[Cod.: CC481 Curso: Administración de Redes]
[Tema: Servidor Proxy]
[Prof.: Manuel Castillo]

Laboratorio dirigido 9

Servidor Proxy

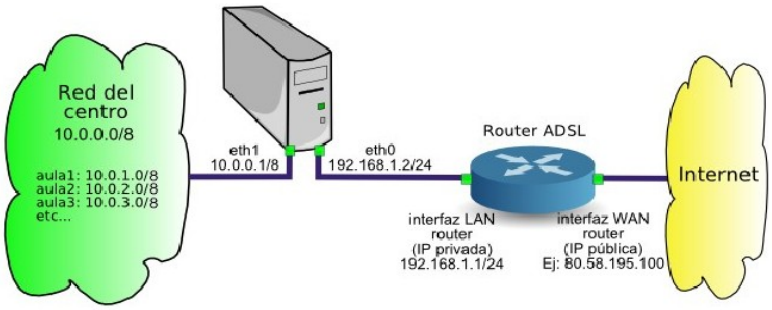
Instrucciones:

1. Fecha de entrega será antes del **domingo a las 23:59**.
2. El formato de entrega será *pdf*.
3. El laboratorio tendrá una puntuación sobre 20.
4. Para los laboratorios de servicios se deberá realizar capturas de cada paso que se están llevando a cabo. Además de realizar un **videotutorial** al final que muestre la correcta implementación de nuestro servicio.
5. La primera hoja será para la portada que se especificará, el número de laboratorio, nombre y apellidos de los integrantes, nombre asignatura y el escudo de la UNI.
6. La segunda página será el índice. Donde se deberá tener en cuenta la página de cada Actividad.
7. Las citas y extracciones realizadas de Internet se deberán especificar. No se corregirá el laboratorio en caso de copiar y pegar fragmentos sin especificar.
8. Utilizar letra clara y adecuada a un documento técnico con tamaño 12 y márgenes superior e inferior de 3 cm y laterales de 2,5 cm.
9. Se corregirá la claridad y exactitud de la pregunta, en ningún caso se expondrán fundamentos no preguntados, además de la claridad del documento.
10. En las actividades realice una captura de pantalla mínimo por actividad para verificar su autoría.

0. Antecedentes

Pasemos a ver como implementar Seguridad básica en nuestra Red *LAN* implementando un Servidor como Proxy de nuestra Red. La implementación de un servidor Proxy es fundamental para nuestra Red ya que realizamos una seguridad adicional muy necesaria para salvaguardar la integridad interna rechazando las amenazas externas y, por supuesto, filtrando tráfico.

Como hemos venido trabajando hasta ahora vamos a utilizar nuestro máquina virtual **minirouter** simulando el router de nuestra Red y además una máquina servidor que será nuestro **servidor proxy** y un **cliente** para comprobar la correcta implementación. Recordad el **cambio de nombre de host** en los ficheros "*/etc/hosts*" y "*/etc/hostname*"



1. Introducción

Como podemos observar en la imagen anterior podremos observar un servidor Proxy donde su trabajo fundamental es hacer de intermediario entre los *PC's* de la red y el router de conexión a Internet. Ya vimos en la teoría el funcionalidad principal de esta implementación y que podemos resumir en, por ejemplo, cuando un *PC* desea acceder al Internet, este equipo realiza la petición al servidor Proxy y es el *Proxy* quién accede a Internet (la seguridad aquí es clara, ocultamos al equipo el acceso a exterior y todo tráfico se hace a través del *Proxy*). Por tanto, el *PC* realiza la petición al *Proxy* y este se la pasará al router.

1.1. Ventajas de la implementación de un *Proxy*

Podemos resumir la principales ventajas de la implementación de un Servidor Proxy como las siguientes:

- Los equipos no tienen acceso al *router* como hemos venido diciendo. Todas la comunicación pasarán por el *Proxy* teniendo todo el flujo de comunicación bajo control. Esto es importante porque podemos permitir o denegar acceso *web*, *ftp*, *mail*...
- Recordemos el trabajo de *proxy* caché: Las páginas se guardan en memoria temporal del *proxy*, obviamente esto acelerará la descarga cuando varias estaciones de nuestra *LAN* acceden a las mismas páginas.
- Limitar el acceso a *url's*.
- Controlar el tráfico a subredes o equipos concretos de la *RED*. Por ejemplo, y en relación al caso anterior podemos controlar el trafico para diferentes aulas o equipos si en la red *10.0.X.Y* donde X sería las subredes u oficinas e Y los equipos según la oficina.
- Mantener un registro "*log*" de todas las conexiones en nuestra Red, así podemos ver qué páginas de contenido inadecuado acceden los usuarios de nuestra red.
- Añadir un plus de seguridad ante ataque externos, empezamos a construir nuestro primer "*firewall*".

1.2. Inconvenientes de la implementación de un Proxy

Es importante conocer también los inconvenientes de la implementación de un servidor *Proxy* para evaluar si es recomendable implantarlo o no:

- Necesitaremos configurar todas las aplicaciones de que dispondremos de un servidor *Proxy*: navegadores, cliente *ftp*, cliente correo...
- Si el *Proxy* falla la conexión a Internet se queda inactiva. Esto es simplemente porque como ya sabemos todas las conexiones pasan a través de nuestro *Proxy*. Una solución sencilla es tener uno de repuesto preparado.

- Estar manteniendo el *Proxy* constantemente.

