las dos redes y como zona caché para acelerar el acceso a páginas web o poder restringir el acceso a contenidos.
Es decir, la función de un servidor proxy es centralizar el tráfico de una red local hacia el exterior (Internet). Sólo el equipo que incorpora el servicio proxy debe disponer de conexión a Internet y el resto de equipos salen a través de él.
Como las peticiones hacía Internet de los equipos de la red local son interceptadas por el servidor proxy, éste puede realizar una tarea de filtrado de accesos, impidiendo aquellos destinos que estén expresamente prohibidos en los archivos de configuración del servicio. Squid no es un filtro de contenidos pero puede actuar como tal.
En el aula se suele utilizar este servicio ya que permite llevar un control sobre la actividad de la red hacia el exterior del aula. En este caso lo usual es que el equipo que hace la función de servidor proxy disponga de dos interfaces de red. Una de ellas es utilizada para atender a la red local y la otra proporciona la conexión con Internet. Las peticiones de páginas web que se realizan desde el aula son interceptadas por la interfaz interna y reenviadas a la interfaz externa si cumplen los requisitos establecidos desde el servicio proxy.
Hay que tener en cuenta que la mayoría de los servidores web permiten la configuración como proxy-caché (Apache, IIS,...), pero Squid sólo es un proxy y no puede servir páginas por sí mismo.
Cuando decimos que Squid también funciona como caché significa que está guardando copia de los datos obtenidos de otras peticiones y de esa forma acelera el acceso a estos datos si se producen peticiones similares. Sólo se accederá de nuevo a las páginas originales cuando se detecte que se han producido modificaciones, es decir los datos almacenados difieren de los datos en el servidor web de origen.
Normalmente no existe una sola caché, sino que se tienen varios servidores (en máquinas diferentes) relacionados entre sí mediante una estructura en árbol.
Como resumen, las principales funciones de Squid son las siguientes:

- Permite el acceso web a máquinas privadas (IP privada) que no están conectadas directamente a Internet.
- Controla el acceso web aplicando reglas.
- Registra el tráfico web desde la red local hacia el exterior.
- Controla el contenido web visitado y descargado.
- Controla la seguridad de la red local ante posibles ataques, intrusiones en el sistema, etc.
- Funciona como una caché de páginas web. Es decir, almacena las páginas web visitadas por los usuarios y de esta manera las pueden enviar a otros usuarios sin tener que acceder a Internet de nuevo.
- Guarda en caché las peticiones DNS e implementa una caché para las conexiones fallidas.
- Registra logs de todas las peticiones cursadas.
- Soporta el protocolo ICP que permite integrar cachés que colaboran y permite crear jerarquías de cachés y el intercambio de datos.

Uno de los términos más confusos y a la vez de mayor utilidad en un servidor proxy Squid es la 'caché de disco'. Cuando hablamos de 'caché de disco', muchos la relacionan erróneamente con la 'caché de buffer', de la cual también hablaremos.
Tener una caché de disco en un servidor proxy linux con Squid representa un ahorro considerable en ancho de banda y navegabilidad. Esto se debe a que almacena los sitios webs que visitan los usuarios de nuestra LAN, para que la próxima vez que visiten el mismo sitio, la caché del proxy sea quien le facilite la página, y solo se actualiza cuando haya cambios (al menos en teoría).
Esta caché puede tener otras; en dependencia del nivel se puede configurar como padre o hermano. En otras palabras, se pueden establecer múltiples discos duros dedicados a "cachéar" páginas webs, interconectados entre sí, para acelerar la navegación.

Configurando la Caché de Disco

Para establecer el espacio del disco duro para la caché, se necesita modificar el parámetro correspondiente a la caché de directorios en el Squid.

