Servidor proxy y de caché Squid

Squid (que significa «calamar») es un programa open source, creado en 1996, que implementa un servidor proxy y proxy inverso (*reverse proxy*), para los protocolos FTP, Gopher, HTTP y HTTPS.

En el marco de la certificación LPIC-2, vamos a estudiar sus funciones proxy HTTP/HTTPS.

1. Roles de los servidores proxy

Un servidor proxy o proxy inverso es un intermediario entre el cliente y el servidor de un protocolo de software:

- Un servidor proxy recibe la solicitud del cliente, la envía al servidor como si él fuera el emisor de esta, recupera la respuesta y la envía al cliente. El servidor no conoce al cliente que ha iniciado la conexión.
- Un proxy inverso (*reverse proxy*) recibe la solicitud del cliente en lugar del servidor final, la transmite al servidor final, recupera la respuesta y la transmite al cliente. El cliente no conoce, por lo tanto, al servidor destino de la conexión.

Existen programas proxy para la mayoría de los protocolos de software de la familia TCP/IP (FTP, Gopher, HTTP, etc.). Los más usados son los proxys y proxys inversos HTTP.

El rol de proxy inverso se verá en al tema siguiente, Nginx.

a. Protección de los clientes

En muchas organizaciones, los puestos de trabajo y los servidores internos no se encuentran directamente accesibles desde Internet. Los sistemas que deben estar accesibles desde el interior (intranet) y desde el exterior de la organización (Internet o extranet) están situados en una estructura de red particular llamada DMZ (*DeMilitarized*

Zone), protegida por routers y firewalls. El proxy HTTP está instalado en esta zona y hace de intermediario entre los clientes HTTP y los servidores HTTP externos. Estos últimos no están directamente en contacto con los clientes, comunican exclusivamente con el servidor proxy, lo que limita los riesgos de seguridad.

b. Servidores de caché

El proxy trata las solicitudes de todos los clientes y las respuestas de los servidores. Puede almacenar en su caché elementos que transitan entre los dos (archivos, resoluciones de nombres y de direcciones, etc.). Si un cliente solicita una página HTML, solicitada anteriormente por otro cliente, el proxy proporcionará los elementos estáticos presentes en su caché (si siguen siendo válidos), lo que limitará el tráfico de red y optimizará el tiempo de respuesta.

c. Filtrado y registros

El servidor proxy es un punto de paso obligatorio entre los clientes http y los servidores. Podemos configurarlo para que guarde en un registro todas las peticiones recibidas y para aplicar filtros en las solicitudes y respuestas.

Una organización puede usar el servidor proxy HTTP para prohibir algunos sitios, de manera global o para algunos clientes. También puede usar datos de los registros para establecer un seguimiento más o menos detallado de la actividad de los clientes HTTP.

d. Límites

Un servidor proxy HTTP debe ser conocidos por todos los clientes de la organización. Estos deben estar configurados específicamente. Incluso si la mayoría de los navegadores de Internet pueden ser configurados para detectar que se encuentran detrás de un proxy, hay que activarlo en cada puesto cliente.

Ya que el proxy es un punto de paso obligatorio, puede bloquear todo el acceso a una red o Internet en el caso de que haya un fallo o disfunción. Para limitar este inconveniente, podemos crear grupos de servidores proxy para asegurar la tolerancia de fallos.

2. Configuración básica del servidor proxy Squid

El servidor Squid está incluido en el paquete de software `squid`. El arranque y el paro del daemon `squid` se puede gestionar con un script `init System V`, `/etc/init.d/squid`, o usando `systemd`.

