



2020

En este documento se describirán las siguientes conceptos: Firewall (iptables, nftables), proxy (squid), uso de reglas MANGLE, Radius con DNie, DansGuardian, ELK.

S21AR

SEGURIDAD PERIMETRAL EN LA INTRANET

José Antonio Díaz Gómez

Índice

1.	Introducción.....	2
2.	Iptables, nftables.....	2
2.1.	Iptables.....	2
a.	Introducción.....	2
b.	Explicación.....	2
c.	DIFERENCIA ENTRE NEW, ESTABLISHED, RELATED e INVALID	6
d.	Instalación de Firewall dedicado a la seguridad Perimetral.	7
2.2.	Nftables	11
b.	Conversión de Iptables a nftables.	12
3.	SQUID (Proxy, proxy caché).....	14
3.1.	Introducción.....	14
3.2.	Conceptos sobre cachés	14
3.3.	Instalación.....	15
3.4.	Configuración elemental.....	15
3.5.	Recursos dedicados a Squid	16
3.6.	Control de acceso	17
3.7.	HTPPS (Docker)	23
3.8.	SARG	28
4.	Filtrador de contenido (DansGuardian).	31
5.	Radius con DNle.	36
b.	Instalación de FreeRADIUS.....	37
c.	Configuración de FreeRADIUS	37
d.	Punto de acceso	40
e.	Clientes	43
f.	Pruebas	48

1. Introducción.

En este documento se explicaran los siguientes apartados:

2. Iptables, nftables.

2.1. Iptables

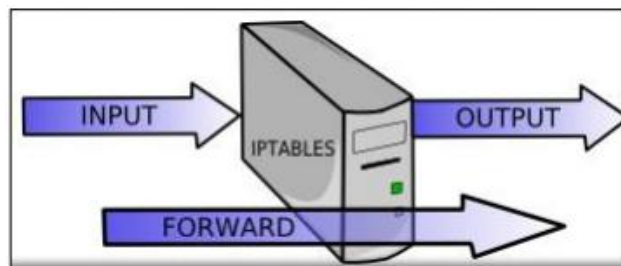
a. Introducción.

Cuando utilizas un sistema Linux para conectar tu red local a Internet, tienes la posibilidad de permitir o no cierto tipo de tráfico. Las cabeceras de los paquetes IP contienen información sobre el destino (de forma que se puede prevenir el acceso a ciertos sitios de Internet), el origen (se pueden evitar conexiones desde sitios concretos de Internet). Otra información que se obtiene de las cabeceras es el protocolo utilizado (ICMP, UDP, TCP) y el puerto. Normalmente los protocolos de alto nivel utilizan para sus conexiones puertos determinados (también llamados well known sockets). De esa forma, la mayor parte de las peticiones de documentos html se harán a destinos de Internet por el puerto 80, el envío de correo se hará por el puerto 25, o las conexiones vía telnet se harán usando el puerto 23.

Mediante **iptables**, se puede filtrar el tráfico por una gran variedad de criterios, mediante reglas (algo así como sentencias si condición entonces acción). Las reglas se integran en tres grupos diferenciados, **reglas de entrada** (para paquetes que llegan al router), **reglas de encaminamiento** (decisión sobre encaminar paquetes o no), y **reglas de salida** (paquetes que salen del router por alguna interfaz). Este programa es tan versátil que nos permite indicar en cada regla, la cadena (entrada, encaminamiento, salida) el protocolo, interfaz, direcciones de origen y destino, y el puerto utilizado en la conexión, así como la acción a tomar (denegar, rechazar, aceptar o enmascarar) caso que determinado paquete cumpla la regla.

b. Explicación.

Un cortafuegos o firewall es una aplicación para permitir o denegar el tráfico entre dos redes, normalmente entre Internet (red externa) y una red corporativa (red interna o privada) basándose en una serie de reglas, denominadas filtros.



La característica fundamental de todo firewall es la de filtrar:

- **INPUT:** Paquetes de datos que van dirigidos a la propia máquina ya sea por la interfaz externa como la interna.
- **OUTPUT:** Paquetes de datos enviados desde el firewall hacia el exterior (ya sea internet, LAN, DMZ, etc.).

- **FORWARD:** Paquetes de datos que pasan de una red a otra es decir atraviesan el “firewall” ya sea desde la red externa hacia la interna o viceversa, pero nunca tienen como destino ni origen el firewall.

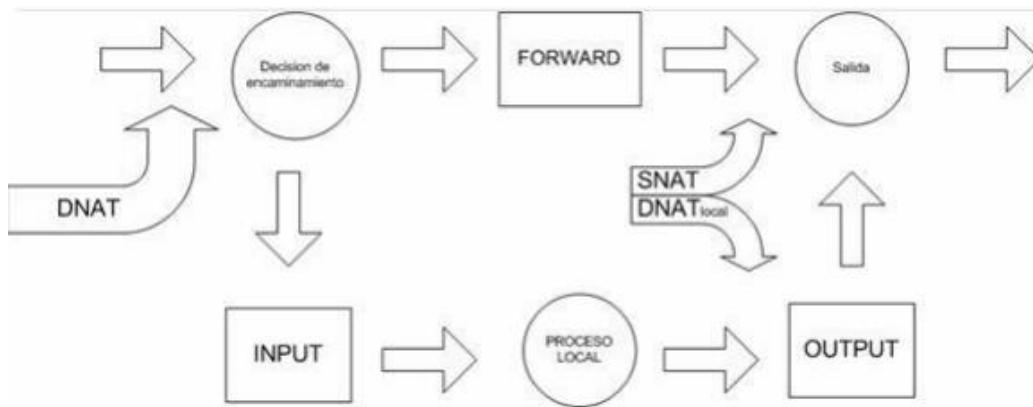
Cualquiera de estos paquetes podemos filtrarlos por IP/subred de origen/destino, protocolo, servicio/puerto origen/destino, MAC, interfaz de red, tiempo (hora, día), etc.

Además tiene otras funciones como la de registrar toda las comunicaciones (para después ser tratadas con diferentes fines), hacer NAT, etc.

Iptables es una herramienta de Linux (no es un servicio de red) que se implementa directamente en el Kernel de la máquina es decir es parte del sistema operativo, que se pone en marcha con un simple script shell en el arranque de la máquina (por ejemplo en el /etc/rc.local).

Hay dos maneras de implementar un firewall:

- Política por defecto **ACEPTAR**: en principio to lo que entra, sale o pasa por el firewall se acepta y solo se denegará lo que se diga explícitamente. No recomendada.
- Política por defecto **DENEGAR**: todo esta denegado, y solo se permitirá pasar, entrar o salir por el firewall aquellos que se permita explícitamente.



La estructura de una orden de iptables sigue el siguiente patrón:

iptables -t [tabla] – [tipo de operación] [cadena] -- [reglas con parámetros] –acción

Dónde:

- **-t [tabla]:** tabla puede tomar los siguientes valores.
 - **Filter:** es la tabla responsable de bloquear o permitir que un paquete continúe su camino. Es la tabla por defecto y por eso no es necesario ponerla en cada comando. Se usa con las cadenas **INPUT, OUTPUT, FORWARD** y se le aplican las reglas de **ACCEPT, DROP, REJECT, LOG**.
 - **Nat:** es la tabla responsable de traducir redirecciones de IP y/o puerto de origen y de destino, es decir lo que conocemos como NAT. Contiene las cadenas **PREROUTING** (paquetes entrante pasan a través de esta cadena antes de que se consulte la tabla de enrutado, por ejemplo cuando queremos que el puerto 80 lo redireccione al puerto squid 3128). O cadena de **POSTROUTING** (paquetes salientes pasan por esta cadena después de haberse tomado la decisión de enrutado, por ejemplo cuando queremos enmascarar mediante NAT el tráfico de la red interna por la IP de la tarjeta de la red externa o pública).

- **Mangle:** es la tabla responsable de ajustar las opciones de paquetes, como son la calidad del servicio (QoS, TTL, mark). Diseñada para efectos avanzados
- – **[tipo de operación]:** Comandos.
 - **A:** Añadir una regla al final de la cadena especificada.
 - **L:** listar cadenas de una determinada tabla o todas (#iptables -n -L -v). O si queremos ver las tabla nat (#iptables -L -t nat).
 - **D:** Borrar una reglas en tiempo real, se puede realizar mediante por el número o teclear la regla entera
 - **I:** Insertar (añadir al principio) una regla en tiempo real a menos que se especifique un número y lo hará delante de la regla indicada
 - **F:** Operación de flush (eliminar y reiniciar todas las cadenas de una determinada tabla que estén cargadas en el kernel).
 - **X:** Borrado de reglas personalizadas en algunas distribuciones por defecto.
 - **Z:** Inicializar los contadores de paquetes y bytes que aparece junto a cada regla.
 - **N:** Crea una nueva cadena con un nombre especificado por el usuario (es como crear un alias).
 - Etc.

- **Comparaciones generales e implícitas.**

-i	Interfaz de entrada (eth0, eth1, wlan0, lo, etc.). Este parámetro opcional puede ser usado solamente con las cadenas INPUT y FORWARD cuando es usado con la tabla filter y la cadena PREROUTING con las tablas nat y mangle. Ej. -i eth0
-o	Interfaz de salida (eth0, eth1, wlan0, lo, etc.). Configura la interfaz de red de salida para una regla y puede ser usada solamente con las cadenas OUTPUT y FORWARD en la tabla de filtro y la cadena POSTROUTING en las tablas nat y mangle. Ej. -o eth0
-s	Host/red de origen (desde donde procede el paquete). Ej. -s 192.168.1.0/24, -s 192.168.8.23
-d	Host/red de destino (hacia dónde va el paquete). Ej. -d 192.168.1.0/24.
-p	Protocolo que se va a comparar Ej. (-p icmp) (-p udp) (-p tcp).
--sport	Puerto de origen. Ej. (-p tcp -- sport 80) (-p tcp --sport 20:23)
--dport	Puerto de destino. Ej.: (-p tcp -- dport 22) (-p udp --dport 123, 45)

- **Comparaciones explícitas o filtros explícitos.**

Las comparaciones explícitas son aquellas que se deben cargar específicamente con la opción --m o --match.