Este parámetro sirve para definir tanto la ubicación de la caché de navegación como su tamaño, por default viene comentado y apunta a c:/squid/var/caché, pero si deseas aumentarlo necesitas modificarlo.

cache_dir tipo ruta espacio dir1 dir2

ruta: dirección dentro de nuestro disco duro donde alojaremos la caché. (Típicamente es: c:/squid/var/logs/cache)

espacio: espacio en mega bytes que se destinara para la caché, según el espacio que quieras asignar y el disponible en el disco duro puedes poner desde 10MB hasta 10GB o mas, para este ejemplo asignaremos 1GB.

dir1: indica los directorios de primer nivel en los que se dividirá la caché.

dir2: indica los directorios de segundo nivel en los que se dividirá la organización de la caché.

tipo: El algoritmo que squid usara para leer, escribir y borrar la caché, puedes elegir entre ufs, aufs, diskd, coss y null

Ufs el es algoritmo que squid usa por default para escribir y leer la caché, también es el más lento y antiguo, todo el proceso de lectura, escritura, recepción de peticiones, envío de objetos, etc. se hace con un solo demonio. Cuando tienes muchos accesos simultáneos se pueden producir cuellos de botella considerables, para mejor esto existen otras opciones.

AUFS es en esencia mismo que ufs, la diferencia es que lanza múltiples demonios para la lectura, escritura, borrado, etc. Como contra consume una gran cantidad de procesador y memoria cuando se ve ante una carga de trabajo alta.

DISKD también basado en ufs y aufs, la diferencia en este caso es que solo levanta un demonio extra, sería como ufs multiplicado por dos.

COSS Utiliza un sistema de ficheros especial y optimizado para squid, toda la información se guarda en un solo fichero y se escriben en bloques de datos. Teóricamente es el mejor ya que ofrece un mayor rendimiento que las otras opciones, desgraciadamente no es del todo estable por lo cual no es recomendable usarlo, aparte al usarlo en squid para Windows, estamos limitados al tamaño máximo que pueda tener un archivo en este sistema.

NULL No guarda ninguna caché en el disco duro.

Ejemplo: Aumentamos la caché a 1GB usando ufs.

cache_dir ufs c:/squid/var/caché 1000 16 256

Edite su archivo de configuración según la versión de Squid que vaya a usar:

```
sudo /etc/squid/squid.conf

o

sudo /etc/squid3/squid.conf
```

Busque la línea

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

```
o

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

Y cambie el valor 100, que viene por defecto, por el tamaño que quiera asignarle expresado en MB
Ejemplo de asignación de 100GB aproximadamente en un disco de 200GB (50%):

```
#Default:

cache_dir ufs /var/spool/squid 100000 16 256
```

NOTA: Cambie **ufs** por **aufs** para mejorar el rendimiento.

```
#Default:

cache_dir aufs /var/spool/squid 100000 16 256
```

En caso de disponer de más discos para repartir entre ellos el caché, se pueden especificar varias líneas de *caché_dir*

Vaciando y reconstruyendo la caché

En ocasiones la Caché de Disco necesita una purga general. Esto se debe a que se llena demasiado y en algunos SO basados en Linux, (al parecer un bug) la caché se desborda o no respeta los parámetros que le hemos dado en el archivo de configuración *squid.conf*, causando saturación y ralentización en la conectividad y en nuestro servidor. Este proceso también debe hacerse cuando actualizamos la versión de Squid.

Detenemos Squid

```
sudo service squid stop
```

net stop squid (Windows)

Nos aseguramos que esté apagado

```
sudo service squid status
```

Borramos la carpeta de la caché

```
rm -rf cache/
```

Rmdir /q /s cache

Creamos la carpeta nueva

```
mkdir cache
```

Cambiamos los permisos

```
chown proxy:proxy  caché/
```

Creamos directorios

```
squid -zX
```

squid –z

Iniciamos Squid

```
sudo service squid start
```

net start squid (Windows)

Caché de Buffer
La "otra" caché, más conocida como Caché de Buffer, define la cantidad máxima de memoria RAM que utilizará Squid para sus procesos. La cantidad se establece de acuerdo a la RAM física total asignada al servidor y teniendo en cuenta los demás procesos. Su valor por defecto es 8 MB (en algunas versiones de Squid este valor es diferente)
Lo ideal es por cada 256MB asignar 32MB. O sea si el servidor tiene 4GB de RAM, se le debe asignar 512MB aproximadamente, aunque esto es relativo y depende mayormente de la RAM libre de su servidor.