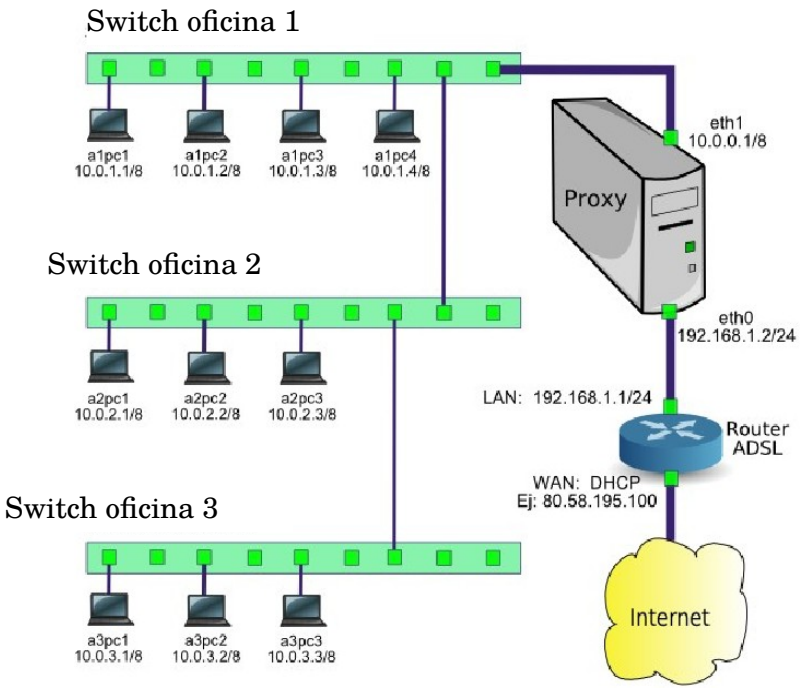
1.3. Un ejemplo de implementación

Veamos un ejemplo de configuración para una Red *WAN* en la que tenemos varios edificios, oficinas y equipos. Para ello vamos a utilizar la configuración interna más global y la que nos da más autonomía de implementación que es una de clase A con rango *10.0.0.0/8* siguiendo el esquema "*10.W.X.Y*" donde vamos a utilizar rango exactos con "*W*" para el edificio, "*X*" para el aula e "*Y*" para el número del PC. Como ya sabemos esta configuración nos permite 254 edificios, 254 oficina y 254 hosts en cada oficina.

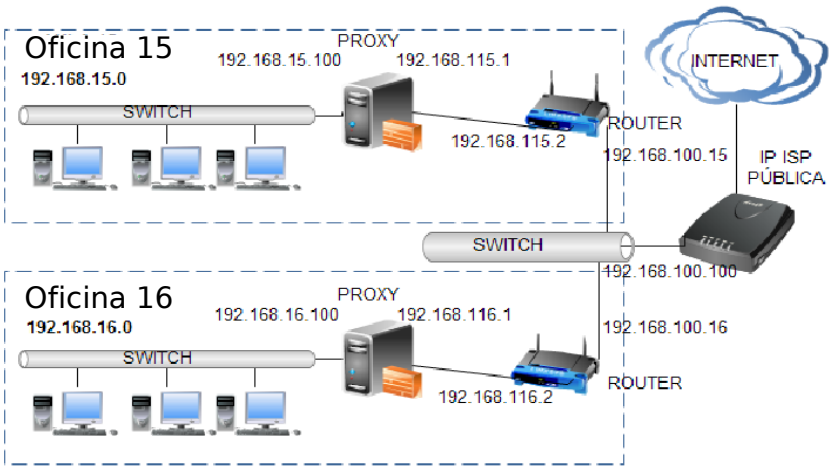
Adicionalmente, podremos usar un servidor Proxy para controla aún mas el acceso a internet y la seguridad de nuestra red por oficina. Comentar que podremos usar el rango de direcciones 1*92.168.X.Y* donde *X* representa al número de oficina e Y al host dentro de la oficina.

Por último resaltar que un servidor *Proxy* dispone de dos interfaces e red, esto es, simple y llanamente para poder separar las redes, por un lado la red de la oficina y por otro la red que va al *router* que da salida a Internet.

Ejemplo de un Proxy para una *LAN*



Ejemplo de Proxy con router por oficina



2. Proxy *Squid*

2.1. Introducción

Squid es uno de los paquetes de servidores proxy más importante que existen en GNU/Linux. Entre las característica más importantes podemos resaltar:

- ***Proxy* caché:** Proporciona servicio *ftp*, *https* y *http* a los clientes que se encuentren en la red LAN de manera que almacena localmente dichas solicitudes para agilizar posteriores solicitudes.
- ***Proxy SSL*:** Acelera las peticiones como como *proxy* caché pero con la peculiaridad de que dichas peticiones a internet se realizarían cifradas añadiendo un plus de seguridad.
- ***ICP*, *HTCP*, *CARP*, Caché *digests*:** Estos protocolos tienen la peculiaridad de conocer a un proxy si otros proxy caché tienen un recursos determinado almacenado.
- ***Proxy* transparente:** Es un tipo de *proxy* bastante interesante ya que actúa como firewall de manera que las solicitudes son enrutadas por medio de unas reglas y enviadas al *proxy Squid* con la peculiaridad de que no hay que configurar los clientes de la red

- ***WCCP*:** Con la premisa de que el proxy es el que establece el control de la conexión, la peculiaridad es que intercepta redirige el tráfico que un *router* hacia los *proxy* caché.

- **Control de Accesos:** Claramente esta configuración establece reglas de control de acceso, permitiendo denegar o aceptar tráfico.

- **Aceleración de servidores *HTTP*:** Esta configuración acelera enormemente el tiempo de respuesta de nuestra conexión a internet de manera que las solicitudes se almacenan en la caché de Squid y si hay una misma solicitud devolverá la información almacenada. En caso de haber alguna actualización deberá ser actualizada en la caché.

- ***SNMP*:** Permite activar el protocolo *SNMP* de manera que se podrá supervisar, analizar y comunicar información de estado entre las máquinas de nuestra red.

- **Caché de resolución *DNS*:** Encargado de la búsqueda de nombres de dominio en nuestra red.

Enlaces de referencia:

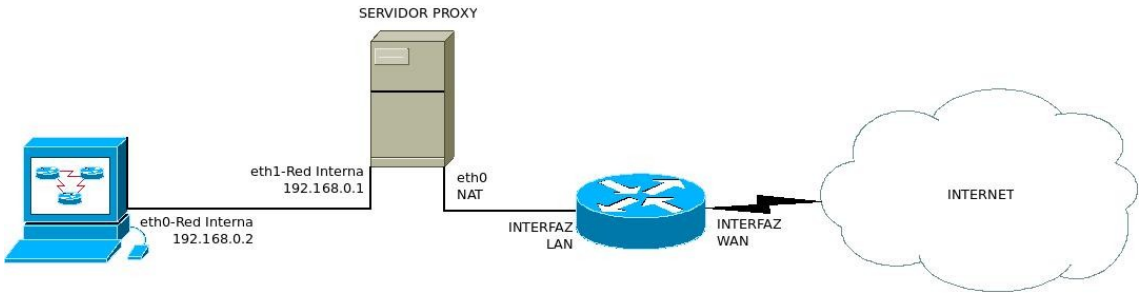
- http://tuxjm.net/docs/Manual_de_Instalacion_de_Servidor_Proxy_Web_con_Ubuntu_Server_y_Squid/html-multiples/index.html
- http://www.squid-cache.org/
- https://help.ubuntu.com/community/Squid

3. Configuración de las máquinas virtuales

Una vez visto los aspectos teóricos de la importancia de implementación de un Proxy en nuestra red y en concreto el paquete *Squid,* vamos a proceder a la configuración de nuestras máquinas.