--mac-source	iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01 Como ya se ha dicho, esta comparación se emplea para buscar paquetes basándose en su dirección MAC de origen. La dirección MAC indicada debe tener el siguiente aspecto: XX:XX:XX:XX:XX:XX, pues de lo contrario no sería correcta (el programa la consideraría una dirección "ilegal"). La comparación también puede ser invertida (la admiración vale con otras opciones) con !, y se parecerá a ! --mac-source 00:00:00:00:00:01 . En este ejemplo se aceptarán todos los paquetes excepto aquellos que provengan de la tarjeta Ethernet especificada (la que tenga la dirección MAC 00:00:00:00:00:01). Además, sólo será válida en las cadenas PREROUTING, FORWARD e INPUT y en ninguna más.
---------------------	---

--timestart --timestop --kernelz --days	<p>Ejemplo: iptables -A INPUT -p tcp -dport 22 -m time --timestart 09:00 --timestop 18:00 --kernelz --days Mon,Tue,Wed</p> <p>Ejemplo: iptables -A FORWARD -p tcp -dport 80 -m date --datestart yy:mm:ddT12:00:00 --datestop yy:mm:ddT13:00:00 --kernelz</p> <p>Hay que ponerle la opción --kernelz para que funcione como espera.</p>
--string	<p>Para comparar con cadenas de texto que viajan en la parte de datos. Hay que realizarlo mediante una inserción de regla, no mediante A.</p> <p>Ejemplo: iptables -I FORWARD -p tcp -dport 80 -m string --string "facebook.com" --algo kmp</p>
--state	<p>La extensión state (estado) se emplea conjuntamente con el código de seguimiento de conexiones (connection tracking) del núcleo. La comparación de estado accede a la máquina de seguimiento de conexiones y averigua en qué estado se encuentra el paquete. Éste procedimiento funciona con prácticamente todos los protocolos, incluyendo aquellos que no poseen estado, como el ICMP y el UDP.</p> <p>Ejemplo: -m state --state ESTABLISHED, RELATED</p>
Multiport	<p>La extensión multiport se emplea para especificar puertos múltiples y rangos de puertos múltiples. Si no tuviéramos la funcionalidad de esta comparación, necesitaríamos escribir múltiples reglas del mismo tipo simplemente para comparar diferentes puertos no contiguos (que no se pueden especificar mediante un rango).</p> <p>iptables -A INPUT -p tcp -m multiport --source-port 4543,334 iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110 iptables -A INPUT -p tcp -m multiport --port 22,53,80,110</p>

- **-j [acción]:**
 - **ACCEPT:** Aceptar el paquete.
 - **DROP:** Rechazar el paquete sin notificar al host origen.
 - **REJECT:** Rechazar el paquete y se envía notificación de error al host origen.
 - **REDIRECT:** Sirve para redirigir paquetes y flujos hacia una máquina de la red local o DMZ. También sirve para redirigir peticiones entre puerto del mismo firewall para la activación de servicios (como por ejemplo para llevar puerto 80 al 3128 (squid)).
 - **MASQUERADE:** Sólo válida para reglas de la tabla Nat usada por enmascaramiento de la dirección IP origen de forma dinámica.
 - **LOG:** Este objeto funciona para registrar información detallada sobre los paquetes que pasan por el firewall.
 - **Etc.**

Ejemplo:

```
# iptables -t filter -A FORWARD -p tcp -s 192.168.1.2 -d 192.168.0.3 -j ACCEPT
```

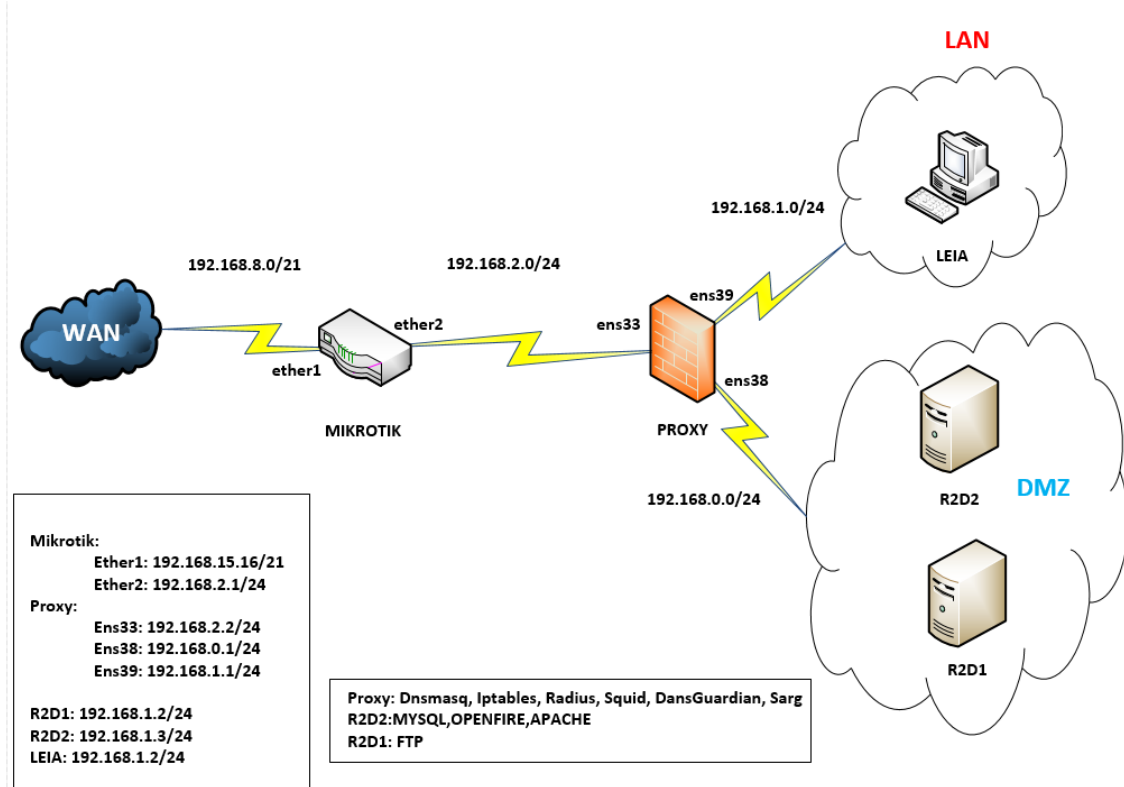
Componente	Descripción
-t filter	Vamos a trabajar con la tabla filter (tabla por defecto)...
-A FORWARD	...añadiendo la siguiente regla a su cadena FORWARD.
-p tcp	Selecciona los paquetes cuyo protocolo sea TCP...
-s 192.168.1.2	...cuya dirección origen sea 192.168.168.1.2...
-d 192.168.168.0.3	... y cuya dirección destino sea 192.168.0.3.
-j ACCEPT	Acepta esos paquetes para su reenvío.

c. DIFERENCIA ENTRE NEW, ESTABLISHED, RELATED e INVALID

Estado	Explicación
NEW	El estado " NEW " nos indica que el paquete es el primero que vemos. Esto significa que el primer paquete del módulo conntrack vea en una conexión será etiquetado de esta manera. Este comportamiento puede llevar a determinados problemas en determinados casos, pero también puede ser extremadamente útil si necesitamos captar conexiones perdidas de otros cortafuegos.
ESTABLISHED	El estado " ESTABLISHED " (establecido) ha visto tráfico en ambas direcciones y por tanto admitirá continuamente los paquetes de ese flujo. Las conexiones "establecidas" son bastante fáciles de comprender: el único requisito para alcanzar el estado " ESTABLISHED " es que un host envíe un paquete y obtenga una respuesta del otro host.
RELATED	El estado " RELATED " (relacionado) es uno de los más complejos. Una conexión se considera "relacionada" cuando está ligada a otra conexión ya "establecida". Por este motivo, para que una conexión se considere en estado " RELATED " primero deberemos de tener otra conexión en estado " ESTABLISHED ": la conexión será considerada como "relacionada" siempre que el módulo conntrack pueda entender que está relacionada con la principal.
INVALID	El estado " INVALID " (inválido) implica que el paquete no puede ser identificado o que no tiene ningún estado. Esto puede ser debido a varias razones, como que el sistema se ha quedado sin memoria disponible, o mensajes ICMP de error que no responden a ninguna conexión conocida. Normalmente es una buena idea eliminar (DROP) todo aquello que se encuentre en este estado.

d. Instalación de Firewall dedicado a la seguridad Perimetral.

Para la siguiente práctica este será nuestro esquema de red:



- Para ello deberemos de crear un script llamado por ejemplo firewall.sh y darle permisos de ejecución.

```
root@Proxy:/home/frodo# chmod +x firewall.sh
root@Proxy:/home/frodo#
```

- Convertir nuestro firewall en uno seguro, es decir que las políticas por defecto sea denegar todo (DROP). Comprobar que no podemos realizar ni ping ni ssh.

```
root@Proxy:/home/frodo# iptables -L -n
Chain INPUT (policy DROP)
target prot opt source destination

Chain FORWARD (policy DROP)
target prot opt source destination

Chain OUTPUT (policy DROP)
target prot opt source destination
root@Proxy:/home/frodo# ssh localhost
root@localhost's password:
Permission denied, please try again.
root@localhost's password:
```

- Abrir conexión vía localhost. Tenemos que abrirlo tanto en INPUT como en OUTPUT, ya que el paquete sale desde la máquina pero es que llega también a la misma máquina, sala por la interfaz -i y llega por la interfaz -o. Además tendremos que habilitar el enrutamiento entre las tarjetas de redes.


```
#Hacer de nuestra máquina un router
echo "1" > /proc/sys/net/ipv4/ip_forward

#Limpieza
iptables -F
iptables -X
iptables -Z
iptables -t nat -F
iptables -t nat -X
iptables -t nat -Z

#Establecer políticas por defecto.
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Respuestas para las conexiones establecidas
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#Permitiendo via localhost hacer cualquier cosa.
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

- Comprobación que desde la misma máquina (máquina virtual) funciona el ping y el ssh (ya sea poniendo localhost o la IP, eso da igual, ya que el paquete va localhost), pero no si lo hacemos desde nuestro equipo anfitrión no funciona ninguno de los dos. No sólo funcionarían estos protocolos, sería cualquiera que pongamos

```
root@Proxy:/home/frodo# ping localhost
PING localhost (localhost (:::1)) 56 data bytes
64 bytes from localhost (:::1): icmp_seq=1 ttl=64 time=0.018 ms
64 bytes from localhost (:::1): icmp_seq=2 ttl=64 time=0.058 ms
^C
--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1025ms
rtt min/avg/max/mdev = 0.018/0.038/0.058/0.020 ms
root@Proxy:/home/frodo# ssh localhost
root@localhost's password:
Linux Proxy 4.9.0-7-amd64 #1 SMP Debian 4.9.110-1 (2018-07-05) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 11 13:58:09 2020 from ::1
root@Proxy:~#
```

- Vamos a abrir el protocolo ICMP y servicio SSH para accesos externos e internos, es decir permitir los paquetes que procedan desde dónde sea pero que su destino sea el interfaz ens33 de firewall. Tendremos que permitir que nuestro firewall responda a peticiones externas e internas por eso es necesario añadir la línea (**iptables -t filter -A OUTPUT/INPUT -m state --state ESTABLISHED, RELATED -j ACCEPT**, previamente establecidos (ESTABLISHED y conexiones relacionadas a una establecida (RELATED)). No es necesario poner las NEW (ya que abre otras muchas reglas). Esta regla se podría haber puesto expresamente para el ping y el ssh, pero tal y como está escrita ya valdrá para cualquiera entrada.