```
#Default:

cache_mem 512 MB
```

Para consultar el consumo de la RAM, ejecute el comando

```
free -m
```

tasklist /FI “SERVICES eq SQUID”

Otras cachés y configuraciones
1. Ajuste "caché_swap_low" y "caché_swap_high", al valor por default. Estas opciones se encargan de realizar la limpieza de la caché al llegar a los porcentajes indicados (90% y 95%)

```
cache_swap_low 90

cache_swap_high 95
```

2. Ajuste "memory_pools" al valor por default "off", para que Squid libere memoria RAM que no está usando el servidor y la coloque en la fuente de memoria.

```
memory_pools off
```

3. Ajuste "half_closed_clients" al valor por default "off". (Si es un proxy inverso, póngalo en "on"). Cerrar conexiones de los clientes no cerradas correctamente.

```
half_closed_clients off
```

4. Evite que Squid se atasque y ponga los "dns_nameservers" de su red local.

```
dns_nameservers 8.8.8.8 8.8.4.4
```

Cache_replacement_policy
A través de un parámetro (cache_replacement_policy) Squid incluye soporte para los siguientes algoritmos para el caché:

- LRU Acrónimo de Least Recently Used, que traduce como Menos Recientemente Utilizado. En este algoritmo los objetos que no han sido accedidos en mucho tiempo son eliminados primero, manteniendo siempre en el caché a los objetos más recientemente solicitados. Ésta política es la utilizada por Squid de modo predefinido. No tiene en cuenta el tamaño de los objetos.

- LFUDA Acrónimo de Least Frequently Used with Dynamic Aging, que se traduce como Menos Frecuentemente Utilizado con Envejecimiento Dinámico. En este algoritmo los objetos más solicitados permanecen en el caché sin importar su tamaño optimizando la eficiencia (hit rate) por octetos (Bytes) a expensas de la eficiencia misma, de modo que un objeto grande que se solicite con mayor frecuencia impedirá que se pueda hacer caché de objetos pequeños que se soliciten con menor frecuencia.

- GDSF Acrónimo de GreedyDual Size Frequency, que se traduce como Frecuencia de tamaño GreedyDual (codicioso dual), que es el algoritmo sobre el cual se basa GDSF. Optimiza la eficiencia (hit rate) por objeto manteniendo en el caché los objetos pequeños más frecuentemente solicitados de modo que hay mejores posibilidades de lograr respuesta a una solicitud (hit). Comportamiento mejor con objetos pequeños.

NOTA: De acuerdo al portal de Squid, el que ha demostrado mejor desempeño en escenarios de alta carga de trabajo es **heap LFUDA**.

Ejemplos:
cache_replacement_policy heap LRU
memory_replacement_policy heap LRU

maximum_object_size_in_memory y maximum_object_size

Sirve para asignar el tamaño máximo de los archivos a guardar en la caché, el default es 4MB, es decir, mientras no asignemos este parámetro squid no guardara ningún objeto mayor a 4MB cosa relativamente común en la actualidad, el tamaño debe definirse en kilobytes.

Ejemplo: asignar que guarde objetos de hasta 200MB.

maximum_object_size 204800 KB

maximum_object_size_in_memory 10 KB (memoria RAM)
maximum_object_size 5120 KB (disco duro)

dns_nameservers

Este parámetro sirve para asignar los dns manualmente, en algunos casos hay problemas con la resolución de los dominios y el asignar los dns arregla el problema.

Ejemplo:

dns_nameservers 208.67.222.222 208.67.220.220

reference_age

Establece el tiempo máximo que estará el contenido en la caché sin ser requerido, esto nos ayuda a no tener demasiado contenido en la caché al eliminar lo que no es usado.

Ejemplo: Asignando dos días de duración en la caché.

reference_age 2 days

cache_peer

Sirve para asignar servidores Proxy intermediarios (padres) y hermanos (paralelos).

Sintaxis:

cache_peer servidor tipo http_port icp_port opciones

