

Introducción.

¿Qué es Servidor Intermediario (Proxy)?

El término en inglés «**Proxy**» tiene un significado muy general y al mismo tiempo ambiguo, aunque invariablemente se considera un sinónimo del concepto de «**Intermediario**». Se suele traducir, en el sentido estricto, como **delegado** o **apoderado** (el que tiene el que poder sobre otro).

Un **Servidor Intermediario** (Proxy) se define como una computadora o dispositivo que ofrece un servicio de red que consiste en permitir a los clientes realizar conexiones de red indirectas hacia otros servicios de red. Durante el proceso ocurre lo siguiente:

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

Los **Servidores Intermediarios** (Proxies) generalmente se hacen trabajar simultáneamente como muro cortafuegos operando en el **Nivel de Red**, actuando como filtro de paquetes, como en el caso de **iptables**, o bien operando en el **Nivel de Aplicación**, controlando diversos servicios, como es el caso de **TCP Wrapper**. Dependiendo del contexto, el muro cortafuegos también se conoce como **BPD** o **Border Protection Device** o simplemente **filtro de paquetes**.

Una aplicación común de los **Servidores Intermediarios** (Proxies) es funcionar como caché de contenido de Red (principalmente HTTP), proporcionando en la proximidad de los clientes un caché de páginas y ficheros disponibles a través de la Red en servidores HTTP remotos, permitiendo a los clientes de la red local acceder hacia éstos de forma más rápida y confiable.

Cuando se recibe una petición para un recurso de Red especificado en un **URL** (Uniform **R**esource **L**ocator) el **Servidor Intermediario** busca el resultado del **URL** dentro del caché. Si éste es encontrado, el **Servidor Intermediario** responde al cliente proporcionado inmediatamente el contenido solicitado. Si el contenido solicitado no estuviera disponible en el caché, el **Servidor Intermediario** lo traerá desde servidor remoto, entregándolo al cliente que lo solicitó y guardando una copia en el caché. El contenido en el caché es eliminado luego a través de un algoritmo de expiración de acuerdo a la antigüedad, tamaño e historial de **respuestas a solicitudes** (hits) (ejemplos: **LRU**, **LFUDA** y **GDSF**).

Los **Servidores Intermediarios** para contenido de Red (Web Proxies) también pueden actuar como filtros del contenido servido, aplicando políticas de censura de acuerdo a criterios arbitrarios.

Acerca de Squid.

Squid es un **Servidor Intermediario** (Proxy) de alto desempeño que se ha venido desarrollando desde hace varios años y es hoy en día un muy popular y ampliamente utilizado entre los sistemas operativos como GNU/Linux y derivados de Unix®. Es muy confiable, robusto y versátil y se distribuye bajo los términos de la Licencia Pública General GNU (**GNU/GPL**). Siendo sustento lógico **libre**, está disponible el código fuente para quien así lo requiera.

Entre otras cosas, **Squid** puede funcionar como **Servidor Intermediario** (Proxy) y **caché de contenido de Red** para 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 consiste de un programa principal como servidor, un programa para búsqueda en servidores **DNS**, programas opcionales para reescribir solicitudes y realizar autenticación y algunas herramientas para administración y y herramientas para clientes. Al iniciar **Squid** da origen a un número configurable (5, de modo predefinido a través del parámetro **dns_children**) de procesos de búsqueda en servidores **DNS**, cada uno de los cuales realiza una búsqueda única en servidores **DNS**, reduciendo la cantidad de tiempo de espera para las búsquedas en servidores **DNS**.

NOTA ESPECIAL: Squid no debe ser utilizado como Servidor Intermediario (Proxy) para protocolos como **SMTP, POP3, TELNET, SSH, IRC**, etc. Si se requiere intermediar para **cualquier protocolo distinto a HTTP, HTTPS, FTP, GOPHER** y **WAIS** se requerirá implementar obligatoriamente un enmascaramiento de IP o **NAT** (Network Address Translation) o bien hacer uso de un servidor **SOCKS** como **Dante** (<http://www.inet.no/dante/>).

URL: <http://www.squid-cache.org/>

Algoritmos de caché utilizados por Squid.

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.**

- **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). Tiene una eficiencia por **octetos** (Bytes) menor que el algoritmo **LFUDA** debido a que descarta del caché objetos grandes que sean solicitado con frecuencia.