Ejemplo

Instalación del paquete `squid` en una distribución CentOS 8:

```
yum list squid
CentOS-8 - AppStream                            1.3 kB/s | 4.3 kB     00:03
CentOS-8 - Base                                 156 kB/s | 3.9 kB     00:00
CentOS-8 - Extras                               57 kB/s | 1.5 kB     00:00
Extra Packages for Enterprise Linux Modular 8 - x86_64  48 kB/s |  34 kB     00:00
Extra Packages for Enterprise Linux 8 - x86_64  89 kB/s |  34 kB     00:00
Extra Packages for Enterprise Linux 8 - x86_64  8.6 MB/s | 6.6 MB     00:00
Paquetes disponibles
squid.x86_64          7:4.4-8.module_el8.1.0+197+0c39cdc8          AppStream
```

```
yum install squid
Última comprobación de caducidad de metadatos hecha hace 0:04:21, el mar 19 mayo 2020 09:07:30 CEST.
Dependencias resueltas.
================================================================================
================================================
Paquete                Arquitectura      Versión
            Repositorio               Tam.
================================================================================
================================================
Instalando:
squid                  x86_64            7:4.4-8.module_el8.1.0+197+0c39cdc8
            AppStream             3.6 M
Instalando dependencias:
libecap                x86_64            1.0.1-2.module_el8.1.0+197+0c39cdc8
        AppStream             29 k
perl-Digest-SHA        x86_64            1:6.02-1.el8
            AppStream             66 k
Activando flujos de módulos:
squid                                    4
```

```
Resumen de la transacción
================================================================================
Instalar 4 Paquetes

Tamaño total de la descarga: 3.7 M
Tamaño instalado: 15 M
¿Está de acuerdo [s/N]?:S
[...]
Instalado:
 squid-7:4.4-8.módulo_el8.1.0+197+0c39cdc8.x86_64
 libecap-1.0.1-2.módulo_el8.1.0+197+0c39cdc8.x86_64
 perl-Digest-SHA-1:6.02-1.el8.x86_64

!Listo!
```

Mostramos el estado del servicio `squid`, usando `systemd`:

```
systemctl status squid
  squid.service - Squid caching proxy
  Loaded: loaded (/usr/lib/systemd/system/squid.service; disabled;
vendor preset: disabled)
  Active: inactive (dead)
    Docs: man:squid(8)
```

El servidor Squid no está activo por defecto.

a. El archivo de configuración squid.conf

El directorio de configuración del servidor Squid es `/etc/squid`. En este directorio se encuentra su archivo de configuración por defecto, `squid.conf`.

Se trata de un archivo de texto, autodocumentado, pudiendo incluir otros archivos, en particular `/etc/squid/conf.d/debian.conf` para las distribuciones de tipo Debian.

La configuración por defecto es relativamente simple e implementa las funciones de proxy y caché.

El puerto usado por el daemon está definido por la directiva `http_port`. Por defecto, se trata del puerto 3128. Históricamente, el puerto usado por un proxy HTTP es el 8080.

Los archivos de registro por defecto se encuentran en el directorio `/var/log/squid: access.log` y `cache.log`.

Ejemplos

Directivas de un archivo por defecto en una distribución CentOS 8.

Sin los comentarios:

```
acl localnet src 0.0.0.1-0.255.255.255  # RFC 1122 "this" network (LAN)
acl localnet src 10.0.0.0/8             # RFC 1918 local private network (LAN)
acl localnet src 100.64.0.0/10          # RFC 6598 shared address space (CGN)
acl localnet src 169.254.0.0/16         # RFC 3927 link-local (directly plugged) machines
acl localnet src 172.16.0.0/12          # RFC 1918 local private network (LAN)
acl localnet src 192.168.1.0/16         # RFC 1918 local private network (LAN)
acl localnet src fc00::/7               # RFC 4193 local private network range
acl localnet src fe80::/10              # RFC 4291 link-local (directly plugged) machines
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
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         # filemake
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost manager
http_access deny manager
http_access allow localnet
```

```
http_access allow localhost
http_access deny all
http_port 3128
coredump_dir /var/spool/squid
refresh_pattern ^ftp:          1440    20%     10080
refresh_pattern ^gopher:       1440    0%      1440
refresh_pattern -i (/cgi-bin/|\?) 0     0%      0
refresh_pattern .              0       20%     4320
```

Estas directivas definen las listas de control de acceso (directivas `acl`) que se utilizan para autorizar o no las solicitudes desde las direcciones IP y hacia los números de puerto (directivas `http_access allow` o `deny`).

Las directivas `refresh_pattern` definen los parámetros de gestión de los datos en caché para diferentes tipos de URL.