servidor: dirección ip del servidor Proxy o en su defecto nombre de dominio del mismo.

tipo: parent (padre) cuando pasamos por el para salir a Internet o sibling (hermano) o en paralelo.

http_port: puerto http, típico 3128 y 8080.

icp_port: puerto icp.

Ejemplo: Asignaremos que nuestro Proxy sale a Internet mediante otro Proxy.

cache_peer 192.168.1.1 parent 8080 3130 Proxy-only

refresh_pattern

Es para establecer el tiempo de refresco entre determinados elementos, es decir, el tiempo que estarán estos elementos en caché antes de comprobar que haya nuevos.

refresh_pattern [-i] regex min porcentaje max [options]

Donde: regex: Expresión regular puede ser para tipos de archivos \.mp3 \.avi o protocolos ^ftp: ^gopher: general .

min: tiempo en minutos que estará un objeto antes de refrescarse

porcentaje: porcentaje de la edad de los objetos sin tiempo de caducidad que se consideraran para refrescarse.

max: tiempo máximo en minutos sin una explícita duración asignada.

Tenemos que partir del hecho que en la internet actual, donde prima el contenido dinámico, no es lógico cachear todos los contenidos de la web, ya que seguramente perderíamos funcionalidad en muchas páginas e incluso podríamos estar ofreciendo contenido desactualizado. Sin embargo, es posible forzar que squid guarde en caché determinados tipos de ficheros que sabemos que no se suelen generar de forma dinámica, por ejemplo imágenes, hojas de estilo, archivos flash, archivos comprimidos e incluso archivos pdf.

Para conseguir esto, utilizamos el parámetro de configuración refresh_pattern en el fichero /etc/squid3/squid.conf, con la siguiente sintaxis:

El parámetro min esta expresado en minutos, e indica el límite inferior en que el objeto se va a considerar invalido, es decir mientras el tiempo que el objeto está almacenado en caché es menor que este tiempo se considera válido. De la misma forma el parámetro max, expresado también en minutos, indica el máximo tiempo que un objeto puede ser considerado válido, es decir si un objeto lleva más tiempo almacenado en caché que el indicado en el parámetro max se considera inválido.

Si el tiempo que un objeto lleva guardado en la caché está entre estos dos valores, ¿es válido o está caducado? En este caso entra en juego el last-modified factor (LM-factor), que es la relación que existe entre el tiempo que lleva el objeto guardado en la caché y la edad que tiene. ¿Y cómo podemos calcular la edad? Pues será el tiempo transcurrido desde la última modificación del objeto (cabecera last-modified) y el momento en que se ha recibido (cabecera date). Es decir, si el LM factor es menor que el porcentaje indicado en el parámetro percent, el objeto se considerará válido. Pongamos un ejemplo: si un objeto que acabamos de recibir tiene 6 horas de edad, es decir la última modificación se realizó hace 6 horas, y hemos indicado un porcentaje del 50% el objeto se considerará válido en la caché las próximas 3 horas.

Por lo tanto podemos resumir el algoritmo que sigue el refresh_pattern de la siguiente manera:

La respuesta está caducada si el tiempo guardado en caché es mayor que el tiempo max.

La respuesta es válida si el LM-factor es menor que el porcentaje que hemos indicado en la directiva refresh_pattern.

La respuesta es válida si el tiempo guardado en caché es menor que el tiempo min.

En cualquier otro caso, la respuesta se considera caducada.

Ejemplo: evitando que los mp3 se descarguen más de una vez en dos meses.

refresh_pattern \.mp3 43200 90% 43200

Ejemplo 2: evitando que el contenido general se descargue más de una vez al día.

refresh_pattern . 1440 90% 1440

Recuerde que a mayor tamaño, mayor serán los archivos que almacenará el Squid y más rápido se llenará la caché.