```
# Respuestas para las conexiones establecidas
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#Permitiendo via localhost hacer cualquier cosa.
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

#Permitiendo paquetes icmp por todas las tarjetas tanto de entrada como de salida.
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -i ens38 -o ens33 -p icmp -j ACCEPT
iptables -A FORWARD -i ens39 -o ens33 -p icmp -j ACCEPT
#iptables -A FORWARD -i ens39 -o ens38 -p icmp -j ACCEPT

#Permitiendo las conexiones por ssh
iptables -A INPUT -i ens33 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -i ens33 -d 192.168.0.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -i ens33 -d 192.168.0.3 -p tcp --dport 22 -j ACCEPT
```

- Comprobamos que no hemos posibilitado que nuestro firewall pueda establecer comunicación (INPUT), por ejemplo vía ssh, ntp, dns, ping, http, etc. Con los equipo de la red intermedia (ens33). Tendremos que permitir dicha comunicación y además de permitir las respuestas establecidas. Al tener la regla de ESTABLISHED de INPUT y OUTPUT no hará falta poner las reglas de OUTPUT porque las reglas están permitidas en ambas direcciones.

```
#Permitiendo paquetes icmp por todas las tarjetas tanto de entrada como de salida.
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -i ens38 -o ens33 -p icmp -j ACCEPT
iptables -A FORWARD -i ens39 -o ens33 -p icmp -j ACCEPT

#Permitiendo las conexiones por ssh
iptables -A INPUT -i ens33 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -i ens33 -d 192.168.0.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -i ens33 -d 192.168.0.3 -p tcp --dport 22 -j ACCEPT

#iptables -A OUTPUT -o ens38 -p tcp --dport 22 -j ACCEPT
#iptables -A OUTPUT -o ens39 -p tcp --dport 22 -j ACCEPT

#Permitiendo las conexiones DHCP por puerto ens38
iptables -A INPUT -i ens38 -p udp -m multiport --dport 67,68 -j ACCEPT
iptables -A OUTPUT -o ens38 -p udp -m multiport --dport 67,68 -j ACCEPT

#Permitiendo las conexiones DHCP por puerto ens39
iptables -A INPUT -i ens33 -p udp -m multiport --dport 67,68 -j ACCEPT
iptables -A INPUT -i ens39 -p udp -m multiport --dport 67,68 -j ACCEPT
iptables -A OUTPUT -o ens39 -p udp -m multiport --dport 67,68 -j ACCEPT

#Permitiendo UDP
iptables -A INPUT -i ens33 -p udp -m multiport --dport 53,1812 -j ACCEPT
iptables -A OUTPUT -o ens33 -p udp -m multiport --dport 53,33434:33524,1812 -j ACCEPT

iptables -A INPUT -i ens38 -p tcp -m multiport --dport 20,21 -j ACCEPT
iptables -A OUTPUT -o ens38 -p tcp -m multiport --dport 20,21 -j ACCEPT

#Permitiendo TCP
iptables -A INPUT -i ens33 -p tcp -m multiport --dport 80,443 -j ACCEPT
iptables -A OUTPUT -o ens33 -p tcp -m multiport --dport 80,443 -j ACCEPT
```

- Habilitar el enmascaramiento desde las redes internas (LAN, DMZ) al exterior. Las redes irán enmascaradas con la ip de la red intermedia del proxy.

```
#Enmascarar salida hacia la WAN
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ens33 -j SNAT --to 192.168.3.2
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ens33 -j SNAT --to 192.168.3.2
```

- Lo siguiente que haremos será realizar los FORWARD en el firewall, para que los hosts de intranet se les permita accesos al exterior en los puertos http (80), https (443), ssh (22), dns (53), icmp (realizado en el apartado anterior), openfire (5222,7777,9090,9091), ftp (20,21), traceroute (33434:33524, puerto para pruebas), mysql (3306).a todos los equipos desde la red interna a la red externa. Habrá que permitir las respuestas de FORWARD es decir las que proceden de internet a intranet.

```
#FORWARD de DMZ/WAN. PROTOCOLO TCP/UDP
iptables -A FORWARD -i ens38 -o ens33 -p tcp -m multiport --dport 22,53,80,443,5222,7777,9090,9091 -j ACCEPT
iptables -A FORWARD -i ens38 -o ens33 -p udp -m multiport --dport 53,33434:33524 -j ACCEPT
iptables -A FORWARD -i ens33 -o ens38 -p tcp -m multiport --dport 20,21,22,53,80,443,40000:40100,5222,7777,9090,9091 -j ACCEPT

#FORWARD TCP LAN/WAN
iptables -A FORWARD -i ens39 -o ens33 -p tcp -m multiport --dport 20,21,22,53,3306 -j ACCEPT

#FORWARD UDP LAN/WAN
iptables -A FORWARD -i ens39 -o ens33 -p udp -m multiport --dport 53,123,80,443,3306 -j ACCEPT

#FORWARD LAN/DMZ
iptables -A FORWARD -i ens39 -o ens38 -d 192.168.0.3 -p tcp -m multiport --dport 20,21,22,25,80,443,3306,5222,7777,9090,9091 -j ACCEPT
iptables -A FORWARD -i ens39 -o ens38 -d 192.168.0.2 -p tcp -m multiport --dport 20,21,22,25,80,443,3306,5222,7777,9090,9091 -j ACCEPT
```

- También pondremos reglas de PREROUTING para que el router sepa a qué dirección debe de mandar los paquetes de un puerto en concreto.

```
#PREROUTING
iptables -t nat -A PREROUTING -i ens33 -p tcp -m multiport --dport 80,443,9090,9091,5552,7777 -j DNAT --to 192.168.0.3
iptables -t nat -A PREROUTING -i ens33 -p tcp -m multiport --dport 20,21 -j DNAT --to 192.168.0.2
iptables -t nat -A PREROUTING -p tcp --dport 40000:40100 -j DNAT --to 192.168.0.2:40000-40100
```

- Por ultimo queremos entrar por ssh a los equipos de la DMZ pero con distintos puertos.

```
#PREROUTING SSH
iptables -t nat -A PREROUTING -p tcp --dport 2202 -i ens33 -j DNAT --to 192.168.0.2:22
iptables -t nat -A PREROUTING -p tcp --dport 2203 -i ens33 -j DNAT --to 192.168.0.3:22
```

Una vez hecho esto podremos hacer que se inicie las siguientes reglas poniendo este script en el rc.local por ejemplo.

NOTA: En este caso se ha hecho una optimización del script firewall.sh, los script creados serán adjuntados con este documento.

2.2. Nftables

Es un proyecto de netfilter que tiene como objetivo reemplazar el marco existente de tablas {ip, ip6, arp, eb}. Proporciona un nuevo marco de filtrado de paquetes, una nueva utilidad en el espacio de usuario (nft) y una capa de compatibilidad para las tablas {ip, ip6}. Utiliza los hooks existentes, el sistema de seguimiento de las conexiones, los componentes de la línea de espera de los paquetes de red del espacio de usuario y el subsistema de registro de netfilter.

a. Ventajas

- **Simplicidad en sintaxis**

El mayor cambio que le puede gustar es la simplicidad. Con iptables, tenemos que configurar cada regla y usar la sintaxis que se puede comparar con los comandos normales. Entonces ejecutamos iptables con -A INPUT -s 192.168.1.20 etc. Con nftables, tenemos una sintaxis mucho más simple, que se parece a BPF (Berkely Packet Filter). La sintaxis de nftables está inspirada en la sintaxis tcpdump. Esto significa líneas más cortas y menos repetición.

Ejemplo:

```
nft add rule inet traffic-filter input tcp dport { 22, 80, 443 } accept
```

- **Reglas combinadas**

El ejemplo anterior incluye otra gran mejora: reglas combinadas. Entonces, en lugar de repetir líneas para cada puerto, podemos combinarlas. Esto es útil para los puertos UDP / TCP y también para los tipos ICMP.

Ejemplo:

Configurar la tabla IPv6 y la cadena de entrada.

```
nft add table ip6 traffic-filter nft add chain ip6 traffic-filter input
```

Permitir varios paquetes ICMP IPv6

```
nft add rule ip6 traffic-filter input icmpv6 type { nd-neighbor-solicit, echo-request, nd-router-advert, nd-neighbor-advert } accept
```

- **Múltiples acciones**

Una regla puede contener múltiples acciones. Con iptables, esto significaría dividir las reglas y saltar a diferentes bloques.

- **Protocolos combinados**

Al igual que la opción de combinar varias acciones, nftables permite definir una regla que admitirá tanto IPv4 como IPv6. Mucho mejor que usar iptables y ip6tables y sincronizar reglas entre los dos.

- **Pares de valores concatenados**

Dentro de conjuntos y mapas, los campos se pueden combinar para una evaluación adicional. Por ejemplo, la combinación de una dirección IP con un número de puerto. En lugar de hacer reglas individuales, estos datos pueden colocarse en una matriz de datos y luego usarse.

```
nft add element traffic-filter dict { 192.168.0.1 : drop, 192.168.0.2 : accept }
```

- **Más flexibilidad**

Con iptables tiene varias cadenas bases predeterminadas. Con nftables siempre comienzas con una pizarra en blanco. Simplemente agregue lo que necesita, desde cadenas hasta reglas.

- **Exportación fácil de datos**

Para aquellos que desean almacenar la configuración, hay una opción de exportación disponible. Nftables admite la exportación en XML y salida JSON.

```
nft export json
```

b. Conversión de Iptables a nftables.

En el caso de Debian 9 deberemos de instalar la herramienta.

```
root@Proxy:/home/frodo# apt install nftables
```

También lo podemos habilitar al arranque.

```
root@Proxy:/home/frodo# systemctl enable nftables
Created symlink /etc/systemd/system/multi-user.target.wants/nftables.service → /lib/systemd/system/nftables.service.
```

Lo siguiente que haremos será instalar la herramienta que nos ayudara a cambiar las reglas iptables a nftables.

```
root@Proxy:/home/frodo# apt install iptables-nftables-compat
```

Una vez instalada solo deberemos ejecutar iptables-translate con la sentencia de iptables y podremos convertir una línea de iptables en nftables.

```
root@Proxy:/home/frodo# iptables-translate -A INPUT -i lo -j ACCEPT
nft add rule ip filter INPUT iifname lo counter accept
```

Ahora para convertir las reglas que tenemos en el fichero firewall.sh en nftables deberemos primero copiar el fichero original.

```
root@Proxy:/home/frodo# cp firewall.sh firewallori.sh
```

Una vez hecho esto ejecutaremos el comando sed para reemplazar la palabra iptables por iptables-translate.