Para esta práctica vamos a **prescindir de nuestro *minirouter*** para no aumentar la complejidad de configuración y asumimos la implementación

bajo el router local de nuestra red. Para ellos vamos a necesitar un servidor y un cliente con la siguiente configuración de red como muestra la imagen.

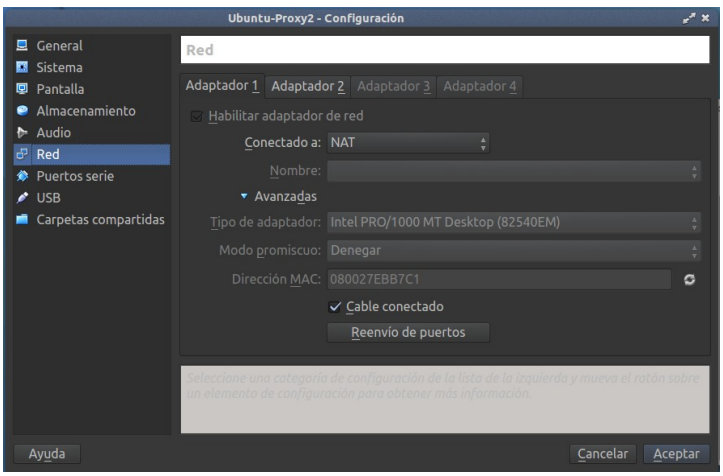


3.1. Configuración del servidor

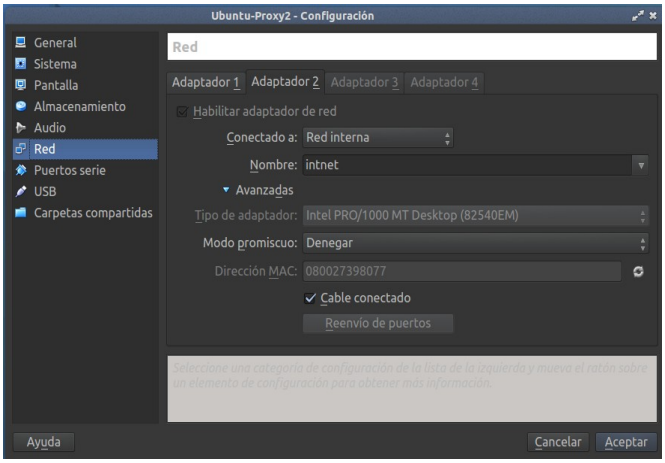
Veamos la configuración en la parte del servidor. Aunque vamos a ver la configuración posteriormente y aprovechando que hemos visto ya el tema de *DHCP*, el servidor lo vamos a configurar como servidor *DHCP* y *Proxy*. Ya sabemos que un servidor Proxy tiene dos tarjetas de red una que deriva el tráfico al *router* y otra que lleva la configuración interna. Recordemos que ninguna tarjeta de Red deberá tener la misma dirección física o *MAC* si es así deberemos refrescar dicha dirección.

Por tanto el primer paso será ir a ***Configuración → Red*** y activar **dos tarjetas de red:**

- El adaptador 1 conectado a **NAT** para *eth0*.

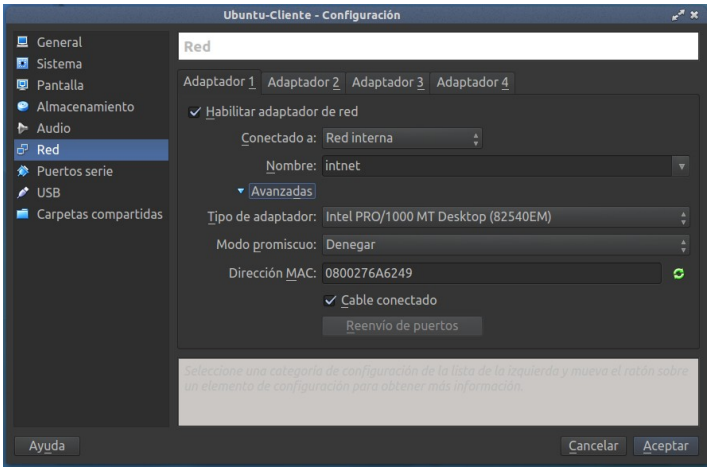


- El adaptador 2 conectado a la red interna para *eth1*.



3.2. Configuración del cliente

Por otro lado tenemos la configuración del cliente como podremos observar en la imagen el cliente se encuentra en la red interna y para salir a la externa deberá pasar por el Proxy como ya hemos explicado. Por tanto, para el cliente es simple, Solamente necesitaremos una interfaz de red o Adaptador 1 conectado a la Red interna.



4. Configuración de red

Una vez vista la configuración física de la red en el cliente y servidor pasemos a la configuración lógica de la misma modificando el fichero "/etc/network/*interfaces*". Este punto solamente vamos a modificar en la parte servidor ya que los clientes solicitaran una *IP* por *DHCP* a dicho servidor como veremos en el punto 6.