```
maximum_object_size 100 MB

caché_replacement_policy heap LFUDA

refresh_pattern ^ftp:          1440    20%     10080

refresh_pattern ^gopher:       1440    0%      1440

refresh_pattern Packages\.bz2$ 0       20%     4320 refresh-ims

refresh_pattern Sources\.bz2$  0       20%     4320 refresh-ims

refresh_pattern Release\.gpg$  0       20%     4320 refresh-ims
```

```
refresh_pattern Release$       0       20%     4320 refresh-ims

refresh_pattern .              0       20%     4320
```

error_directory

Este comando sirve para definir el directorio de los mensajes de error para squid, para nuestra fortuna, la versión de squid para Windows ya incluye los mensajes traducidos al español.

Ejemplo: Asignamos los mensajes de error en español.

error_directory c:/squid/share/errors/Espanish

Rotación de archivos logs.

logfile_rotate 7

squid -k rotate (Linux)

squid –n squid –k rotate (Windows)

cache_access_log /var/log/squid/access.log

cache_log /var/log/squid/cache.log

cache_store_log /var/log/squid/store.log

Concepto de ACL en Squid

Un ACL es una definición de control de acceso, que en Squid se especifica mediante el parámetro acl según la siguiente sintaxis:

acl nombre_acl tipo_acl descripción ...

acl nombre_acl tipo_acl "fichero_de_descripciones" ...

Permitir o denegar una acl:

La sintaxis básica es la siguiente:

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

OR

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

OR

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

Tipos de ACL

src

Especifica una dirección origen de una conexión en formato IP/máscara. Por ejemplo, utilizaremos una acl de tipo src para especificar la red local:

acl red_local src 192.168.1.0/24

También podemos especificar rangos de direcciones mediante una acl de tipo src:

acl jefes src 192.168.1.10-192.168.1.25/16

dst

Especifica una dirección destino de una conexión en formato IP/máscara.

acl google_es dst 216.239.0.0/24

También podemos especificar hosts concretos mediante una acl de tipo dst:

acl google_es2 dst 216.239.59.104/32 216.239.39.104/32 216.239.57.104/32

Las definiciones son idénticas a las acl de tipo src salvo que se aplican al destino de las conexiones, no al origen.

srcdomain y dstdomain

Estos tipos de acl especifican un nombre de dominio.

En el caso de srcdomain es el dominio origen y se determina por resolución DNS inversa de la IP de la máquina, es decir, tendremos que tener bien configurado el DNS de la red local.

En el caso de dstdomain el nombre del dominio se comprueba con el dominio que se haya especificado en la petición de página web.

Por ejemplo: acl google_com dstdomain google.com

srcdom_regex y dstdom_regex

Especifican una expresión regular que verifican los dominios origen o destino. La expresión regular hace distinción entre mayúsculas y minúsculas salvo que incluyamos la oción "-i" que evita dicha distinción.

Por ejemplo

acl google_todos dstdom_regex -i google\..*

Observamos como al incluir "-i" estamos indicando que no haga distinción entre mayúsculas y minúsculas.

time

Este tipo de acl permite especificar una franja horaria concreta dentro de una semana. La sintaxis es la siguiente:

acl nombre_acl_horaria time [dias-abrev] [h1:m1-h2:m2]

Donde la abreviatura del día es:

S - Sunday (domingo)

M - Monday (lunes)

T - Tuesday (martes)

W - Wednesday (miércoles)

H - Thursday (jueves)

F - Friday (viernes)

A - Saturday (sábado)

además la primera hora especificada debe ser menor que la segunda, es decir h1:m1 tiene que ser menor que h2:m2

Por ejemplo

acl horario_laboral time M T W H F 8:00-15:00

Estaríamos especificando un horario de 8 a 15 y de lunes a viernes.

url_regex

Permite especificar expresiones regulares para comprobar una url completa, desde el http:// inicial.

Por ejemplo, vamos a establecer una acl que se verifique con todos los servidores cuyo nombre sea adserver:

url_regex serv_publicidad ^http://adserver.*

En otro ejemplo podemos ver una acl que verifique las peticiones de ficheros mp3:

url_regex ficheros_mp3 -i mp3$

Nota: ver expresiones regulares

referer_regex

Define una acl que se comprueba con el enlace que se ha pulsado para acceder a una determinada página. Cada petición de una página web incluye la dirección donde se ha pulsado para acceder. Si escribimos la dirección en el navegador entonces estaremos haciendo una petición directa.