```
root@Proxy:/home/frodo# sed -i 's/iptables/iptables-translate/g' "firewall.sh"
```

Ahora el contenido del archivo es el siguiente.

```

firewall.sh
firewall.sh
17 iptables-translate -P OUTPUT DROP
18 iptables-translate -P FORWARD DROP
19
20 # Respuestas para las conexiones establecidas
21 iptables-translate -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
22 iptables-translate -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
23 iptables-translate -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
24
25 #Permitiendo via localhost hacer cualquier cosa.
26 iptables-translate -A INPUT -i lo -j ACCEPT
27 iptables-translate -A OUTPUT -o lo -j ACCEPT
28
29 #Permitiendo paquetes icmp por todas las tarjetas tanto de entrada como de salida.
30 iptables-translate -A INPUT -p icmp -j ACCEPT
31 iptables-translate -A OUTPUT -p icmp -j ACCEPT
32 iptables-translate -A FORWARD -i ens38 -o ens33 -p icmp -j ACCEPT
33 iptables-translate -A FORWARD -i ens39 -o ens33 -p icmp -j ACCEPT
34 #iptables-translate -A FORWARD -i ens39 -o ens38 -p icmp -j ACCEPT
35
36 #Permitiendo las conexiones por ssh
37 iptables-translate -A INPUT -i ens33 -p tcp --dport 22 -j ACCEPT
38
39 iptables-translate -A FORWARD -i ens33 -d 192.168.0.2 -p tcp --dport 22 -j ACCEPT
40 iptables-translate -A FORWARD -i ens33 -d 192.168.0.3 -p tcp --dport 22 -j ACCEPT
41
42 #iptables-translate -A OUTPUT -o ens38 -p tcp --dport 22 -j ACCEPT
43 #iptables-translate -A OUTPUT -o ens39 -p tcp --dport 22 -j ACCEPT
44
45 #Permitiendo las conexiones DHCP por puerto ens38
46 iptables-translate -A INPUT -i ens38 -p udp -m multiport --dport 67,68 -j ACCEPT

```

Por ultimo solo tendremos que ejecutar el script firewall.sh y volcarlo en otro archivo que en este caso se llamará nftables.sh.

```
root@Proxy:/home/frodo# ./firewall.sh > nftables.sh
```

Comprobamos el contenido de dicho script.

```

nftables.sh
nftables.sh
1 Aplicando Reglas de Firewall
2 nft flush table ip filter
3 nft delete chain ip filter (null)
4 nft nft flush table ip nat
5 nft delete chain ip nat (null)
6 nft nft nft nft add rule ip filter INPUT ct state related,established counter accept
7 nft add rule ip filter OUTPUT ct state related,established counter accept
8 nft add rule ip filter FORWARD ct state related,established counter accept
9 nft add rule ip filter INPUT iifname lo counter accept
10 nft add rule ip filter OUTPUT oifname lo counter accept
11 nft add rule ip filter INPUT ip protocol icmp counter accept
12 nft add rule ip filter OUTPUT ip protocol icmp counter accept
13 nft add rule ip filter FORWARD iifname ens38 oifname ens33 ip protocol icmp counter accept
14 nft add rule ip filter FORWARD iifname ens39 oifname ens33 ip protocol icmp counter accept
15 nft add rule ip filter INPUT iifname ens33 tcp dport 22 counter accept
16 nft add rule ip filter FORWARD iifname ens33 ip daddr 192.168.0.2 tcp dport 22 counter accept
17 nft add rule ip filter FORWARD iifname ens33 ip daddr 192.168.0.3 tcp dport 22 counter accept
18 nft add rule ip filter INPUT iifname ens38 ip protocol udp udp dport { 67,68 } counter accept
19 nft add rule ip filter INPUT iifname ens39 ip protocol udp udp dport { 67,68 } counter accept
20 nft add rule ip filter INPUT iifname ens33 udp dport 53 counter accept
21 nft add rule ip filter OUTPUT oifname ens33 udp dport 53 counter accept
22 nft add rule ip filter INPUT iifname ens38 ip protocol tcp tcp dport { 20,21 } counter accept
23 nft add rule ip filter INPUT iifname ens33 ip protocol tcp tcp dport { 80,443 } counter accept
24 nft add rule ip filter OUTPUT oifname ens33 ip protocol tcp tcp dport { 80,443 } counter accept
25 nft add rule ip nat POSTROUTING oifname ens33 ip saddr 192.168.0.0 counter snat to 192.168.15.19
26 nft add rule ip nat POSTROUTING oifname ens33 ip saddr 192.168.1.0 counter snat to 192.168.15.19
27 nft add rule ip filter FORWARD iifname ens38 oifname ens33 ip protocol tcp tcp dport { 22,53,80,443,5222,7777,9090,9091 } counter accept
28 nft add rule ip filter FORWARD iifname ens38 oifname ens33 ip protocol udp udp dport { 53,33434-33524 } counter accept
29 nft add rule ip filter FORWARD iifname ens33 oifname ens38 ip protocol tcp tcp dport { 20,21,22,53,80,443,40000-40100,5222,7777,9090,9091 } counter accept
30 nft add rule ip filter FORWARD iifname ens39 oifname ens33 ip protocol tcp tcp dport { 20,21,22,53,80,443,3306 } counter accept
31 nft add rule ip filter FORWARD iifname ens39 oifname ens33 ip protocol udp udp dport { 53,123,3306 } counter accept
32 nft add rule ip filter FORWARD iifname ens39 oifname ens38 ip protocol tcp ip daddr 192.168.0.3 tcp dport { 20,21,22,25,80,443,3306,5222,7777,9090,9091 } counter accept
33 nft add rule ip filter FORWARD iifname ens39 oifname ens38 ip protocol tcp ip daddr 192.168.0.2 tcp dport { 20,21,22,25,80,443,3306,5222,7777,9090,9091 } counter accept
34 nft add rule ip nat PREROUTING iifname ens33 ip protocol tcp tcp dport { 80,443,9090,9091,5552,7777 } counter dnat to 192.168.0.3
35 nft add rule ip nat PREROUTING iifname ens33 ip protocol tcp tcp dport { 20,21 } counter dnat to 192.168.0.2
36 nft add rule ip nat PREROUTING tcp dport 40000-40100 counter dnat to 192.168.0.2:40000-40100
37 nft add rule ip nat PREROUTING iifname ens33 tcp dport 2202 counter dnat to 192.168.0.2:22
38 nft add rule ip nat PREROUTING iifname ens33 tcp dport 2203 counter dnat to 192.168.0.3:22
39 OK. Verifique la configuración de iptables-translate con IPTABLES -L -n

```

Con esto ya se habrían cambiado las reglas de iptables por nftables.

3. SQUID (Proxy, proxy caché).

3.1. Introducción

Un servidor proxy se utiliza para centralizar en ellas comunicaciones de una red local con otras redes o equipos. Inicialmente la comunicación con proxy era una de las formas posibles de establecer comunicaciones con una red pública desde una red privada, aunque actualmente es más habitual hacer esto a través de NAT. El funcionamiento de un servidor proxy es tal que los clientes de la red local en lugar de realizar las peticiones directamente a la red externa, se la solicitan al servidor proxy, éste establece la conexión con el equipo externo y después devuelve la petición al cliente inicial. Esto permite utilizar un proxy cuando se quiere:

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

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

3.2. Conceptos sobre cachés

Los servidores que actúan de proxy-caché se pueden configurar de varias formas, la forma más simple es un solo servidor proxy-caché en la red1 en el que todos los ordenadores pertenecientes a esa red accederán a este servidor, que será el que almacenará todos los datos. Cuando un usuario solicita al servidor una página, éste comprueba si fue actualizada desde que fue almacenada. Si tiene la versión actualizada ahorra al usuario final la descarga de la misma proporcionándosela directamente.

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

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

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

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

3.3. Instalación

Para instalar Squid hay que utilizar simplemente:

```
root@Proxy:/home/frodo# apt install squid3
```

Los ficheros y directorios más importantes son:

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

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

```
root@Proxy:/home/frodo# netstat -putan | grep squid
tcp6      0      0 :::3128          :::*              LISTEN       47387/(squid-1)
udp       0      0 0.0.0.0:46009    0.0.0.0:*         47387/(squid-1)
udp6      0      0 :::39229         :::*              47387/(squid-1)
udp6      0      0 ::1:51254        ::1:39759         ESTABLISHED 47387/(squid-1)
```

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

3.4. Configuración elemental

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

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


```

root@Proxy:/home/frodo# cat /etc/squid/squid.conf | grep -v ^$ | grep -v ^#
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
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

```

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

3.5. Recursos dedicados a Squid

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

- a) Si quisiéramos asignar 256MB de memoria RAM para la caché en RAM de Squid, deberíamos descomentar y modificar la siguiente línea:

```
# cache_mem 256 MB
```

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

```
# maximum_object_size_in_memory 512 KB
```

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

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

El formato genérico de esta directiva es:

`cache_dir` tipo directorio Mbytes L1 L2 [options]

- Tipo. Tipo de sistema de almacenamiento a utilizar (ufs es el único que está definido por defecto en la instalación).
 - Directorio. Ruta del directorio que se va a utilizar para guardar los datos del caché.
 - Mbytes. Cantidad de espacio en disco en Megabytes que se va a utilizar para el caché. Si queremos que utilice el disco entero es recomendable poner aquí un 20% menos del tamaño.
 - L1. Número de subdirectorios de primer nivel que serán creados bajo directorio.
 - L2. Número de subdirectorios de segundo nivel que serán creados bajo cada subdirectorio de primer nivel.
- d) Podemos controlar que peticiones se guardan en la cache y cuales no se guardan en la cache mediante la directiva `cache`. En el siguiente ejemplo estamos diciendo al squid que los dominios (.politecnicomalaga.com, .uma.es) que no se cacheen (por el motivo que sea), que las peticiones a .ubuntu.com sí que se haga. Al poner `cache deny all` al final podríamos haber omitido la línea (`cache deny DominiosNoCachear`).

```
acl DominiosNoCachear dstdomain .politecnicomalaga.com .uma.es .elpais.es
acl DominiosCachear dstdomain .windowsupdate.com .ubuntu.com
cache deny DominiosNoCachear
cache allow DominiosACachear
cache deny all
```

3.6. Control de acceso

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

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

`acl` nombre ACL tipo ACL cadena ...

`acl` nombre ACL tipo ACL "fichero"

Donde:

- Nombre ACL es el nombre que corresponde a esta definición ACL.

- Tipo ACL es el tipo de elemento contenido en esta definición.
- cadena o fichero es el argumento apropiado al tipo ACL, pudiendo haber más de un argumento en la lista.

Vamos a analizar las ACL inicialmente definidas:

```
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
```

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

```
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost manager
http_access deny manager
http_access allow localhost
http_access deny all
```

- No permite conexiones a puertos no definidos en la acl Safe ports.
- No permite conexiones directas (sin cachear), salvo a los puertos definidos en la acl SSL ports.
- Permite acceder a squid desde la dirección IP 127.0.0.1.
- No permite acceso a squid en cualquier otra situación.

3.6.1. Permitir acceso a nuestra red local

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

En nuestro caso podríamos dejar siguientes líneas:

```
acl interna src 192.168.0.0/24
acl interna src 192.168.1.1/24
acl interna src 192.168.3.0/24
acl localhost src 127.0.0.1
```

Y posteriormente permitir acceso a *la red interna* con la siguiente directiva.