El primer aspecto es saber el nombre de las interfaces que están operando. Para ello como sabemos:

```
# ifconfig -a
```

Una vez sabidas las interfaces como siempre modifiquemos el archivo de configuración. Para la interfaz que actúa por ***NAT*** no deberemos modificarla ya que es la de salida pero la interfaz de red para la **red Interna** deberemos indicar una dirección fija. En mi caso como podemos observar voy a utilizar ***eth1*** que actuará como *NAT* y ***eth3*** que actuará para la red interna, para el vuestro no tiene que ser así por lo que hay que configurar correctamente este aspecto para que no problemas en la hora de asignación:

```
# Configuración IP Static
        auto eth3
        iface eth3 inet static
        address 192.168.0.1
        netmask 255.255.255.0
        broadcast 192.168.0.255
# The primary network interface
        auto eth1
        iface eth1 inet dhcp
```

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# Configuracion IP Static
auto eth3
iface eth3 inet static
address 192.168.0.1
netmask 255.255.255.0
broadcast 192.168.0.255
#allow-hotplug eth1

# The primary network interface
 auto eth1
 iface eth1 inet dhcp
```

5. Configuración del proxy *Squid* como servidor *DHCP*

Un aspecto adicional que vamos a ver y para recordar prácticas pasadas es configurar nuestro servidor *Proxy* como servidor *DHCP* también para que las computadoras internas a nuestra mini red *LAN* le soliciten una *IP* del rango de *IPs* configuradas.

5.1. Instalación del paquete *DHCP*

El primero paso para que un servidor actúe como "*router*" y asigne direcciones IPs dinámicas, será la instalación del paquete *DHCP*

```
# sudo apt-get install isc-dhcp-server
```

5.2. Configuración DHCP en el servidor

Como ya conocemos deberemos modificar los parámetros de servicio *DHCP*, para el cual deberemos de modificar el fichero "***/etc/dhcp/dhcpd.conf***". Aunque primeramente hagamos una copia de seguridad.

```
# sudo cp /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.ori
```

Y editamos el fichero:

```
# sudo vim /etc/dhcp/dhcpd.conf
```

Añadiendo/modificando los siguiente

```
option domain-name-servers 8.8.8.8, 8.8.4.4;
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.10 192.168.0.40;
        option domain-name-servers 8.8.8.8, 8.8.4.4;
        option routers 192.168.0.1;
        option broadcast-address 192.168.0.255;
        default-lease-time 600;
        max-lease-time 7200;
}
```

```
# option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers 8.8.8.8, 8.8.4.4;

default-lease-time 600;
max-lease-time 7200;

subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.10 192.168.0.40;
        option domain-name-servers 8.8.8.8, 8.8.4.4;
        option routers 192.168.0.1;
        option broadcast-address 192.168.0.255;
        default-lease-time 600;
        max-lease-time 7200;
}
```

Como vemos la configuración de *DHCP* que hemos realizado es dinámica de manera que no hay *IPs* reservadas y para lo cual asigna dentro del rango que se ha determinado. Si no nos acordamos vayamos a la práctica de *DHCP* y veremos como se podían asignar de forma estática y dinámica direcciones.

Por último reiniciemos el servicio *DHCP* para que actualice la nueva configuración de red.

```
# sudo service isc-dhcp-server restart
```

5.3. Comprobar asignación *IP* en el cliente

Para el cliente al estar en la red Interna debería haber obtenido una *IP* del rango especificado en el servidor. Comprobemos si es así

```
# ifconfig -a
```

```
manuel@cliente1:~$ ifconfig -a
eth1      Link encap:Ethernet  direcciónHW 08:00:27:6a:62:49
          Direc. inet:192.168.0.10  Difus.:192.168.0.255  Másc:255.255.255.0
          Dirección inet6: fe80::a00:27ff:fe6a:6249/64 Alcance:Enlace
          ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
          Paquetes RX:26 errores:0 perdidos:0 overruns:0 frame:0
          Paquetes TX:46 errores:0 perdidos:0 overruns:0 carrier:0
          colisiones:0 long.colaTX:1000
          Bytes RX:3031 (3.0 KB)  TX bytes:7462 (7.4 KB)
```

En caso de no ser así, se deberá configurar el archivo "***/etc/network/interfaces***" de manera que reciba la *IP* por *DHCP*.

```
The primary network interface
auto eth1
iface eth1 inet dhcp
```

Y por último reiniciar el servicio para que obtenga la nueva configuración.

```
# sudo service networking restart
```

6. Instalación de *Squid*

Ya hemos realizado la instalación del servidor *DHCP* y hemos verificado que se asignen direcciones *IP* de forma automática. A partir de ahora nos vamos a centrar en el trabajo con nuestro servidor *Proxy*.

6.1. Instalación del paquete

Lo primero que necesitaremos realizar es instalar nuestro servidor *Proxy* a través del paquete ***Squid***.

```
# sudo  apt-get install squid
```

6.2. Archivos de configuración de *Squid*

Antes de comenzar a implementar nuestra práctica veamos los archivos instalados con el paquete ***squid***. El archivo principal es */etc/squid/squid.conf* que hace referencia a otros cuantos ficheros.

```
# ls -ltr  /etc/squid
```

```
manuel@ub-sr-2:~$ sudo ls -ltr /etc/squid3/
total 212
-rw-r--r-- 1 root root 206944 ene 30  2013 squid.conf
-rw-r--r-- 1 root root   1547 ene 30  2013 errorpage.css
-rw-r--r-- 1 root root    421 ene 30  2013 msntauth.conf
```

De momento vamos a centrarnos en el archivo principal como hemos comentado, ya comentaremos llegado a su tiempo en los demás.

6.3. Copia de seguridad al fichero principal

Ya hemos comentado que muy importante realizar una copia de seguridad a nuestros ficheros de configuración para podre realizar un *backup* en caso de emergencia.

```
# sudo cp  /etc/squid/squid.conf  /etc/squid/squid.conf.ori
```

6.4. Manipulando los servicios *Squid*

Con la instalación del paquete *Squid* se han instalado también los scripts de manipulación del servicio en "***/etc/init.d/squid***". Veamos los servicios:

- Arranque del Servicio:

```
# sudo service squid start
# sudo  /etc/init.d/squid start
```

- Parar el Servicio:

```
# sudo service squid stop
# sudo  /etc/init.d/squid stop
```

- Reiniciar el Servicio:

```
# sudo service squid restart
# sudo /etc/init.d/squid restart
```

- Recargar la configuración del Servicio:

```
# sudo service squid reload
# sudo /etc/init.d/squid reload
```

7. Configurando el firewall del Servidor

Aunque tenemos un tema que explica al detalle el *firewall* de un servidor con el paquete *IPTABLES* necesitaremos configurar nuestro servidor para que pueda asignar direcciones a los clientes que lo soliciten. Se ha creado un fichero para la configuración que vamos a realizar en estos momentos y en él podemos observar una breve explicación de cada sentencia que realiza. Un aspecto muy importante es cambiar las variables de "***INTERNET***" y "***LAN_IN***" por las interfaces de red correctas según hayamos configurado nuestro servidor.

El primer paso que deberemos realizar es la descarga del archivo para esta práctica, el archivo está configurado para un ***proxy* transparente**:

- bit.ly/IplUZw

Tener en cuenta que en el fichero se encuentran las interfaces correctas para la configuración de la guía no para las de su máquina. Por tanto, cambie el nombre de las interfaces (expuesta como constantes al comienzo del script) por la correcta configuración de sus máquinas.

Así mismo, tener en cuenta que la siguiente regla firewall es el que nos otorga acceso a Internet para los clientes en nuestra red local.