Por ejemplo vamos a establecer una acl para todas las páginas a las que hayamos accedido pulsando en una ventana de búsqueda de google:

acl pincha_google referer_regex http://wwww.google.*

req_mime

Las acl de tipo req_mime se utilizan para comprobar el tipo de petición mime que realiza un cliente, y se puede utilizar para detectar ciertas descargas de ficheros o ciertas peticiones en túneles HTTP.

Esta acl sólo comprueba las peticiones que realiza el cliente, no comprueba la respuesta del servidor. Esto es importante para tener claro qué estamos haciendo y qué no.

Por ejemplo

acl subida req_mime_type -i ^multipart/form-data$

acl javascript req_mime_type -i ^application/x-javascript$

acl estilos req_mime_type -i ^text/css$

acl audiompeg req_mime_type -i ^audio/mpeg$

rep_mime_type

Este tipo de acl se utiliza para verificar el tipo de respuesta recibida por el proxy. Este tipo de acl, analiza una respuesta del servidor por lo que sólo le afectas las reglas de respuesta como http_reply_access y no las reglas http_access que se aplican a las peticiones.

Por ejemplo

acl javascript rep_mime_type -i ^application/x-javascript$

acl ejecutables rep_mime_type -i ^application/octet-stream$

acl audiompeg rep_mime_type -i ^audio/mpeg$

http_access

Este es el parámetro que permite o deniega accesos a una o más acl.

La sintaxis de uso es:

http_access allow|deny [!]acl ...

Observamos cómo cada acl puede ir precedida por un signo "!" que indicaría que lo contrario.

Por ejemplo, para permitir acceso fuera del horario laboral, según una acl que definimos anteriormente:

http_access allow !horario_laboral

Para denegar el acceso en horario laboral

http_access deny horario_laboral

Para dar acceso completo a la red local

http_access allow red_local

Por ejemplo

acl red1 src 192.168.0.0/24

acl red2 src 192.168.1.0/24

http_access allow red1 red2

Permitiría el acceso a todas aquellas conexiones que procedieran a la vez de red1 y de red2. Para permitir acceso a las dos redes podríamos:

acl red1 src 192.168.0.0/24

acl red2 src 192.168.1.0/24

http_access allow red1

http_access allow red2

o también, de forma más simple:

acl redes src 192.168.0.0/24 192.168.1.0/24

http_access allow redes

Ejemplos de Acl y http_access

Denegar las peticiones de ficheros de tipo exe, pif, bat y cmd

acl tipoexe url_regex -i exe$ pif$ bat$ cmd$

http_access deny tipoexe

Denegar el acceso a todos los servidores cuyo nombre comience por "popup", "banner" o "ads"

acl incordios url_regex http://popup.* http://ads.* http://ads.*

http_access deny incordios

Denegar todos los ficheros de tipo exe que provengan de virus_fijo.com

acl anti_exe url_regex -i exe$

acl vfijo dstdomain virus_fijo.com

http_access deny anti_exe vfijo

Denegar las conexiones a las url completas o incompletas que hay en el fichero "/etc/squid/lista_negra_1.txt"

acl ln1 url_regex "/etc/squid/lista_negra_1.txt"

http_access deny ln1

Autenticación de usuarios

auth_param basic program "c:/squid/libexec/ncsa_auth.exe" "c:/squid/etc/claves"

auth_param basic realm Servidor Proxy Super

acl autenticar proxy_auth REQUIRED

http_access allow autenticar

http_access deny all

Con estos parámetros sólo nos crea crear los usuarios. Hay que diferenciar los usuarios del sistema con los usuarios del Proxy. Añadimos los usuarios con el comando htpasswd

Ejemplo:

Htpasswd -c archivodeclave nombre_del_nuevo_usuario (con la –c es sólo la primera vez, y es para crear el fichero de claves)

Htpasswd archivodeclave nombre_del_nuevo_usuario

La clave se puede almacenar en texto plano, o cifrada con md5, sha.

Ejemplo:

Htpasswd /etc/squid julia –m (añade el usuario Julia con encriptación md5)

Por medio de RADIUS

auth_param basic program "c:/squid/libexec/squid_radius_auth.exe" -f "c:/etc/squid_radius_auth.conf"

auth_param basic realm Proxy con autenticación RADIUS

acl radius-auth proxy_auth REQUIRED

http_access allow radius-auth

http_access deny all

Quiero controlar el canal que usa Squid

Si está plenamente convencido de ello entonces, los Delay Pools son de su interés. Los Delay Pools son la herramienta para llevar a cabo el control de ancho de banda del Proxy (rate limiting y traffic shaping). La belleza de estos radica en que controlan el ancho de banda, sin causar penalidades sobre los objetos traídos desde el caché. En lenguaje técnico de Proxy, los Delay Pools afectan los caché misses, no los caché hits.

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