```
http_access allow localhost
http_access allow interna
```

3.6.2. Restricciones de acceso

Una vez que podemos acceder a otras redes desde la red local utilizando el proxy, podemos establecer restricciones a nuestros usuarios en función de la dirección o dominio destino, la hora, el día, el tipo de fichero solicitado, etc.

A continuación mostraremos algunos ejemplos con las ACL más significativas.

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

```
acl host srcdomain LEIA
```

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

```
acl instagram dstdomain instagram.com
```

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

```
acl inapropiados dstdom_regex "/etc/squid/dominios_inapropiados.txt"
```

- **Control por día y hora.** Permite controlar el acceso en función de la hora o el día.

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

- **Expresión regular que concuerda con la URL.** Permite establecer reglas en función de expresiones regulares que aparezca en la URL de la petición. Se busca en la url solicitada las palabras (filtros) en la url que solicita el cliente para denegar o aceptar.

```
acl BlackUrl url_regex -i facebook diario elpais
```

Para más información sobre las acls: <https://wiki.squid-cache.org/SquidFaq/SquidAcl>

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

```
http_access deny all
http_access allow redinterna
```

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

Las ACL son especialmente útiles cuando queremos prohibir el acceso a una lista de sitios inapropiados. Squid no está optimizado para gestionar una larga lista de sitios, pero puede gestionar un número concreto de sitios sin problemas.

```
acl porno dst_regex \.playboy.com \.sex.com \.conejitas.com # acl URL Inapropiadas
```

```
http_access deny porno
http_access allow localhost
http_access allow redinterna
http_access deny all
```

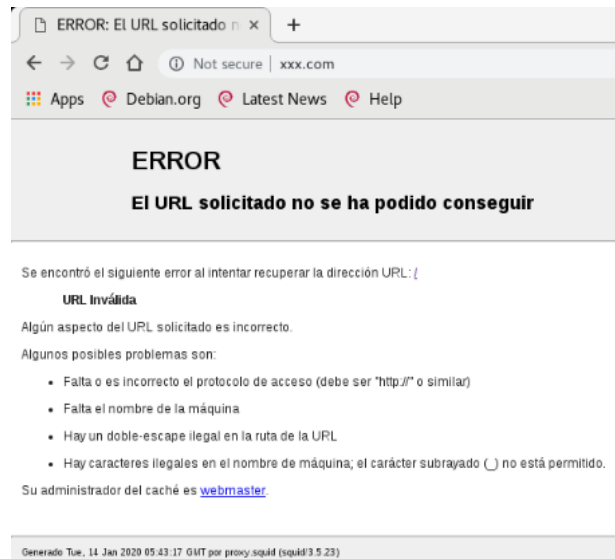
En este caso las direcciones que serán consideradas como inadecuadas son las que tienen los dominios establecidos en la acl. Estas URL tendrán filtrado el acceso y no será posible acceder a ellas, tal como indica la directiva **http_access deny porno**.

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

```
acl inapropiados dstdom_regex "/etc/squid/dominios_inapropiados.txt"
```

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

Comprobaremos que ha sido definida la acl correctamente desde un cliente de la LAN.



Captura del archivo que ha sido utilizado en la práctica.

```
http_port 192.168.1.1:3128 intercept

visible_hostname proxy.squid

acl redinterna src 192.168.0.0/24
acl redinterna src 192.168.1.0/24
acl redinterna src 192.168.3.0/24
acl localhost src 127.0.0.1

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 280     # http-mgmt
acl Safe_ports port 488     # gss-http
acl Safe_ports port 777     # multiling http
acl CONNECT method CONNECT

acl BlackUrl url_regex -i facebook diario elpais
acl host srcdomain LEIA
acl inapropiados dstdom_regex "/etc/squid/dominios_inapropiados.txt"

http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost manager
http_access deny manager

http_access deny BlackUrl
http_access deny host
http_access deny inapropiados
http_access allow localhost
http_access allow redinterna
http_access deny all

cache_mem 256 MB

maximum_object_size_in_memory 512 KB

maximum_object_size 4 MB

cache_dir ufs /var/spool/squid 100 16 256

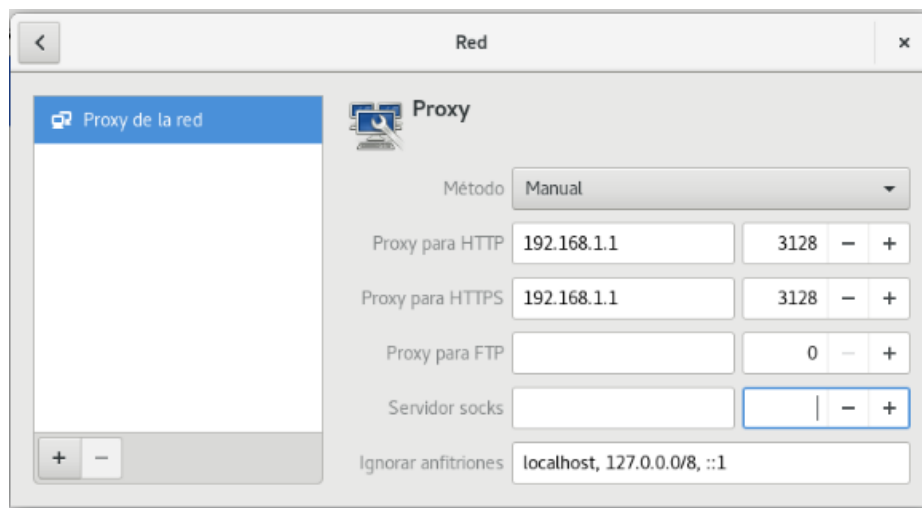
error_directory /usr/share/squid/errors/Spanish
access_log daemon:/var/log/squid/access.log squid
cache_log /var/log/squid/cache.log
```

3.6.3. Configuración de los clientes.

El cliente de la red local lo deberemos de configurar si no tenemos el proxy en modo transparent que se explicará en la sección siguiente. En este caso lo explicare para el navegador Chromium pero más o menos en los demás navegadores es igual.

Para ello seccionaremos **Settings** → **Open proxy settings** → **Metodo** → **Avanzado**.

En esta ventana ponemos la dirección del servidor proxy y el puerto 3128.



Si trabajamos en modo consola y deseamos definir la variable de entorno `http_proxy`, podemos usar el siguiente comando:

```
root@leia:/home/frodo# export http_proxy="http://192.168.1.1:3128"
```

3.6.4. Proxy transparente

Existe la posibilidad de configurar Squid para que todos los usuarios de una red naveguen a través del proxy sin necesidad de tener que configurar cada una de sus aplicaciones, esto es lo que se denomina proxy transparente. Es más, se hace para que naveguen a través del proxy sin saberlo o sin quererlo, asumiendo todas las restricciones que éste imponga.

En primer lugar hay que utilizar la siguiente línea de iptables, que hará que todo el tráfico con destino al puerto 80 se redirija al 3128:

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

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

```
http_port 3128 intercept
```

En las versiones anteriores se sustituía `intercept` por `transparent`.

```
http_port 3128 transparent
```

3.7. HTTPS (Docker)

Por defecto la versión de squid 3 no está disponible HTTPS por lo que en este apartado se ha optado por instalar docker con un contenedor que tenga squid para permitir el descifrado SSL y el filtrado HTTPS.

CONFIGURAR EL REPOSITORIO

Comenzaremos por instalar docker en la máquina, para ello primero tendremos que añadir y configurar algunos repositorios.

Primero actualizamos los repositorios.

```
root@Proxy:/home/frodo# apt update
```

Instalaremos los siguientes paquetes para permitir el uso de un repositorio sobre HTTPS.

```
apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common
```

A continuación agregaremos la clave oficial de docker.

```
root@Proxy:/home/frodo# curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
OK
```

Usaremos el siguiente comando para configurar el repositorio estable.

```
add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) \
    stable"
```

INSTALAR DOCKER ENGINE

Actualizaremos los repositorios.

```
root@Proxy:/home/frodo# apt update
```

Seguidamente instalaremos la última versión de docker

```
root@Proxy:/home/frodo# apt-get install -y docker-ce docker-ce-cli containerd.io
```

Comprobaremos que docker ha sido correctamente instalado.

```
root@Proxy:/home/frodo# docker -v
Docker version 19.03.5, build 633a0ea838
```


Lo siguiente que haremos será instalar el paquete git para poder hacer un gitclone del repositorio github.

```
root@Proxy:/home/frodo# apt install git
```

```
root@Proxy:/home/frodo# git clone https://github.com/diladele/docker-websafety
Cloning into 'docker-websafety'...
remote: Enumerating objects: 510, done.
remote: Total 510 (delta 0), reused 0 (delta 0), pack-reused 510
Receiving objects: 100% (510/510), 89.54 KiB | 0 bytes/s, done.
Resolving deltas: 100% (205/205), done.
```

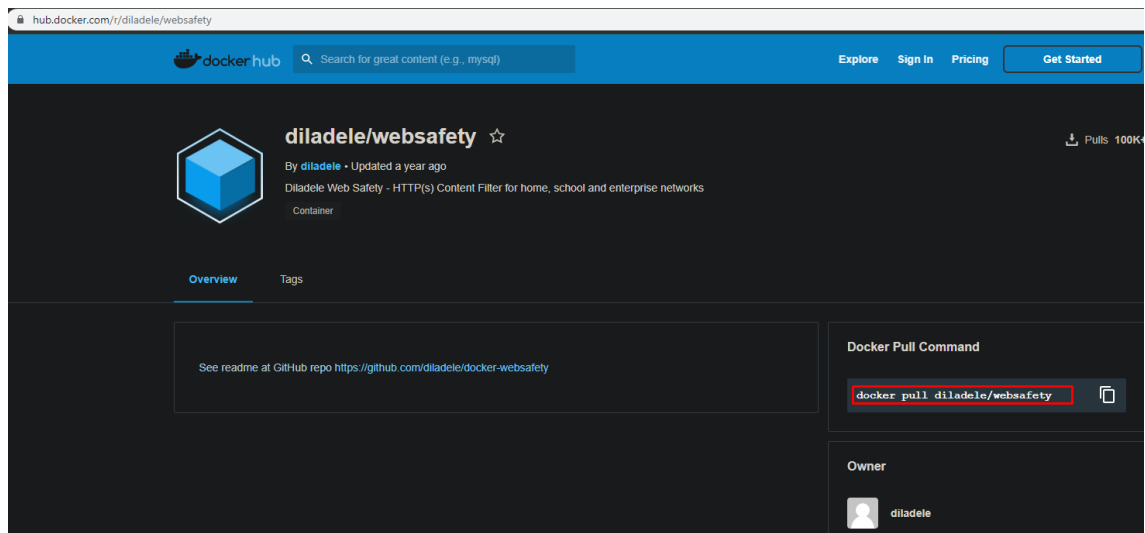
Nos habremos descargado una carpeta con el dockerfile y unos scripts.