Con esta configuración, solamente se autorizan las solicitudes emitidas desde redes internas, y solamente hacia una lista de puertos (80 HTTP, 443 HTTPS, 21 FTP, etc.).

El puerto usado por el daemon está definido en la directiva `http_port 3128`.

Directivas de un archivo por defecto en una distribución Debian 10.

Sin los comentarios:

```
acl localnet src 0.0.0.1-0.255.255.255  # RFC 1122 "this" network (LAN)
acl localnet src 10.0.0.0/8             # RFC 1918 local private network (LAN)
acl localnet src 100.64.0.0/10          # RFC 6598 shared address space (CGN)
acl localnet src 169.254.0.0/16         # RFC 3927 link-local (directly plugged) machines
acl localnet src 172.16.0.0/12          # RFC 1918 local private network (LAN)
acl localnet src 192.168.1.0/16         # RFC 1918 local private network (LAN)
acl localnet src fc00::/7               # RFC 4193 local private network range
acl localnet src fe80::/10              # RFC 4291 link-local (directly plugged) machines
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
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 deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost manager
http_access deny manager
include /etc/squid/conf.d/*
http_access allow localhost
http_access deny all
http_port 3128
coredump_dir /var/spool/squid
refresh_pattern ^ftp:          1440    20%     10080
refresh_pattern ^gopher:       1440    0%      1440
refresh_pattern -i (/cgi-bin/|\?) 0     0%      0
refresh_pattern .              0       20%     4320
```

El directorio complementario de inclusión `/etc/squid/conf.d/` *solamente contiene el archivo* `debian.conf` :

```
logfile_rotate 0
```

Con esta configuración, solamente se autorizan las solicitudes emitidas desde redes internas:

```
http_access allow localhost
http_access deny all
```

y únicamente hacia una lista de puertos (80 HTTP, 443 HTTPS, 21 FTP, etc.).

El puerto usado por el daemon está definido por la directiva `http_port 3128` .

b. Gestión del caché

Por defecto, el caché del servidor Squid se encuentra exclusivamente en memoria. El tiempo de acceso es, por lo tanto, muy rápido, pero el contenido del caché se perderá en el caso de que el servicio se pare de manera abrupta.

Existen diferentes directivas que permiten configurar un caché almacenado en el disco. La principal es `cache_dir`.

Sintaxis

```
cache_dir ufs CaminoDir NúmMB NúmDir1 NúmDir2
```

Donde:

`ufs`	Tipo de gestión de almacenamiento. `ufs` es el tipo por defecto.
`CaminoDir`	Directorio raíz de la caché. Tiene que existir y Squid debe poder escribir en él.
`NœmMB`	Tamaño en megabytes de la caché útil. El tamaño indicado no debe superar el 80 % del espacio de disco disponible. 100 MB por defecto.
`NœmDir1`	Número de directorios de trabajo de primer nivel. 16 por defecto.
`NœmDir2`	Número de directorios de trabajo de segundo nivel. 256 por defecto.

Las directivas `refresh_pattern` permiten especificar las reglas de conservación de los objetos en la caché. Estas definen, en función de expresiones regulares que se aplicarán a la URL, el período de validez de los diferentes tipos de URL en la caché.

c. Listas de control de acceso

El control de acceso permite definir cuáles son los clientes autorizados y hacia qué direcciones y números de puertos.

La directiva `acl` permite crear conjuntos de direcciones y de puertos (ACL, *Access Control List*), que serán utilizados para definir las autorizaciones y las prohibiciones.

Sintaxis

```
acl Nombre TipoLista Direcciones|NúmPuerto
```

Con:

`Nombre`	Nombre de la lista en la que se añadirá el elemento.
`TipoLista`	Tipo de lista de control de acceso: `src` (emisores), `dst` (destinatarios), `port`, `dstdomain` (nombres DNS de los hosts o de los dominios), `url_regex` (expresión regular aplicada a la URL).
`Direcciones`	Dirección(es) que se añadirán a la lista. Se puede tratar de una dirección simple, de un identificador de red en notación CIDR, o de un intervalo entre dos direcciones (`dir1-dir2`). Podemos usar nombres de ACL predefinidas: `all`, `manager` (gestor del caché), `localhost` o `to_localhost`.
`NœmPuerto`	Número de puerto que se añadirá a la lista.