Antes de hablar sobre ellas imagine que un Delay Pool, especialmente uno de la clase 1, es como un tanque de agua el cual tiene un tubo de entrada y otro de salida. El tubo de entrada debido a su diámetro y a la apertura de la llave de paso solo permite que el agua entre a una rata fija. El tubo de salida debido a su gran diámetro no tiene estas limitaciones y puede vaciar el tanque inmediatamente si la llave de paso se abre lo suficiente. Finalmente, el tanque siempre almacena una cantidad máxima de agua.

Clases de Delay Pools

En Squid 2.x existen tres clases de Delay Pool.

Clase 1

El Delay Pool clase 1 define una única estructura de control (en nuestra abstracción un solo tanque).

Este limita el uso del canal de manera global sin importar cómo lo usan los clientes internamente o cómo está definida lógicamente la LAN. En el inglés técnico se habla de la definición de un único aggregate bucket. Ésta es la opción indicada si usted desea limitar el ancho de banda que usa Squid, sin importar cómo lo emplean los usuarios.

Clase 2

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

Clase 3

Este es un Delay Pool clase 1 con 256 Delay Pools clase 2 subordinados a este. En ingles técnico un aggregate bucket, 256 network buckets, y 65,536 individual buckets..

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

Sintaxis:

delay_pools N

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

Ejemplo:

delay_pools 3

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

Definición de la clase

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

Syntaxis:

delay_class id class

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

Ejemplo:

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

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

delay_class 3 2 ## el Dalay Pool número 3 será clase 2

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

Parámetros del Delay Pool

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

Sintaxis:

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

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

Ejemplos:

delay_parameters 1 76800/230400 42800/100000 10000/70000

Un Delay clase 3 con 600Kb/s (76800B/s) en total para navegación, con un tamaño para ráfagas (burst) globales de 1800Kb (230400Bytes). Para cada subred se asigna un canal máximo de 334.3Kb/s un tamaño para ráfagas de 781.2Kb (100000Bytes) con un ancho de banda para cada host de 78.1Kb/s (10000B/s) con la posibilidad de ráfagas de descargas de 546.8Kb (70000Bytes).

Note que los valores para cada subred y host exceden los límites de canal disponible si hay más de 4 clientes navegando en una subred o dos subredes demandan todo el canal asignado. En este caso se produce una condición de competencia y el primero que solicita el canal es el que lo obtiene. Es de suponer que esta asignación es para una organización con usuarios que navegan poco y requieren un buen desempeño al momento de solicitar un archivo. Este es un buen ejemplo de cómo se puede jugar con los parámetros del Delay Pool teniendo en cuenta las costumbres de navegación de la organización.

delay_parameters 1 76800/230400

Un Delay clase 1. Se usan máximo 600Kb/s de ancho de banda con ráfagas de descarga de 1800Kb.

Solo se limita el ancho de banda que usa en total sin importar cómo se distribuye el canal entre los clientes, lo cual da la posibilidad a condiciones de competencia por el ancho de banda en todo momento.

delay_parameters 1 340787/1022361 10000/200000

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

Aquí el peor caso se presenta cuando hay 34 clientes demandando todo el canal asignado de forma continua.

En los Delay Pools clase 2 y clase 3 es posible deshabilitar los buckets que no se desea utilizar

colocando -1/-1 en el bucket correspondiente. Por ejemplo:

asigno el canal a hosts en una red sin límite global

delay_parameters 1 -1/-1 10000/200000

asigno canal en una red a hosts individuales sin límites por subred

delay_parameters 2 340787/1022361 -1/-1 10000/200000

asigno canal en una red a cada su red sin importar los hosts

delay_parameters 3 340787/1022361 10000/200000 -1/-1

Delay_access

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

Sintaxis:

delay_access id allow acl name | deny acl name

Nivel del bucket al inicio

La directiva delay_initial_bucket_level, controla qué tan lleno estará el bucket al arranque del Squid o cuando se reconfigura. Esto afecta tanto a los individual buckets como a los aggregate buckets.