```
# iptables -t nat -A POSTROUTING 192.168.0.1/24 -o enp0s3 -j MASQUERADE
```

Para no tener que configurar *IPTABLES* cada vez que reiniciemos el equipo vamos a hacer que lance el script cada vez que se reinicie nuestro equipo. El primer paso es copiar el *script* descargado a "**/etc/init.d**"

```
# sudo cp iptables_proxy.sh /etc/init.d
```

Deberemos de cambiar el propietario y permisos:

```
# sudo chown root:root iptables_proxy.sh
# sudo chmod 700 iptables_proxy.sh
```

Por último editamos el fichero "***/etc/rc.local***" para que ejecute el *shell* que acabamos de crear al iniciar el sistema. Para ello añadimos antes de "exit 0" del fichero: XXX ver como es para ubuntu 18.04

```
### END INIT INFO
cd /etc/init.d
./iptables_proxy.sh
```

```
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Run /etc/rc.local if it exist
### END INIT INFO
cd /etc/init.d
./iptables_proxy.sh

PATH=/sbin:/usr/sbin:/bin:/usr/bin
```

Por último reiniciamos el servidor para que realice los cambios y una vez realizado verificamos que las nuevas reglas de Firewall.

```
# sudo shutdown -r now
# sudo iptables -S
```

8. Configuración del *proxy Squid*

Como hemos venido trabajando existen para los diferentes servicios una gran cantidad de parámetros de configuración. Para un *Proxy Squid* igualmente tenemos una gran cantidad aunque pasemos a ver los más relevantes.

8.1. Opciones para autenticación

El primer parámetro que vamos a ver es el parámetro que establece las opciones de autenticación del *Proxy*. Este parámetro es "***OPTIONS FOR AUTHENTICATION***". Este parámetro es muy importante ya que lo que realiza es solicitar una contraseña para poder navegar por Internet. Un servidor *LDAP* sería el encargado de almacenar los usuarios y contraseñas y en función de los grupos a los que pertenecen los usuarios se pueden

permitir o denegar acceso. Al no haber visto este tipo de Servidor no realizaremos esta configuración aunque es importante que si la utilizamos en nuestra red es una medida de seguridad bastante importante. Un gran inconveniente es que se deberían administrar usuarios uno a uno, inconveniente que nos lleva a una carga administrativa de red muy elevada. Nosotros vamos a configurar por redes oficinas debido a la simplicidad.

```
#http_access allow localnet

acl mi_red src 192.168.0.0/24
http_access deny mi_red

#acl aulas_sin_inet src "/etc/squid3/range_without_conection"
#http_access allow aulas_sin_inet

http_access allow localhost
```

8.2. Opciones de Red

Este parámetro establece el puerto de escucha del *proxy*, aunque es recomendable cambiarlo por materia de seguridad como vimos en *SSH* vamos a dejarlo como viene por defecto.

El sistema de proxy como hemos podido observar es un *proxy* transparente, es decir, un *proxy* en el que no es necesario configurar los navegadores de los equipos clientes para poder conectarnos a internet. Esto es muy recomendable ya que el *proxy* se mantiene en segundo plano y el usuario navegará por internet sin cerciorarse que todas sus conexiones se están redirigiendo a través del *Proxy*. Como anécdota en nuestra configuración *firewall* si nos fijamos vemos que las peticiones al puerto *80* las redirige al puerto 3128 y así las reciba *Squid*.

Por tanto vamos a indicar que el *proxy* trabaje de modo transparente, para ello modificamos el parámetro anterior por:

```
http_proxy 192.168.0.1:3128 transparent
```

```
# Squid normally listens to port 3128
http_port 192.168.0.1:3128 transparent

#  TAG: https_port
```

8.3. Control de Acceso

Este parámetro es muy importante pero debido a que no disponemos de una red veremos el concepto de implementación para la red en un solo cliente.

Siguiendo con la configuración este parámetro establece los permisos de accesos, es decir, deniega o permite la navegación por la red. Siguiendo una estructura correcta de implementación lo primero que deberemos realizar el crear un lista de control de acceso y posteriormente dar permisos a dichas listas.

Para implementar una lista deberemos crearla utilizando la palabra ***acl*** seguida del nombre de la lista y la condición que cumplirán los miembros. Podemos resumir en las siguientes:

- ***src***: *IPs* o *URLs* origen
- ***dst***: *IPs* o *URLs* destino
- ***port***: Puertos.
- ***proto***: Protocolos.

Veamos un ejemplo (**todavía no configuremos**):

Si para nuestra Red local utilizamos el direccionamiento *192.168.1.0/24* lo más normal sería crear una lista para definir a toda nuestra red.

```
acl todos src 192.168.1.0/24
```

Concretamente, como venimos viendo en los dibujos anteriores podremos observar una red local *192.168.X.0* para la oficina *X*, concretamente nosotros mostremos las oficinas *15* y *16*.

```
acl oficina15 src 192.168.15.0/24
acl oficina16 src 192.168.16.0/24
```

Otro aspecto muy importante es crear lista para los jefes de área o gerentes, es por ello que según la jerarquía de la empresa ellos tendrán unos permisos diferentes a los demás por lo que se deberán realizar utilizando *acl todos src 192.168.1.0/24* o su *IP* final pero de forma ordenada o por oficinas. Por ejemplo:

```
acl jefe15 src 192.168.15.10 192.168.15.11
```

Otro aspecto importante es crear un archivo para las diferentes oficinas o departamentos de una empresa, de manera que le concedamos acceso a ellos pero a través de este archivo (esto es muy utilizado ya que estructura nuestras listas y no pone en un mismo fichero todo dificultando la vista del mismo)

```
acl deptSeguridad src  /etc/squid/depinf
```

Por último deberemos dar permisos a las listas, para ello utilizamos la palabra clave ***http_access*** donde ***allow*** concederá permiso y ***deny*** denegará permiso. Veamos algunos ejemplos según la lista que haya realizado.