También se pueden definir elementos de la lista en un archivo (una línea por cada elemento), usando la sintaxis:

```
acl Nombre TipoLista "CaminoArchivo"
```

Las comillas son obligatorias.

Ejemplo

La lista de control de acceso de `Safe_ports`, definida en el archivo de configuración por defecto, contiene los números de puertos autorizados:

```
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
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
```

d. Directivas de control de acceso: http_access

Una vez que las listas de control de acceso estén definidas, podemos usarlas en las directivas que autorizan o deniegan los accesos.

Sintaxis

```
http_access Tipo [!]NombreAcl1 [ ... NombreAclN ]
```

Donde:

`Tipo`	Tipo de control: autorizado (`allow`) o denegado (`deny`).
`!`	El control se aplica a todos los elementos que no están en las ACL especificadas.
`NombreAcl1[... NombreAclN]`	ACL implicada(s) en el control. Podemos usar las ACL predefinidas `all`, `manager`, `localhost` o `to_localhost`.

En ausencia de directiva `http_access`, todos los accesos se deniegan.

Ejemplo

Estas dos directivas combinadas solamente autorizan el acceso desde la máquina local:

```
http_access allow localhost
http_access deny all
```

El orden de las directivas `http_access` es importante: el servidor las aplica en el orden de aparición en el archivo. En cuanto una de las directivas encaja con una solicitud en curso, el control de acceso se aplica y se termina el recorrido de las directivas. Por lo tanto, si una directiva autoriza una solicitud antes de otra que pueda denegarla, esta última no se podría aplicar.

Se aconseja poner siempre en la última línea del control de acceso la directiva `http_access deny all`, que deniega todo lo que no ha sido explícitamente autorizado anteriormente.

3. Ejemplo de servidor Squid

Este ejemplo muestra la configuración de un servidor Squid en una distribución CentOS 8, y su uso desde un navegador de Internet en una máquina de la red interna.

a. Configuración e inicio del servidor

Utilizamos el archivo de configuración por defecto del servidor Squid, modificándolo:

- Puerto 8080.
- Acceso autorizado solamente a los hosts de la red IPv4 `192.168.1.0/24` y a la máquina local.
- Puertos de servidores autorizados: HTTP (80), HTTPS (443) y los puertos dinámicos (1025-65535).
- Caché solamente en memoria.