```
root@Proxy:/home/frodo# ls docker-websafety/
README.md  src
root@Proxy:/home/frodo# ls docker-websafety/src/
build.sh  contents  Dockerfile  run.sh
```

Una vez descargado lo siguiente deberemos de bajarnos la imagen de diladele de Docker Hub para ello nos iremos a la siguiente URL.


<https://hub.docker.com/r/diladele/websafety>

La siguiente página nos dirá que comando utilizar para poder bajarnos la imagen de Docker Hub.



hub.docker.com/r/diladele/websafety

dockerhub Search for great content (e.g., mysql) Explore Sign In Pricing Get Started


 **diladele/websafety** ☆ Pulls 100K

By diladele • Updated a year ago
Diladele Web Safety - HTTP(s) Content Filter for home, school and enterprise networks
Container

Overview Tags

See readme at GitHub repo <https://github.com/diladele/docker-websafety>

Docker Pull Command
`docker pull diladele/websafety`

Owner
 diladele

Una vez ejecutado ese comando en nuestro equipo ya tendremos la imagen de diladele/websafety descargada.

```

root@Proxy:/home/frodo/docker-websafety/src# docker pull diladele/websafety:7.0
7.0: Pulling from diladele/websafety
c64513b74145: Already exists
01b8b12bad90: Already exists
c5d85cf7a05f: Already exists
b6b268720157: Already exists
e12192999ff1: Already exists
d39ece66b667: Already exists
65599be66378: Already exists
9c6c60f7fc45: Pull complete
0614f09c38d5: Pull complete
fde81bea4770: Pull complete
2d46711fec91: Pull complete
4df0b1d58a4f: Pull complete
818292b319a5: Pull complete
fa7cb3d2ee4f: Pull complete
98fb6a87dcfe: Pull complete
d9986828ad3b: Pull complete
fbafcb4310a0: Pull complete
6cf3e27577cc: Pull complete
6c00042042cc: Pull complete
e52282e48049: Pull complete
12d345060e71: Pull complete
f390d5494468: Pull complete
bd37ddc65652: Pull complete
ed4a5df3d6c7: Pull complete
72f39e1ac416: Pull complete
6a584dcf90a2: Pull complete
1a6c3e9834fc: Pull complete
d3ee127c1ad4: Pull complete
Digest: sha256:a512eb58ae878838e399834c38b14f95d0680ce905e2eeec9ced47c12b4316b2
Status: Downloaded newer image for diladele/websafety:7.0
docker.io/diladele/websafety:7.0

```

Ya solo quedaría crear el contenedor, para ello podremos ejecutar el scripts que está dentro de docker-websafety.

```

GNU nano 2.7.4                                Fichero: docker-websafety/src/run.sh
docker run -it --name websafety-config diladele/websafety:7.0 /usr/local/bin/firstrun.sh
docker run -dt --name websafety --dns=8.8.8.8 --volumes-from websafety-config -p 8000:80 -p 3128:3128 diladele/websafety:7.0

```

Ahora ejecutaremos el script run.sh para que nos cree el contenedor.

```

root@Proxy:/home/frodo/docker-websafety/src# ./run.sh
2f2feb097b26da8ed14b4007ffae6bbadde72b6e090e4b42e429357fe3af1019

```

Ahora comprobaremos que el contenedor esta corriendo con el siguiente comando.

```

root@Proxy:/home/frodo/docker-websafety/src# docker container ls

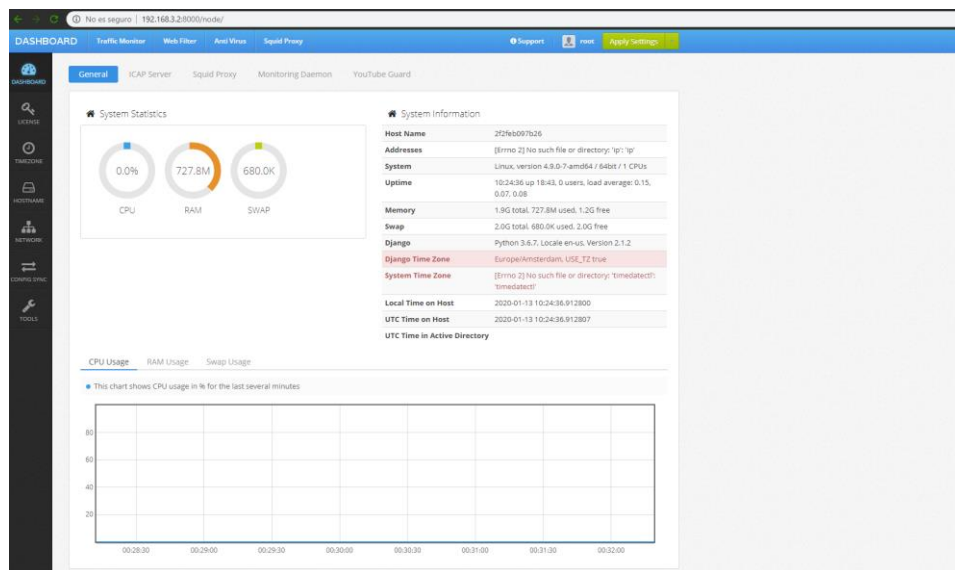
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2f2feb097b26	diladele/websafety:7.0	"/sbin/my_init"	About a minute ago	Up About a minute	0.0.0.0:3128->3128/tcp, 0.0.0.0:8000->80/tcp	websafety

Veremos que se han creados dos reglas iptables, una es para redirigir todo lo que le venga al puerto 3128 de la máquina que se lo redirija al 3128 del contenedor y la otra es que todo lo que venga del puerto 8000 que es el que utiliza el dashboard lo redirija al 80.

pkts	bytes	target	prot	opt	in	out	source	destination	
0	0	ACCEPT	tcp	--	!docker0	docker0	0.0.0.0/0	172.17.0.2	tcp dpt:80
0	0	ACCEPT	tcp	--	!docker0	docker0	0.0.0.0/0	172.17.0.2	tcp dpt:3128

Ahora pondremos la IP de nuestra máquina y el puerto 8000 y podremos acceder al panel web.



Para poder hacer ACLs nos iremos **Squid Proxy** → **SETTINGS** → **Default ACLs**.

The screenshot shows the 'Default ACLs' configuration page in the Squid Proxy interface. The 'SETTINGS' icon in the sidebar is highlighted. The 'Default ACLs' tab is selected in the top navigation bar.

Allowed Subnets:

```
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
fc00::/7
fe80::/10
```

Specify here the subnets in CIDR format that are allowed to use this proxy. For example 10.0.0.0/8. Each subnet must be on a separate line. These subnets will form the *localnet* ACL in squid.conf.

Prohibited Subnets:

Specify here the subnets in CIDR format that are prohibited from using this proxy. For example 10.0.0.0/8. Each subnet must be on a separate line.

Additional SSL Ports:

Specify here the list of additional ports to allow establishing CONNECT tunnels to. Default ports are 443. Ports must be separated by space.

Additional Safe Ports:

Specify here the list of additional ports to allow to connect to. Default ports are 21 70 80 210 280 443 488 591 777 1025-65535. Ports must be separated by space.

Advanced ACLs:

```
acl inapropiados dst dom_regex \.playboy.com \.sex.com \.xxx.com
```

Specify here the advanced acls you would like to put into configuration file. Lines are written to configuration file as-is so please be careful.

Advanced http_access:

```
http_access deny inapropiados
```

Una vez definidas las ACLs deberemos de reiniciar squid e ICAP.

Support root **Apply Settings**

Miscellaneous

DASHBOARD Traffic Monitor Web Filter Anti Virus Squid Proxy

Apply Configuration Settings and restart ICAP daemon and Squid proxy if required

Save And Restart

- All changes from Web UI are saved to configuration storage.
- ICAP daemon (wsicapd) is unconditionally restarted.
- Squid Proxy is unconditionally restarted.
- All active HTTP and ICAP connections are terminated, browsers may need a refresh of all opened pages.
- New configuration settings are applied immediately.

Restart Squid Proxy & ICAP Server

Save And Reload

- All changes from Web UI are saved to configuration storage.
- ICAP daemon (wsicapd) is reloaded using standard HUP message.
- Squid Proxy is reloaded using standard HUP message.
- New connections from browsers may not pick up configuration changes if they are serviced by persistent ICAP connections with old configuration.
- New configuration settings are applied to new ICAP connections only.

Reload Squid Proxy & ICAP Server

Comprobaremos que se ha reiniciado correctamente.

DASHBOARD Traffic Monitor Web Filter Anti Virus Squid Proxy

Restart Successful

Restart of Squid Proxy and/or Diladele Web Safety ICAP completed successfully. The following log shows command executed.

Exit code - 0

No exceptions raised.

STDOUT

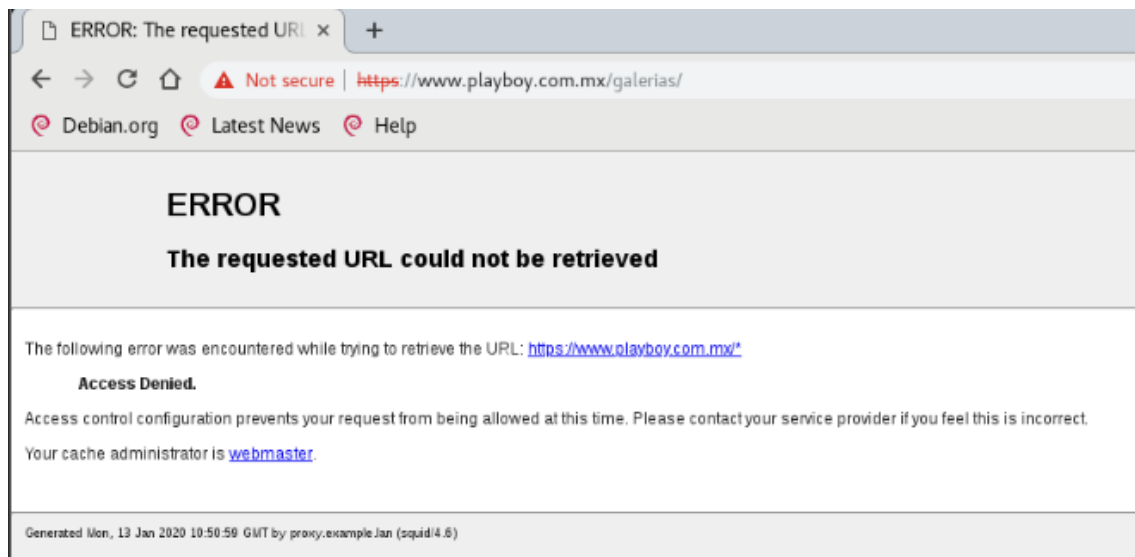
```
Restarting Web Safety ICAP Daemon...
ok: run: wsicap: (pid 1462) is
Restarting Web Safety Monitoring Daemon...
ok: run: wsmgr: (pid 1475) 0s
Restarting Web Safety G5B Daemon...
ok: run: wsgsb: (pid 1482) 1s
Reloading Squid Proxy Server...
ok: run: squid: (pid 1489) 0s
Reload successful!
```

STDERR

STDERR stream is empty.