```
http_access allow todos
http_access deny oficina15
```

Nuestra configuración

Antes de configurar los accesos de nuestros clientes vamos a ver si funciona correctamente nuestro servidor. Buscamos la línea que pone a continuación:

```
http_access allow localhost
```

Y la cambiamos por:

```
http_access deny localhost
```

Actualizamos nuestro proxy:

```
sudo service squid reload
```

Si abrimos el navegador del servidor y buscamos alguna página veremos que no podrá abrir ninguna página, esto es síntoma que de que todo va bien.

Volvemos a poner ***allow*** y hacemos un ***reload*** del servicio para volver a la configuración inicial.

Veamos ahora la configuración para nuestro cliente, lo primero que deberemos realizar es configurar nuestro proxy para que permita todo el rango IPs en el que se encuentra nuestro cliente configurando el fichero "***/etc/squid/squid.conf***".

```
acl mi_red src 192.168.0.0/24
http_access allow mi_red
```

Como podemos observar le damos permiso con ***allow***, en cambio si denegamos el acceso con ***deny*** podremos comprobar que el cliente no tiene acceso a internet. Acuérdense de hacer un ***reload*** al servicio *Squid*.

Un aspecto importante es que si buscamos páginas que se encuentran en caché de nuestro servidor nos la abrirá por lo que deberemos de jugar con páginas que no hayamos abierto posteriormente desde el cliente y que se almacenen en caché.

Si hemos denegado el servicio y la página no se encuentra en caché no tendría que dar una salida parecida en el navegador a la que se muestra:

8.4. Opciones de memoria caché

Como ya sabemos del *Proxy*, una de sus funcionalidades más importantes es trabajar como memoria caché para no que las páginas se queden guardadas y no congestionar la red interna con el tráfico externo. Es decir, El parámetro "***cache_mem***" especifica la cantidad ideal de memoria que será usada para mantener el cache de los objetos entrantes en memoria.

Por lo tanto vamos a establecer un límite de nuestra memoria *RAM* para este parámetro. Lo más recomendable es utilizar un tercio del total de nuestra memoria. En mi caso:

```
cache_mem 512 MB
```

```
#       exceed this limit to satisfy the new requests.  When the load
#       decreases, blocks will be freed until the high-water mark is
#       reached.  Thereafter, blocks will be used to store hot
#       objects.
#Default:
 cache_mem 512 MB

#  TAG: maximum_object_size_in_memory   (bytes)
```

8.5. Opciones de caché de disco

Otra muy importante opción para que un proxy actúe como *Caché* es el parámetro "***cache_dir***". Este parámetro establece un espacio de disco duro utilizado para la caché. Es decir, define el directorio para almacenar los objetos en cache de disco, este parámetro define la ruta del directorio, el tipo de cache de disco y su tamaño.

En nuestra configuración vamos y teniendo en cuenta que hemos instalado el servidor en un Disco duro de *8GB* vamos a crearle una caché de disco de *1GB*. Un aspecto importante de este parámetro es que deberemos especificarle el formato (*Squid* utiliza un formato ***ufs***), la carpeta donde se almacenará la caché, el tamaño de la misma en *MB*, el número de subdirectorios de primer nivel y por último de segundo nivel.

Veamos mi configuración:

```
cache_dir ufs /var/spool/squid 1024 16 256
```

```
# Uncomment and adjust the following to add a disk cache directory.
cache_dir ufs /var/spool/squid3 1048 16 256

#  TAG: store_dir_select_algorithm
```

8.6. Permitir/Denegar desde ciertas *IPs* a *urls* concretas

Como ya hemos comentado con un servidor proxy podemos limitar el tráfico externo tanto a una *IP* especifica como a un rango de *IPs* que por ejemplo pertenecen a una misma oficina.

Como ya hemos estado estudiando es recomendable que se creen diferentes ficheros para permitir o denegar accesos y que no estén todas las

listas en el fichero "***squid.conf***". Para nuestro caso vamos a crear un fichero para ver como afecta esta opción en nuestra implementación.

Por ejemplo vamos a crear un fichero para poner nuestra lista de acceso (deberemos asegurarnos de que no tener el mismo rango o IPs permitidas y denegadas a la vez).

```
# sudo vim /etc/squid/range_without_conection
```

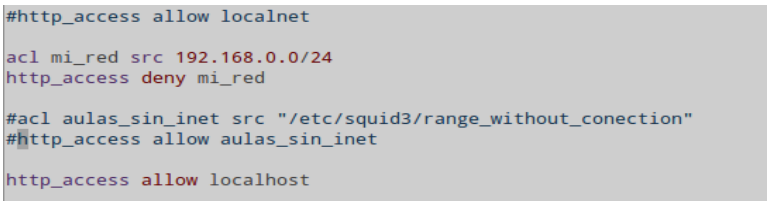
Y lo editamos de manera que representaríamos los rangos, en nuestro caso:

```
192.168.0.0/24
```

Y el fichero de configuración de ***squid***:

```
acl aulas_sin_inet src "/etc/squid/range_without_conection"
http_access deny aulas_sin_inet
```

Veamos la imagen:



Acordémonos de que siempre que hagamos una configuración deberemos recargar o reiniciar el servicio *Squid:*

```
# sudo /etc/init.d/squid reload
```

8.7. Denegar páginas *webs*

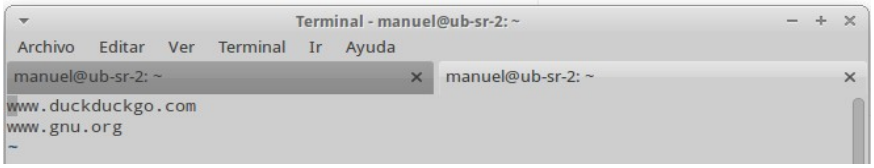
Como ya sabemos también un *Servidor Proxy* nos permite controlar el tráfico a determinadas páginas web según sean las políticas de la empresa. Esta opción es de gran importancia cuando queremos cerrar el tráfico a dichas webs de forma que los usuario de la subred no podrán conectarse a ellas.

Para ello lo primero que vamos a realizar es crear un fichero de texto en el que vamos a especificar las páginas prohibidas.