Sustento lógico necesario.

Para poder llevar al cabo los procedimientos descritos en este manual y documentos relacionados, usted necesitará tener instalado al menos lo siguiente:

- Al menos squid-2.5.STABLE6
- httpd-2.0.x (Apache), como auxiliar de caché con aceleración.
- **Todos** los parches de seguridad disponibles para la versión del sistema operativo que esté utilizando. No es conveniente utilizar un sistema con posibles vulnerabilidades como Servidor Intermediario.

Debe tomarse en consideración que, de ser posible, se debe utilizar **siempre** las versiones estables más recientes de todo sustento lógico que vaya a ser instalado para realizar los procedimientos descritos en este manual, a fin de contar con los parches de seguridad necesarios. **Ninguna versión de Squid anterior a la 2.5.STABLE6 se considera como apropiada** debido a fallas de seguridad de gran importancia.

Squid no se instala de manera predeterminada a menos que especifique lo contrario durante la instalación del sistema operativo, sin embargo viene incluido en casi todas las distribuciones actuales. El procedimiento de instalación es exactamente el mismo que con cualquier otro sustento lógico.

Instalación a través de yum.

Si cuenta con un sistema con CentOS o White Box Enterprise Linux 3 o versiones posteriores, utilice lo siguiente y se instalará todo lo necesario junto con sus dependencias:

```
yum -y install squid httpd
```

Instalación a través de up2date.

Si cuenta con un sistema con Red Hat™ Enterprise Linux 3 o versiones posteriores, utilice lo siguiente y se instalará todo lo necesario junto con sus dependencias:

```
up2date -i squid httpd
```

Otros componentes necesarios.

El mandato **iptables** se utilizará para generar las reglas necesarias para el guión de Enmascaramiento de IP. Se instala de modo predefinido en todas las distribuciones actuales que utilicen **núcleo** (kernel) versiones 2.4 y 2.6.

Es importante tener actualizado el núcleo del sistema operativo por diversas cuestiones de seguridad. No es recomendable utilizar versiones del kernel anteriores a la **2.4.21**. Actualice el núcleo a la versión más reciente disponible para su distribución.

Si cuenta con un sistema con CentOS o White Box Enterprise Linux 3 o versiones posteriores, utilice lo siguiente para actualizar el núcleo del sistema operativo e **iptables**, si acaso fuera necesario:

```
yum -y update kernel iptables
```

Si cuenta con un sistema con Red Hat™ Enterprise Linux 3 o versiones posteriores, utilice lo siguiente para actualizar el núcleo del sistema operativo, e **iptables** si acaso fuera necesario:

```
up2date -u kernel iptables
```

Antes de continuar.

Tenga en cuenta que este manual ha sido comprobado varias veces y ha funcionado en todos los casos y si algo no funciona solo significa que usted no lo leyó a detalle y no siguió correctamente las indicaciones.

Evite dejar **espacios vacíos** en lugares indebidos. El siguiente es un ejemplo de como **no** se debe habilitar un parámetro.

Mal

```
# Opción incorrectamente habilitada  
http_port 3128
```

El siguiente es un ejemplo de como **si** se debe habilitar un parámetro.

Bien

```
# Opción correctamente habilitada  
http_port 3128
```

Configuración básica.

Squid utiliza el fichero de configuración localizado en `/etc/squid/squid.conf`, y podrá trabajar sobre este utilizando su editor de texto simple preferido. Existen un gran número de parámetros, de los cuales recomendamos configurar los siguientes:

- `http_port`
- `cache_dir`
- Al menos una **Lista de Control de Acceso**
- Al menos una **Regla de Control de Acceso**
- `httpd_accel_host`
- `httpd_accel_port`
- `httpd_accel_with_proxy`

Parámetro `http_port`: ¿Que puerto utilizar para Squid?

De acuerdo a las asignaciones hechas por **IANA** y continuadas por la **ICANN** desde el 21 de marzo de 2001, los **Puertos Registrados** (rango desde 1024 hasta 49151) recomendados para **Servidores Intermediarios** (Proxies) pueden ser el 3128 y 8080 a través de **TCP**.

De modo predefinido **Squid** utilizará el puerto 3128 para atender peticiones, sin embargo se puede especificar que lo haga en cualquier otro puerto disponible o bien que lo haga en varios puertos disponibles a la vez.