vi /etc/squid/squid.conf

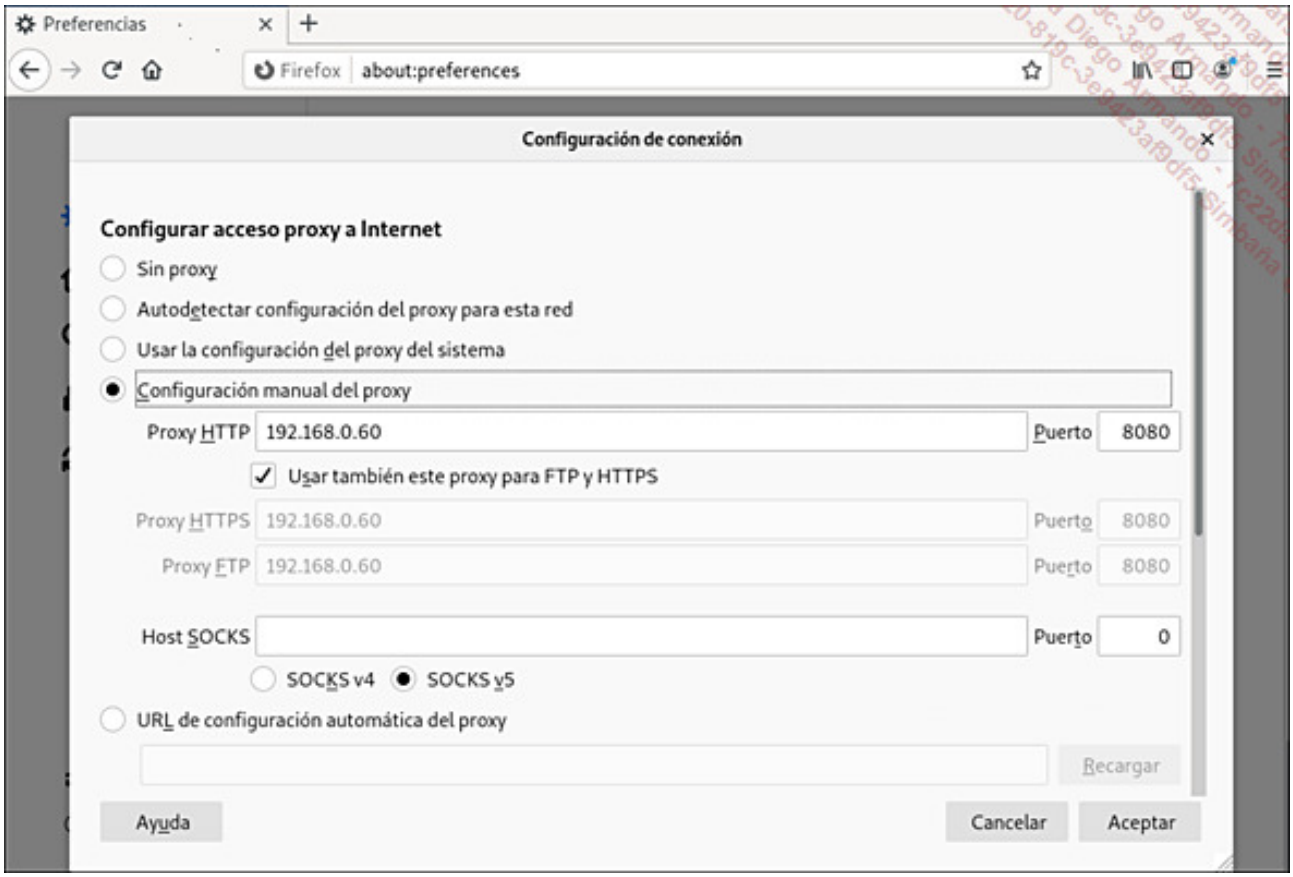
```
acl localnet src 192.168.1.0/24
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 443         # https
acl Safe_ports port 1025-65535  # unregistered ports
acl CONNECT method CONNECT
http_access allow Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access deny all
http_port 8080
coredump_dir /var/spool/squid
refresh_pattern ^ftp:           1440    20%     10080
refresh_pattern ^gopher:        1440    0%      1440
refresh_pattern -i (/cgi-bin/|\?) 0     0%      0
refresh_pattern .               0       20%     4320
```

Se inicia el servidor:

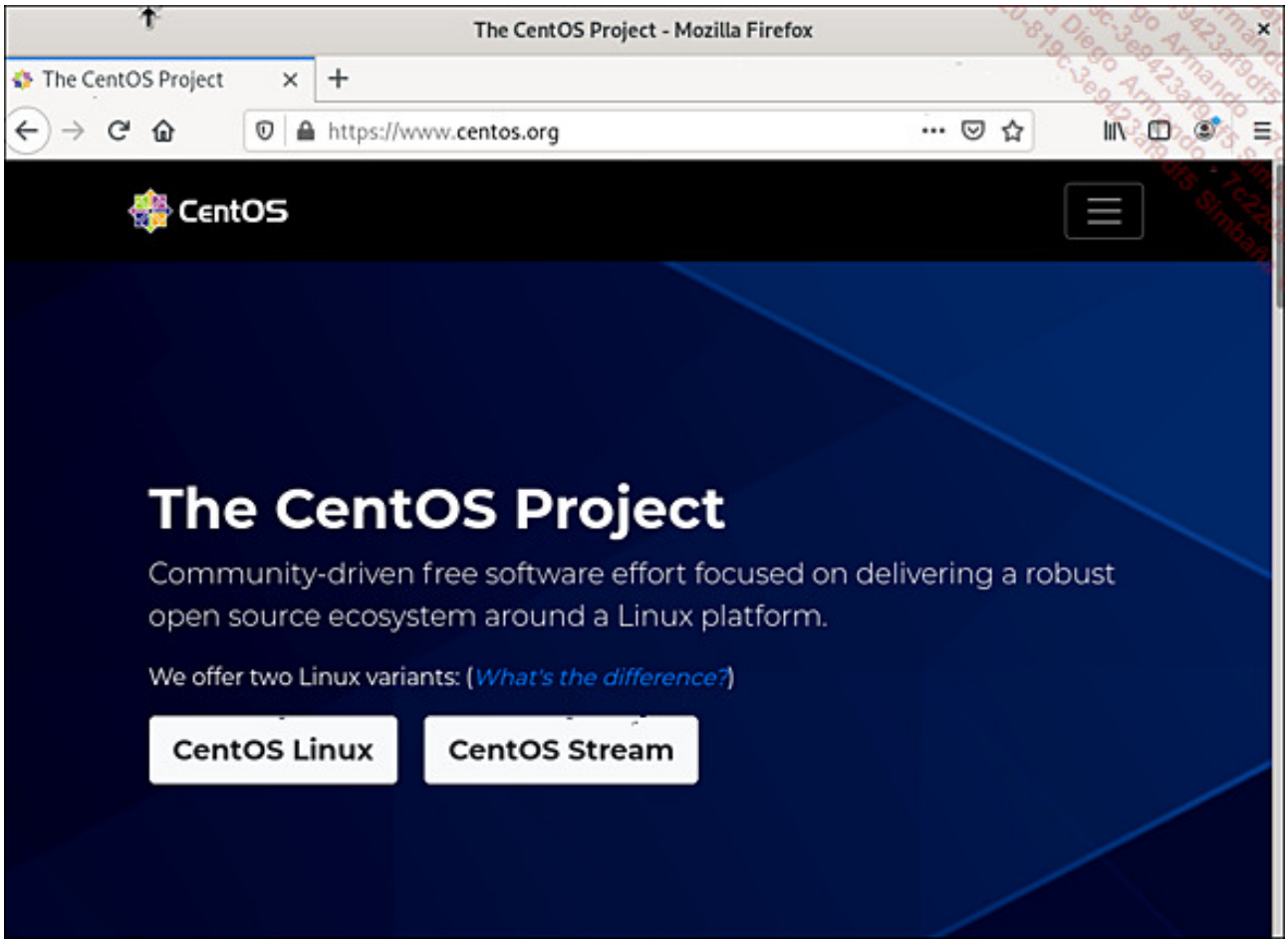
```
systemctl start squid
ps -ef | grep squid
root     9706    1  0 11:26?       00:00:00 /usr/sbin/squid -f /etc/squid/squid.conf
squid    9708  9706  0 11:26?       00:00:00 (squid-1) --kid squid-1 -f /etc/squid/squid.conf
squid    9709  9708  0 11:26?       00:00:00 (logfile-daemon) /var/log/squid/access.log
root     9711  7711  0 11:26 pts/0   00:00:00 grep --color=auto squid
```

b. Configuración y comprobación del cliente

El navegador del cliente tiene que estar configurado para usar el servidor proxy con su número de puerto. Por ejemplo, configuración manual del proxy usando Firefox:



A continuación podemos usar normalmente el navegador:



En el archivo de registro de los accesos en el servidor, `/var/log/squid/access.log` , vemos la petición de conexión del cliente:

```
1589881206.405     36 192.168.1.5 TCP_TUNNEL/200 39 CONNECT
www.centos.org:443 - HIER_DIRECT/2001:4de0:aaae::202 -
```

La dirección del cliente es `192.168.1.5` , y ha solicitado una conexión HTTPS en el puerto 443.

c. Configuración y comprobación de un cliente denegado

En el servidor Squid, modificamos la configuración para denegar el acceso al cliente que tenga la dirección `192.168.1.5` :