Web Safety for Squid Proxy, version 7.0.0.7A5E-amd64-linux-ubuntu18, (c) Diladele B.V., 2018. On Top

Ahora nos iremos al cliente y pondremos una url que hemos denegado y vemos como la ha bloqueado.



Web safety aparte de traer como proxy squid, también trae un monitorizador para el proxy squid.

Esta ventana muestra el historial de las últimas peticiones hechas, te da información como la página web que ha sido solicitada, la IP que la ha solicitado y el día y la hora.

Browsing History												
Time	Incident	Policy	User Name	User IP	Host	Category	Verdict	Level	Module	Param 1	Scan Size	
13-01-2020 11:50:26	13	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:50:26	4	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:50:19	2	default	-	192.168.1.2	http://xxx		block	headers	http_sanitation	xxx	0	0
13-01-2020 11:49:44	202	default	-	192.168.1.2	connect://www.videospornogratix.net:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:44:44	196	default	-	192.168.1.2	connect://www.videospornogratix.net:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:43:44	201	default	-	192.168.1.2	connect://www.videospornogratix.net:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:42:52	117	default	-	192.168.1.2	connect://www.videospornogratix.net:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:42:47	132	default	-	192.168.1.2	connect://www.videospornogratix.net:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:42:47	118	default	-	192.168.1.2	connect://www.videospornogratix.net:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:58	22	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:58	21	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:53	30	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:53	19	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:53	18	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:53	11	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:53	10	default	-	192.168.1.2	connect://www.pornogratisdiano.com:443		block	headers	adult_heuristics	200	0	0
13-01-2020 11:41:50	7	default	-	192.168.1.2	http://xxx		block	headers	http_sanitation	xxx	0	0
13-01-2020 11:37:27	159	default	-	192.168.1.2	http://xxx		block	headers	http_sanitation	xxx	0	0
13-01-2020 11:35:42	35	default	-	192.168.1.2	http://xxx		block	headers	http_sanitation	xxx	0	0

Para más información visiten esta web: <https://docs.diladele.com/index.html>

3.8. SARG

Sarg (Squid Analysis Report Generator) es una herramienta que permite a los administradores de sistemas ver de una manera sencilla y amigable que sitios de Internet visitan (incluso se puede saber hasta la hora en que la visitó) los usuarios de la red local usando los logs de Squid. Genera una lista diaria, semanal, mensual o personalizada con los sitios de Internet que visita cada usuario, cuanto consumió (MB), etc.

Para poder instalar Sarg en Debian solo hay que descargarlo e instalarlo de los repositorios.

```
root@Proxy:/home/frodo# apt install sarg
```

Para configurar sarg se utiliza el fichero de configuración sarg.conf, que se encuentra en **“/etc/sarg/sarg.conf”**.

Copiaremos el fichero de configuración original de sarg y haremos otro a nuestro gusto.

```
GNU nano 2.7.4                                Fichero: /etc/sarg/sarg.conf
lastlog 10
access_log /var/log/squid/access.log
output_dir /var/www/squid-reports/Manual
date_format e
index yes
topsites_num 20
```

- **La primera** opción nos permite decidir cuantos informes se guardaran (10 informes) y cuando llegue a ese valor se borrarán los informes más antiguos, si especificamos el valor cero se guardaran todos los informes
- **La segunda** línea especifica donde se almacenan el fichero log de Squid, sarg utiliza este fichero para generar los diferentes informes, con lo que hay que indicar donde se encuentra.
- **La tercera** línea indica la ruta donde se almacenaran los informes que se generen de forma manual.
- **La cuarta** línea escoge el formato horario de fecha en los reportes; e formato Europeo (dd/mm/yy), u americano (mm/dd/yy) y w semanal (ww.yy).
- **La quinta** línea indica que se genera un fichero index.html
- **La sexta** línea indica que se crea un informe donde aparezcan las 20 web más visitadas.

Como sarg genera los informes mediante una página web, para ver los informes debemos configurar apache.

Crearemos un virtual host y añadiremos el siguiente contenido.

```
Alias "/sarg" "/var/www/squid-reports/Manual/"
<Directory /var/www/squid-reports/Manual/>
    Options FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Para comprobar que todo funciona ejecutaremos el comando sarg el cual nos generará un informe de forma manual en Sarg.

Este comando lo podremos meter en un cron y hacer que haga cada x tiempo un nuevo informe.

```
root@Proxy:/home/frodo# sarg
```

Ahora comprobaremos que el index se ha creado correctamente.


```
root@Proxy:/home/frodo# nano /var/www/squid-reports/Manual/
14Jan2020-14Jan2020/ images/ index.html
```

Para poder acceder a los informes abrimos un navegador y escribimos la siguiente dirección.

https://ip_servidor/sarg

Aquí podremos ver todos los informes detalladamente y más amigablemente.

192.168.3.2/sarg/



SARG Squid Analysis Report Generator

Squid User Access Report

FILE/PERIOD	CREATION DATE	USERS	BYTES	AVERAGE
14Jan2020-14Jan2020	mar 14 ene 2020 16:39:57 CET	2	62.52M	31.26M

Generated by sarg-2.3.10 Apr-12-2015 on 14/ene/2020-16:39



Sarg tiene diferentes opciones entre las cuales top users, nos detalla los hosts que han realizado peticiones y parámetro de consumo de recursos.


SARG Squid Analysis Report Generator

Squid User Access Report

Period: 14 ene 2020
 Sort: bytes, reverse
Top users

[Top sites](#)
[Sites & Users](#)
[Denied accesses](#)

NUM		USERID	CONNECT	BYTES	%BYTES	IN-CACHE-OUT	ELAPSED TIME	MILLISEC	%TIME
1		192.168.1.2	56	62.51M	99,98%	0,33% 99,67%	00:03:57	237.724	99,94%
2		192.168.3.200	5	11.28K	0,02%	0,00% 100,00%	00:00:00	150	0,06%
TOTAL			61	62.52M		0,33% 99,67%	00:03:57	237.874	
AVERAGE			30	31.26M			00:01:58	118.937	

Generated by sarg-2.3.10 Apr-12-2015 on 14/ene/2020-16:39

En la opción Top 20 encontraremos la 20 páginas más visitadas por los equipos de la LAN.



Squid Analysis Report Generator

Squid User Access Report

Period: 14 ene 2020

Top 20 sites

NUM	ACCESSED SITE	CONNECT	BYTES	TIME	USERS
1	security.debian.org	31	62.16M	0:03:55	1
2	ftp.es.debian.org	9	277.09K	0	1
3	192.168.3.2	5	11.28K	0	1
4	proxy.squid:3128	4	16.96K	0	1
5	www.xxx.com	3	1.02K	0	1
6	xxx.com	3	1.00K	0	1
7	r3---sn-2gvjvov-5b2e.gvt1.com	1	21.32K	0	1
8	metadata.ftp-master.debian.org	1	19.51K	0	1
9	extensions.gnome.org:443	1	3.71K	0	1
10	odrs.gnome.org:443	1	3.69K	0	1
11	redirector.gvt1.com	1	1.14K	0	1
12	hot.com	1	631	0	1

Generated by sarg-2.3.10 Apr-12-2015 on 14/ene/2020-16:39

En la opción Denied podremos ver las páginas web que han sido denegadas y que clientes han sido los que la han solicitado.



Squid Analysis Report Generator

Squid User Access Report

Period: 14 ene 2020

Denied

USERID	IP/NAME	DATE/TIME	ACCESSED SITE
192.168.1.2	192.168.1.2	14/01/20-06:41:40	http://proxy.squid:3128
		14/01/20-06:41:51	http://proxy.squid:3128
		14/01/20-06:43:17	http://proxy.squid:3128
		14/01/20-07:00:26	http://proxy.squid:3128

Generated by sarg-2.3.10 Apr-12-2015 on 14/ene/2020-16:39

4. Filtrador de contenido (DansGuardian).

Cuando se instala Squid, podemos controlar las páginas web que se conectan los usuarios de una red local, podemos bloquear ciertas páginas web que no nos interesa que los usuarios puedan acceder. Cuando se requiere un control más exhaustivo del contenido web, la configuración de Squid puede ser más compleja e ineficiente.

DansGuardian es un programa que actúa como filtro de contenido web, estos filtros pueden seguir varios criterios como; URL, palabras o frases contenidas en una página web, tipos de ficheros.

Para instalar DansGuardian solo debemos de ejecutar el siguiente comando:

```
root@Proxy:/home/frodo# apt install dansguardians
```


En nuestro servidor Debian el tráfico se escucha por el puerto 80 esta re direccionado al puerto 3128 donde se encuentra Squid a la escucha. DansGuardian escucha por el puerto 8080, con lo que deberemos re direccionar el tráfico del puerto 80 al 8080, para que DansGuardian puede trabajar como filtro de contenido web, y borrar la regla que re direcciona el puerto 80 al 3128.

```
iptables -t nat -A PREROUTING -i ens39 -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Para poder configurar Dansguardian deberemos de ir y configurar estas opciones.

El fichero de configuración es el siguiente:

```
root@Proxy:/home/frodo# nano /etc/dansguardian/dansguardian.conf
```

En podremos configurar las siguientes líneas.

Esta opción sirve para configurar el fichero.log.

```
loglocation = '/var/log/dansguardian/access.log'
```

Especifica que los mensajes de DansGuardian aparezcan en español, por ejemplo páginas de acceso denegado.

```
language = 'spanish'
```

Indica la IP que DansGuardian aplicara el filtrado, si esta en blanco se aplicara a todas las direcciones IP.

```
filterip = 192.168.1.1
```

Puerto por donde escucha DansGuardian.

```
filterport = 8080
```

Ip por donde escucha el proxy.

```
proxyip = 192.168.1.1
```

Puerto por el que escucha el proxy.

```
proxyport = 3128
```

Ahora procederemos a aplicar algunos filtros, los filtros son guardados en ficheros y son aplicados en el fichero de configuración de DansGuardian.