En el caso de un **Servidor Intermediario** (Proxy) Transparente, regularmente se utilizará el puerto 80 o el 8000 y se valdrá del re-direccionamiento de peticiones de modo tal que no habrá necesidad alguna de modificar la configuración de los **clientes HTTP** para utilizar el **Servidor Intermediario** (Proxy). Bastará con utilizar como puerta de enlace al servidor. Es importante recordar que los **Servidores HTTP**, como Apache, también utilizan dicho puerto, por lo que será necesario volver a configurar el servidor **HTTP** para utilizar otro puerto disponible, o bien desinstalar o desactivar el servidor **HTTP**.

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

Regularmente algunos programas utilizados comúnmente por los usuarios suelen traer de modo predefinido el puerto 8080 (**servicio de cacheo WWW**) para utilizarse al configurar que **Servidor Intermediario** (Proxy) utilizar. Si queremos aprovechar esto en nuestro favor y ahorrarnos el tener que dar explicaciones innecesarias al usuario, podemos especificar que **Squid** escuche peticiones en dicho puerto también. Siendo así localice la sección de definición de `http_port`, y especifique:

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

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

```
#
#   You may specify multiple socket addresses on multiple
lines.
#
# Default: http_port 3128
http_port 192.168.1.254:3128
http_port 192.168.1.254:8080
```

Parámetro `cache_mem`.

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

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

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

De modo predefinido se establecen 8 MB. Puede especificarse una cantidad mayor si así se considera necesario, dependiendo esto de los hábitos de los usuarios o necesidades establecidas por el administrador.

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

```
cache_mem 16 MB
```

Parámetro `cache_dir`: ¿Cuanto desea almacenar de Internet en el disco duro?

Este parámetro se utiliza para establecer que tamaño se desea que tenga el caché en el disco duro para **Squid**. Para entender esto un poco mejor, responda a esta pregunta:

¿Cuanto desea almacenar de Internet en el disco duro? De modo predefinido **Squid** utilizará un caché de 100 MB, de modo tal que encontrará la siguiente línea:

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

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

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

Los números **16** y **256** significan que el directorio del caché contendrá 16 directorios subordinados con 256 niveles cada uno. **No modifique esto números, no hay necesidad de hacerlo.**

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

Parámetro ftp_user.

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

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

Controles de acceso.

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

Listas de control de acceso.

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

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

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

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

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

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

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

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

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

Reglas de Control de Acceso.

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

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

La sintaxis básica es la siguiente:

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

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

```
http_access allow permitidos
```

También pueden definirse reglas valiéndose de la expresión **!**, la cual significa **no**. Pueden definirse, por ejemplo, dos listas de control de acceso, una denominada **lista1** y otra denominada **lista2**, en la misma regla de control de acceso, en donde se asigna una expresión a una de estas. La siguiente establece que se permite el acceso a **Squid** a lo que comprenda **lista1** excepto aquello que comprenda **lista2**:

```
http_access allow lista1 !lista2
```

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

Aplicando Listas y Reglas de control de acceso.

Una vez comprendido el funcionamiento de la Listas y las Regla de Control de Acceso, procederemos a determinar cuales utilizar para nuestra configuración.

Caso 1.

Considerando como ejemplo que se dispone de una red 192.168.1.0/255.255.255.0, si se desea definir toda la red local, utilizaremos la siguiente línea en la sección de **Listas de Control de Acceso**:

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

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

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

```
#  
# Recommended minimum configuration:  
acl all src 0.0.0.0/0.0.0.0  
acl manager proto cache_object  
acl localhost src 127.0.0.1/255.255.255.255  
acl todalared src 192.168.1.0/255.255.255.0
```

A continuación procedemos a aplicar la regla de control de acceso:

```
http_access allow todalared
```

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

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

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

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

Caso 2.

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

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

Denominaremos a esta lista de control de acceso como **redlocal**:

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

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

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

```
#
# Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl redlocal src "/etc/squid/listas/redlocal"
```

A continuación procedemos a aplicar la regla de control de acceso:

```
http_access allow redlocal
```

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

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

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

La regla **http_access allow redlocal** permite el acceso a **Squid** a la **Lista de Control de Acceso** denominada **redlocal**, la cual está conformada por las direcciones IP especificadas en el fichero **/etc/squid/listas/redlocal**. Esto significa que cualquier máquina no incluida en **/etc/squid/listas/redlocal** no tendrá acceso a **Squid**.