Note que los buckets no se crean sino hasta que son referenciados por primera vez. El valor es un porcentaje.

Sintaxis:

delay_initial_bucket_level porcentaje

porcentaje toma valores entre 0-100

Ejemplo:

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

Monitoreo y prueba

Puede monitorear los Delay Pools con el comando:

squidclient mgr:delay | less

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

Implementaciones

Primer caso

Los valores son para una organización con una red con cerca de 1000 usuarios de los cuales solo navegan en Internet unos 300 (esto se logra con autenticación en el Proxy). Los 300 usuarios hacen pocas peticiones al caché. No importa qué descarguen, todo pasa por el mismo Delay Pool.

delay_class 1 3

#600Kbits

delay_parameters 1 76800/240400 42800/100000 10000/70000

delay_initial_bucket_level 90

delay_access 1 allow all

Segundo caso

Una organización con una red con usuarios que son grandes consumidores de canal. Se hacen distinciones entre los tipos de archivo que se traen desde Internet. Por un Delay pasan unos tipos de archivo; por otro pasa la navegación en general y por otro pasa un "cliente especial". Como mínimo para esta organización se requieren 2.6Mb/s.

acl restringir url_regex -i "/etc/squid/extensions"

acl ClienteEspecial src 192.168.2.12/32

cantidad de delay pools

delay_pools 3

delay_class 1 2 # delay para extensiones restringidas

delay_class 2 2 # delay para navegacion en general

delay_class 3 1 # delay para un cliente especial

delay_initial_bucket_level 90

#2.6Mbits en esencia solo me importa aquí el individual bucket.

delay_parameters 1 340787/1022361 10000/200000

le doy un enlace de 128Kb = 16KB = 16384B a cada IP

delay_parameters 2 340787/1022361 16384/200000

delay para el cliente especial

delay_parameters 3 26384/300000

delay_access 3 allow ClienteEspecial !restringir

delay_access 3 deny all

delay_access 2 allow all !ClienteEspecial !restringir

delay_access 2 deny all

delay_access 1 allow restringir

delay_access 1 deny all

Limitaciones

No todo son maravillas con los Delay Pools. Estos poseen limitaciones que deben ser tomadas en consideración:

* No comparten ancho de banda. Esta es la principal limitación. Si existe un solo cliente demandando ancho banda de Internet la cantidad máxima de canal que recibe está limitado por el Delay más restrictivo que lo cobija sin importar que él sea el único haciendo peticiones al caché. Si el cliente tiene asignados 20KB/s y hay 200KB/s

sin usar, seguirá recibiendo los caché misses a máximo 20KB/s. Si su objetivo es optimizar el uso del canal, por este lado, no hay mucho por hacer.

* Si desea manejar unos valores de rate/size en horarios laborales y otros valores en horarios no laborales, en el momento de hacer los cambios puede provocar la interrupción de las descargas. El cambio de valores lo puede realizar con acl's de tiempo o con un reconfigure de Squid a la hora deseada haciéndolo leer un archivo de configuración diferente (dos o tres líneas de cron, dos archivos de configuración de Squid que solo cambian en los valores rate/size del Delay, un enlace simbólico y un script de bash hacen el trabajo).

Squid en modo NO transparente o en modo transparente

Explicado en el archivo que Servicios-Linux (del MEC) a partir de la página 180

Hacer enrutamiento, a modo de ejemplo:

Supongamos un router que tiene dos IP's. La 16.16.0.1 tiene la interfaz eth11, y la IP 192.168.5.3 asociada la interfaz eth10

a) iptables –t nat –I POSTROUTING –o eth10 –j MASQUERADE

b) iptables –t filter –I FORWARD –j ACCEPT

c) Echo "1" > /proc/sys/net/ipv4/ip_forward

Proxy No transparente (Visible)

1º) Cliente tiene que quitar la puerta de enlace y el DNS.

Y en el navegador debe poner la IP 16.16.0.1 y el puerto 1617.

2º) Servidor Proxy en modo NO transparente

a)Squid.conf -> http_port IP:1617

b) iptables –t nat –I POSTROUTING –o eth10 –j MASQUERADE

c) iptables –t filter –I FORWARD –j ACCEPT

Proxy Transparente (No Visible)

1º) Cliente tiene que poner la puerta de enlace y el DNS.

Y en el navegador debe quitar la IP 16.16.0.1 y el puerto 1617

2º) Servidor Proxy en modo transparente

a)Squid.conf -> http_port IP:1617 transparent

b) iptables –t nat –I PREROUTING –i eth11 –p tcp –dport 80 –j REDIRECT –to-port 1617

c) iptables –t nat –I POSTROUTING –o eth10 –j MASQUERADE

d) iptables –t filter –I FORWARD –j ACCEPT

e) instalación de un servidor DNS apt-get install bind9