```
# sudo vim /etc/squid/webs_banned
```

Especificamos las webs que queramos prohibir, por ejemplo:

```
www.duckduckgo.com
www.gnu.org
```



Podemos especificar dominios completos o dominios de nivel superior:

```
.twitter.com
.tv
.xxx
```

Y ahora en vamos a indicarle que cargue el fichero en nuestro archivo "***squid.conf***".

```
# sudo vim /etc/squid/squid.conf
```

Introduciendo nuestras listas de acceso:

```
acl webs-den dstdomain "/etc/squid/webs_banned"
http_access deny webs-den
```

```
acl expr-den url_regex "/etc/squid3/expr_banned"
http_access deny expr-den

acl webs-den dstdomain "/etc/squid3/webs_banned"
http_access deny webs-den

acl mi_red src 192.168.0.0/24
http_access allow mi_red

http_access allow localhost
```

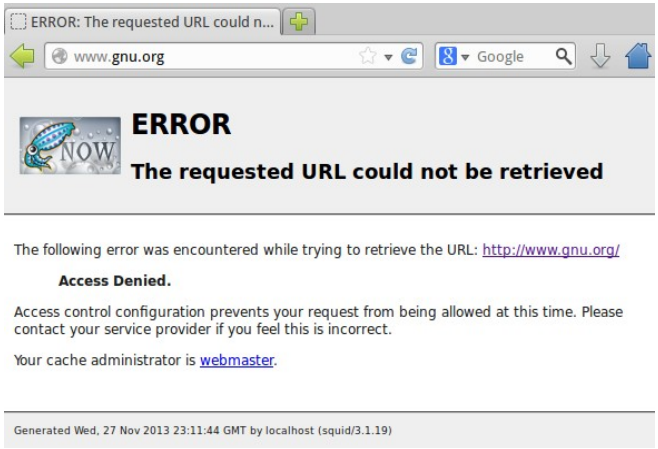
IMPORTANTE: El archivo de configuración S*quid* se lee de arriba hacia abajo de forma secuencial es por lo que deberemos tener mucho cuidado con el orden a la hora de fijar reglas. Las reglas se deberán, en el orden de reglas de denegación al principio y permitidas a continuación en la sección:

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

Recargamos el servicio:

```
# sudo /etc/init.d/squid reload
```

Vemos la salida desde el navegador:



8.8. Denegar palabras en urls

El parámetro tipo "***url_regex***" nos permite denegar en la barra de dirección del navegador cualquier página que contenga las palabras especificadas al igual que con el punto anterior vamos a crear un archivo para almacenar algunas palabras:

```
# sudo vim /etc/squid/expr_banned
```

Con el siguiente contenido:

```
adult
celebri
mp3
otrositioindeseable.com
playstation
porn
sex
sitioindeseable.com
taringa
```

Procedemos a la configuración del archivo "***squid.conf***".

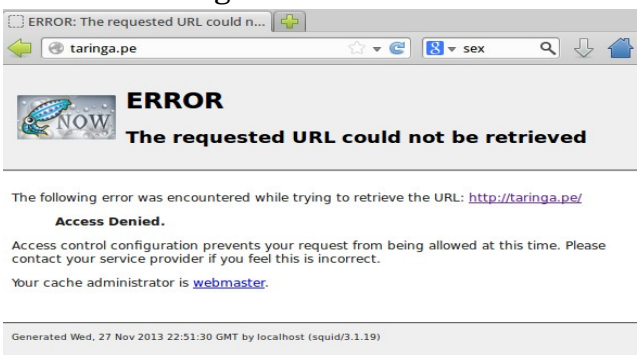
```
acl expr-den url_regex "/etc/squid/expr_banned"
http_access deny expr-den
```

```
acl expr-den url_regex "/etc/squid3/expr_banned"
http_access deny expr-den

acl webs-den dstdomain "/etc/squid3/webs_banned"
http_access deny webs-den

acl mi_red src 192.168.0.0/24
http_access allow mi_red

http_access allow localhost
```

Recargamos el servicio:

```
# sudo /etc/init.d/squid reload
```

Comprobamos en el navegador:



9. Analizando las conexiones

Como administradores de sistemas deberemos de monitorizar la red de forma que si vemos que un usuario o varios usuarios están accediendo a diferentes *webs* que no tienen nada que ver con el trabajo y o que están llevando un gran consumo de ancho de banda procederemos a cerrar dichas conexiones.

En el archivo "***/var/log/squid/access.log***" podremos ver que *PCs* acceden a Internet la fecha y hora, el *PC* y la *url* a la que ha accedido.

Veamos la salida de mi archivo con las pruebas que realizado:

```
# sudo more /etc/var/log/squid/access.log
```

```
1385594454.339    306 192.168.0.10 TCP_MISS/200 919 POST http://ocsp.digicert.com/ - DIRECT/72.21.91.29 application/ocsp-response
1385594454.607    313 192.168.0.10 TCP_MISS/200 919 POST http://ocsp.digicert.com/ - DIRECT/72.21.91.29 application/ocsp-response
1385594462.276      0 192.168.0.10 TCP_DENIED/403 3976 GET http://en.wikipedia.org/wiki/taringa - NONE/- text/html
```

10. Archivo de configuración automática del proxy

Aunque nosotros hemos realizado un *proxy* transparente podemos crear un archivo de configuración automática del *proxy* de manera que en caso de que no fuera transparente no tuviéramos que modificar los navegadores uno a uno. Para ello necesitaremos tener instalado el servidor ***Apache***.

En sí la explicación es bien sencilla si queremos realizar una conexión interna señalaremos mediante el siguiente *script* que la conexión se haga directa en caso de ser externa que se realice a través del *proxy*.

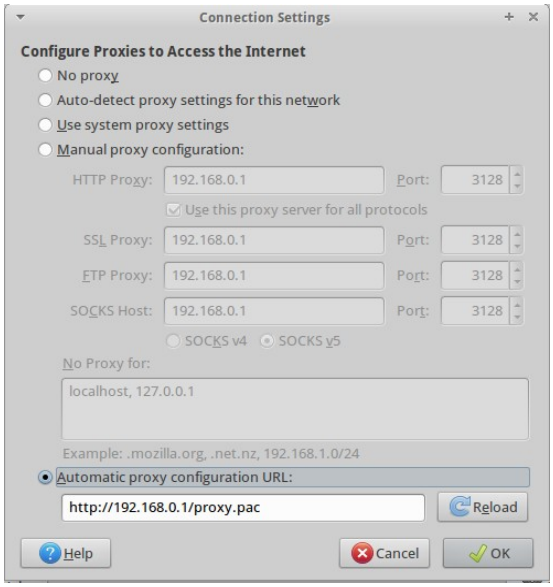
Para ello creamos un archivo en el directorio *"**/var/www**"*