Parámetro `cache_mgr`.

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

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

Parámetro `cache_peer`: caches padres y hermanos.

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

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

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

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

Cuando se trabaja en redes muy grandes donde existen varios Servidores Intermediarios (Proxy) haciendo caché de contenido de Internet, es una buena idea hacer trabajar todos los caché entre si. Configurar caches vecinos como **sibling** (hermanos) tiene como beneficio el que se consultarán estos caches localizados en la red local antes de acceder hacia Internet y consumir ancho de banda para acceder hacia un objeto que ya podría estar presente en otro caché vecino.

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

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

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

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

Caché con aceleración.

Cuando un usuario hace petición hacia un objeto en Internet, este es almacenado en el caché de **Squid**. Si otro usuario hace petición hacia el mismo objeto, y este no ha sufrido modificación alguna desde que lo accedió el usuario anterior, **Squid** mostrará el que ya se encuentra en el caché en lugar de volver a descargarlo desde Internet.

Esta función permite navegar rápidamente cuando los objetos ya están en el caché de **Squid** y además optimiza enormemente la utilización del ancho de banda.

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

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

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

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

Proxy Acelerado: Opciones para Servidor Intermediario (Proxy) Transparente.

Si se trata de un **Servidor Intermediario** (Proxy) transparente, deben utilizarse las siguientes opciones, considerando que se hará uso del caché de un servidor **HTTP** (Apache) como auxiliar:

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

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

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

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

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

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

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

Elimine primero el enlace simbólico actual:

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

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

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

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

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

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

```
service squid start
```

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

```
service squid restart
```

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

```
chkconfig squid on
```

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

Depuración de errores

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

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

```
service squid reload
```

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

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

Ajustes para el muro corta-fuegos.

Si se tiene poca experiencia con guiones de cortafuegos a través de iptables, sugerimos utilizar **Firestarter**. éste permite configurar fácilmente tanto el enmascaramiento de IP como el muro corta-fuegos. Si se tiene un poco más de experiencia, recomendamos utilizar **Shorewall** para el mismo fin puesto que se trata de una herramienta más robusta y completa.

- **Firestarter**: <http://www.fs-security.com/>
- **Shorewall**: <http://www.shorewall.net/>

Re-direccionamiento de peticiones a través de iptables y Firestarter.

En un momento dado se requerirá tener salida transparente hacia Internet para ciertos servicios, pero al mismo tiempo se necesitará re-direccionar peticiones hacia servicio **HTTP** para pasar a través del el puerto donde escucha peticiones **Squid** (8080), de modo que no haya salida alguna hacia alguna hacia servidores **HTTP** en el exterior sin que ésta pase antes por **Squid**. No se puede hacer **Servidor Intermediario** (Proxy) Transparente para los protocolos **HTTPS**, **FTP**, **GOPHER** ni **WAIS**, por lo que dichos protocolos tendrán que ser filtrados a través del **NAT**.

El re-direccionamiento lo hacemos a través de **iptables**. Considerando para este ejemplo que la red local se accede a través de una interfaz eth0, el siguiente esquema ejemplifica un re-direccionamiento:

```
/sbin/iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Lo anterior, **que requiere un guión de cortafuegos funcional en un sistema con dos interfaces de red**, hace que cualquier petición hacia el puerto 80 (servicio HTTP) hecha desde la red local hacia el exterior, se re-direccionará hacia el puerto 8080 del servidor.

Utilizando **Firestarter**, la regla anteriormente descrita se añade en el fichero **/etc/firestarter/user-post**.

Re-direccionamiento de peticiones a través de la opción REDIRECT en Shorewall.

La acción **REDIRECT** en **Shorewall** permite redirigir peticiones hacia protocolo **HTTP** para hacerlas pasar a través de **Squid**. En el siguiente ejemplo las peticiones hechas desde la zona que corresponde a la red local serán redirigidas hacia el puerto 8080 del cortafuegos, en donde está configurado **Squid** configurado como **Servidores Intermediario** (Proxy) transparente.

#ACTION	SOURCE	DEST	PROTO	DEST
REDIRECT	loc	8080	tcp	80