vi /etc/squid/squid.conf

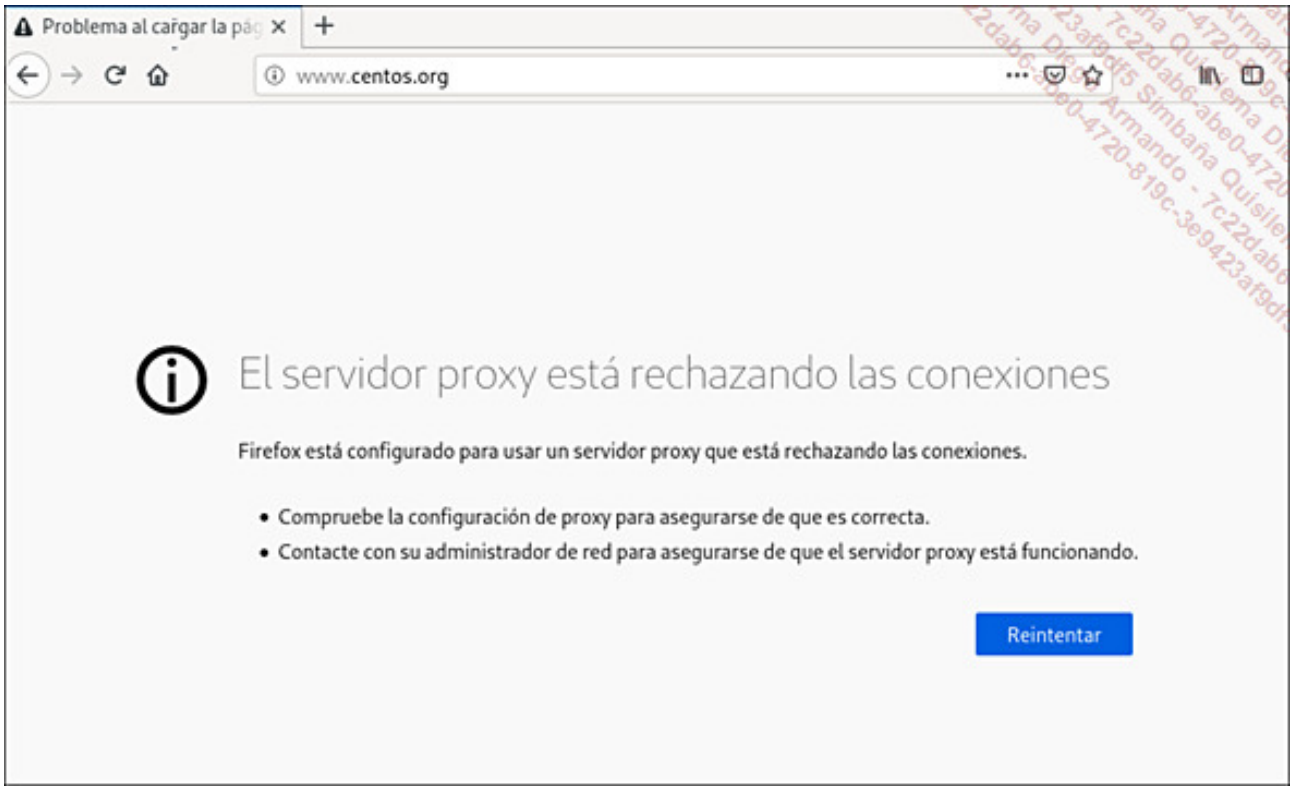
```
acl localnet src 192.168.1.0/24
acl cli_non src 192.168.1.5
acl SSL_ports port 443
acl Safe_ports port 80        # http
acl Safe_ports port 443       # https
acl Safe_ports port 1025-65535  # unregistered ports
http_access deny cli_non
acl CONNECT method CONNECT
http_access allow Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access deny all
http_port 8080
[...]
```

El orden de las directivas `http_access` es importante: el servidor las aplica en el orden del archivo. La directiva de denegación del cliente tiene que estar colocada antes de las directivas de autorización.