```
# sudo vim /var/www/proxy.pac
```

Con el siguiente contenido:

```
function FindProxyForURL(url,host){
        if (isInNet(host, "192.168.0.0", "255.255.255.0"))
                return "DIRECT";
        else if (isInNet(host, "127.0.0.1", "255.255.255.255"))
                return "DIRECT";
        else return "PROXY 192.168.0.1:3128";
}
```

Y el cliente tendría la siguiente configuración:



11. Ampliando la seguridad de nuestra Red

Como hemos estado viendo *Squid* se configura de forma automática sin tener ningún tipo de autenticación. Vamos a ampliar la seguridad con *Squid* suponiendo que los usuario son fijos con sus contraseñas, es decir, vamos a suponer que existe un usuario con su correspondiente contraseña que va disponer de acceso total a internet.

Aunque existen varios mecanismos de autenticación que permiten controlar quien va a poder conectarse a Internet como LDAP, SMB, PAM vamos a utilizar el mecanismo ***ncsa_auth***.

Para requerir autenticación para acceder al servidor proxy necesitamos una lista con usuarios permitidos y sus respectivas claves de autenticación. Por tanto, creamos el fichero */etc/squid/squid_passwd*, se le asigna el usuario "proxy" y grupo "proxy" como propietarios, y se le dá permisos de lectura/escritura únicamente al propietario del fichero.

Para ello lo primero que vamos a crear es el archivo que almacenará los logins de los usuario junto a sus contraseñas cifradas, almacenando en el archivo parejas de valores nombre_usuario:contraseña.

```
# sudo touch  /etc/squid/squid_passwd
```

Cedemos la propiedad del archivo al usuario "proxy":

```
# sudo chown proxy:proxy etcsquid/squid_passwd
```

Concedemos permisos al usuario "proxy" para que él solo pueda leer y escribir en el archivo:

```
# sudo chmod 600 etc/squid/squid_passwd
```

Una vez creado el fichero y dados los permisos adecuados, tendremos que hacer uso de la aplicación "***htpasswd***", del paquete "***apache2-utils***" para generar un usuario/contraseña encriptada en dicho fichero.

```
# sudo apt install apache2-utils
```

Damos de alta a los usuarios (deberemos introducir la contraseña y confirmamos) mediante la aplicación recién instalada:

```
# sudo httpasswd /path/to/file [usuario]
```

A nivel de archivo de configuración "***/etc/squid/squid.conf***" hay que incorporar una serie de parámetros necesarios:

- Indicar el programa de autenticación que va a ser utilizado (en nuestro caso *basic_ncsa_auth*). En esta línea establecemos que el programa *basic_ncsa_auth* se encargue de la autenticación teniendo como registro de usuarios y contraseñas al fichero */etc/squid/squid_passwd*.

```
auth_param basic program /usr/lib/squid/basic_ncsa_auth
/etc/squid/squid_passwd
```

- Se establece como lista de control de acceso la obligación de autenticarse ante Squid. En la sección de listas de acceso agregamos dos nuevas *acl*, ambas con distinta opción.
  - Para una de ellas estableceremos una *acl* en relación a un usuario en particular, por lo que la estructura a utilizar es:

```
acl [nombreMaquina] ident [usuario]
```

  - Dicho usuario tiene que estar registrado en nuestro fichero *squid_passwd* como ya se mostró al usar *htpasswd*. Esta *acl* se puede utilizar para restringir o permitir el acceso de este usuario en particular una vez se haya identificado. Para requerir la autenticación de cualquier cliente añadimos la siguiente *acl*.

```
acl control proxy_auth REQUIRED
```

- Ahora pasamos a modificar la sección de accesos. Para que se nos bloquee el acceso hasta que nos autentifiquemos debemos:
  - primero comentar el permiso de acceso creado para nuestra red ya que, de lo contrario, ya no necesitaríamos autenticarnos y la restricción de cualquier host en general pues de mantenerla se nos bloquea automáticamente la conexión.

- Por último debemos agregar los parámetros para permitir el acceso en caso se autentifique, y en el caso particular que sea el usuario que hemos creado.

```
http_access allow aula usuario

http_access allow control
```

- Por último, hay que relanzar el servicio:

```
# /etc/rc.d/init.d/squid restart
```

Comprobar que para conectarnos nos requiere autenticación.

Práctica

1. Estudiar como borrar la memoria caché de nuestras páginas e implementarlo en nuestro código. Realiza una captura del código modificado.
2. Busque información sobre que es y representa el archivo "***/var/log/squid/cache.log***" y el directorio "***/var/spool/squid***"
3. Busque información sobre los siguientes parámetros e implémentalos en el fichero correspondiente:
    - En cache_dir es más recomendable usar el parámetro ***aufs*** en vez de ***auf.***
    - ***maximun_object_size***
    - ***cache_swap_low*** y ***cache_swap_high***
    - ***cache_replacement_policy***
    - ***error_directory***
4. Busque como restringir de acceso a contenido por extensión y realice una implementación. El parámetro *acl* es ***urlpath_regex.***
5. Otra forma bastante interesantes es restringir por horarios. Realice una implementación dónde se vea claramente este aspecto. El parámetro *acl* es ***time***.
6. Estudie como cambiar la página de acceso denegado en el navegador.
7. Existe el concepto caché en jerarquía en proxys. Busque información de qué y su implementación.
8. Realizar un proxy de acceso por autenticación con servidor *LDAP.*
9. Incluir supervisión contra virus en Squid con SquidClamAV Redirector.
10. Crea un archivo de configuración automática donde permitas acceso directo a Internet a las redes del centro, la dirección de loopback del propio servidor y la página web del centro. El resto de posibles redes de destino deben "salir" a Internet a través del servidor proxy.