Recargamos la configuración del servidor:

systemctl reload squid

En el cliente, abrimos el navegador y solicitamos una URL. La solicitud es denegada por el proxy:



La solicitud denegada se registra en el archivo de registro del servidor Squid:

```
vi /var/log/squid/access.log
1589882500.612      0 192.168.1.5 TCP_DENIED/403 3960 CONNECT
www.centos.org:443 - HIER_NONE/- text/html
```

d. Configuración y comprobación de una URL denegada

En el servidor Squid, modificamos la configuración para autorizar a los clientes de la red local, pero denegar las URL que contengan la cadena de caracteres `facebook.com` :

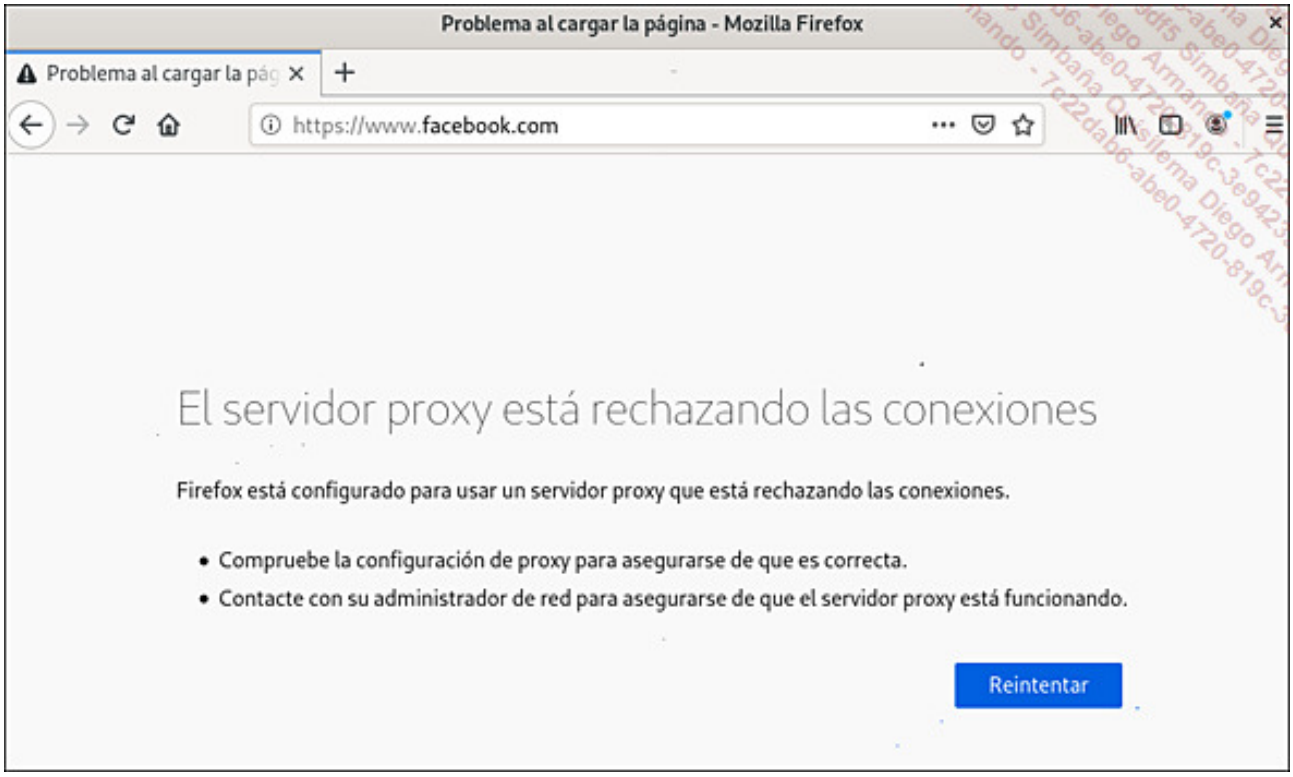
```
vi /etc/squid/squid.conf
acl localnet src 192.168.1.0/24
acl srv_non url_regex facebook.com
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 443         # https
acl Safe_ports port 1025-65535  # unregistered ports
```

```
http_access deny srv_non
acl CONNECT method CONNECT
http_access allow Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access deny all
http_port 8080
[...]
```

Recargamos la configuración del servidor:

```
systemctl reload squid
```

En el cliente abrimos el navegador y solicitamos una URL que contenga la cadena de caracteres `facebook.com`. La solicitud ha sido denegada por el proxy:



La solicitud ha sido denegada y registrada en el archivo de registro del servidor Squid:

vi /var/log/squid/access.log

1589887184.073 0 192.168.1.5 TCP_DENIED/403 3966 CONNECT www.facebook.com:443 - HIER_NONE/- text/html