La ruta de los ficheros es la siguiente:

```

root@Proxy:/home/frodo# ls /etc/dansguardian/lists/
authplugins          blacklists            exceptionphraselist    logsitelist
bannedextensionlist  contentregexplist    exceptionregexpurllist logurllist
banneddiplist        contentscanners       exceptionsitelist     phraselists
bannedmimetyplist    downloadmanagers      exceptionurllist       pics
bannedphraselist     exceptionextensionlist filtergroupslist       urlregexplist
bannedregexheaderlist exceptionfilesitelist  greysitelist          weightedphraselist
bannedregexpurllist  exceptionfileurllist  greyurllist
bannedsitelist        exceptioniplist        headerregexplist
bannedurllist         exceptionmimetyplist   logregexpurllist

```

Por ejemplo vamos a aplicar el filtro de bannedphraselist.

Para ello primero rellenaremos el filtro con algunas palabras y frases prohibidas.

```

<sex>, <xxx>, <mandanga>
#listcategory: "Banned Phrases"

# The following banned phraselists enable Website Content Labeling systems. These

.Include</etc/dansguardian/lists/phraselists/safelabel/banned>
#.Include</etc/dansguardian/lists/phraselists/rta/banned_portuguese>

# The following banned phraselists are included in the default DG distribution.

.Include</etc/dansguardian/lists/phraselists/pornography/banned>
##.Include</etc/dansguardian/lists/phraselists/pornography/banned_portuguese>
#.Include</etc/dansguardian/lists/phraselists/illegaldrugs/banned>

```

Además de poder definir las palabras que quieras filtrar DansGuardian ya trae listas definidas las cuales puedes usar. Como por ejemplo la siguiente.

```

root@Proxy:/etc/dansguardian# nano lists/phraselists/pornography/
banned          weighted_dutch      weighted_malay      weighted_spanish
banned_portuguese weighted_french      weighted_norwegian  weighted_swedish
weighted        weighted_german      weighted_polish
weighted_chinese weighted_italian     weighted_portuguese
weighted_danish weighted_japanese    weighted_russian

```

Por ejemplo mostraremos el contenido de weighted_spanish.

```

GNU nano 2.7.4          Fichero: lists/phraselists/pornography/weighted_spanish
#
# Originally Created By Fernand Jonker
# I highly recommend that you also enable the Portuguese list.
#
# Looking for volunteer contributor ..... please contact phrasemaster@dansguardian.org
#
# Test Sites
# http://www.galeriasprivadas.com/
# http://www.bancodeputas.com/
# http://pornografia.vayateens.com/
# http://www.ibericas.es/enindex.php
#
#listcategory: "Pornography (Spanish)"
#
< sexo ><40>      #sex
< tetas ><50>      #tits
< aberraciones ><30> #aberrations
< asiaticas ><20>   #asians
< bellezas ><20>    #beauties
< borrachas ><20>   #drunks
< camaras ocultas ><40> #hidden cameras
< casadas ><10>     #married
< chicas meando ><50> #girls peeing
< chicas >,<sexo ><50> #girl, sex
< clitoris ><30>    #clitoris
< colegialas ><20>  #college girls
< desnudos ><50>    #nude
< diosas del sexo ><40> #sex goddesses
< lesbianas ><10>   #lesbians
< famosas desnudas ><50> #famous nudes
[ 66 líneas leídas ]
^G Ver ayuda  ^C Guardar  ^N Buscar    ^K Cortar txt ^J Justificar ^C Posición
^X Salir      ^R Leer fich. ^M Reemplazar ^U Pegar txt  ^T Ortografía ^_ Ir a lí

```

También podremos encontrar más listas a parte de pornografía.

```

root@Proxy:/etc/dansguardian# nano lists/phraselists/
badwords/      games/         legaldrugs/    pornography/   travel/
chat/          goodphrases/  malware/       proxies/       upstreamfilter/
conspiracy/    googlesearches/ music/         rta/          violence/
domainsforsale/ gore/         news/         safelabel/     warezhacking/
drugadvocacy/  idtheft/     nudism/       secretsocieties/ weapons/
forums/        illegaldrugs/ peer2peer/     sport/        webmail/
gambling/      intolerance/  personals/    translation/

```

Para poder aplicar un archivo solo tendremos que definirlo de la siguiente forma:

```
bannedphraselist = '/etc/dansguardian/lists/bannedphraselist'
```

Una vez definido reiniciamos dansguardian y comprobamos el filtro.

Esta página estaba por defecto definida en el archivo de filtros de bannedphraselist.



Otros filtros a tener en cuenta.

Archivo	Descripción
bannedphraselist	Contiene una lista de frases prohibidas. Las frases deben estar entre <>. Por defecto incluye una lista ejemplo en inglés. Las frases pueden contener espacios. Se puede también utilizar combinaciones de frases, que si se encuentran en una página, serán bloqueadas.
bannedmimetyplist	Contiene una lista de tipos MIME prohibidos. Si una URL devuelve un tipo MIME incluido en la lista, quedará bloqueada. Por defecto se incluyen algunos ejemplos de tipos MIME que serán bloqueados.
bannedextensionlist	Contiene una lista de extensiones de archivos no permitidas. Si una URL termina con alguna extensión contenida en esta lista, será bloqueada. Por defecto se incluye un archivo ejemplo que muestra como denegar extensiones.
bannedregexprulldlist	Contiene una lista de expresiones regulares ³ que si se cumplen sobre la URL ésta será bloqueada.
bannedsitelist	Contiene una lista de sitios prohibidos. Si se indica un nombre de dominio todo él será bloqueado. Si se quiere sólo bloquear partes de un sitio hay que utilizar el archivo bannedurldlist . También se pueden bloquear los sitios indicados exepctuando los dados en el archivo exceptionsitelist . Existe la posibilidad de descargarse listas negras tanto de sitios como de URLs y situarlas en los archivos correspondientes. Están disponibles en http://dansguardian.org/?page=extras .
bannedurldlist	Permite bloquear partes específicas de un sitio web. bannedsitelist bloquea todo el sitio web y ésta sólo bloquea una parte.
banneduserlist	Lista de los nombres de usuario que estarán bloqueados.

Archivos de excepciones en /etc/dansguardian/

Archivo	Descripción
exceptionsitelist	Contiene una lista de los nombres de dominio que no serán filtrados Es importante tener en cuenta que el nombre de dominio no debe incluir http:// o www .

exceptionip	Contiene una lista de las direcciones IP de los clientes a los que se permite el acceso sin restricciones. Este sería el caso de la dirección IP del administrador.
exceptionuserlist	Lista de los nombres de usuarios que no serán filtrados en el caso de utilizar control de acceso por usuario. Requiere autenticación básica o "ident".
exceptionphraselist	Lista de las frases que, si aparecen en una página web, pasará el filtro.

Para saber más información de que contenido filtrar os dejo este link:

<https://www.ecured.cu/DansGuardian>

5. Radius con DNIE.

FreeRadius es un paquete de software de código abierto y libre distribución que permite implementar un servidor de RADIUS. El servidor de FreeRadius es modular, para facilitar su extensión, y es muy escalable. Además el almacenamiento de la información de autenticación (usuarios/contraseña) se puede realizar directamente (sobre ficheros de textos de configuración propios) o bien preguntando a bases de datos externas, como MySQL o bien ficheros del sistema/etc/passwd

a. PKI del DNIE

Las Autoridades de Certificación que la componen son:

- Una **Autoridad de Certificación raíz** que sólo emite certificados para sí misma y sus Autoridades de Certificación subordinadas.
- Tres **Autoridades de Certificación subordinadas** que emiten certificados para los titulares de DNIE, a cada ciudadano le corresponde uno de ellos, como a priori no sabemos cuál será el que firme el certificado de un ciudadano concreto, lo que haremos es concatenar los tres certificados en un fichero junto a la Autoridad raíz de manera que FreeRADIUS pedirá a OpenSSL que recorra el fichero buscando el certificado necesario para un cliente dado siempre que dejemos el certificado de la Autoridad raíz en último lugar.

Primero hay que descargar los cuatro certificados de la web del DNIE:

https://www.dnielectronico.es/PortalDNIE/PRF1_Cons02.action?pag=REF_076&id_menu=68

Nombre	Fecha de modificación	Tipo	Tamaño
AC DNIE 001	25/07/2018 8:20	Certificado de seg...	2 KB
AC DNIE 002	25/07/2018 8:21	Certificado de seg...	2 KB
AC DNIE 003	25/07/2018 8:22	Certificado de seg...	2 KB
AC DNIE 004	25/07/2018 8:25	Certificado de seg...	2 KB
AC DNIE 005	25/07/2018 8:27	Certificado de seg...	2 KB
AC DNIE 006	25/07/2018 8:28	Certificado de seg...	2 KB
AC RAIZ DNIE 2	16/05/2018 12:56	Certificado de seg...	3 KB

Los descomprimos y pasamos los cuatro ficheros por SFTP al servidor.

```
root@Proxy:/home/frodo/Certificate# ls
ACDNIE001-SHA1.crt ACDNIE002-SHA1.crt ACDNIE003-SHA1.crt ACRAIZ-SHA1.cer
```

Los ficheros vienen en formato DER (.crt), un formato propio de Windows, lo pasamos a PEM que es un formato más propio de Linux para que no haya ningún problema, lo haremos con OpenSSL situándonos en el directorio donde hemos copiado los ficheros ejecutamos lo siguiente.

```
root@Proxy:/home/frodo/certificate# rm ACDNI001.pem
root@Proxy:/home/frodo/certificate# openssl x509 -inform DER -outform PEM -in AC\ DNIIE\ 001.crt -out ACDNI001.pem
root@Proxy:/home/frodo/certificate# openssl x509 -inform DER -outform PEM -in AC\ DNIIE\ 002.crt -out ACDNI002.pem
root@Proxy:/home/frodo/certificate# openssl x509 -inform DER -outform PEM -in AC\ DNIIE\ 003.crt -out ACDNI003.pem
root@Proxy:/home/frodo/certificate# openssl x509 -inform DER -outform PEM -in AC\ DNIIE\ 004.crt -out ACDNI004.pem
root@Proxy:/home/frodo/certificate# openssl x509 -inform DER -outform PEM -in AC\ DNIIE\ 005.crt -out ACDNI005.pem
root@Proxy:/home/frodo/certificate# openssl x509 -inform DER -outform PEM -in AC\ DNIIE\ 006.crt -out ACDNI006.pem
root@Proxy:/home/frodo/certificate# ls
ACDNI001.pem ACDNI002.pem ACDNI003.pem ACDNI004.pem ACDNI005.pem ACDNI006.pem AC DNIIE 001.crt AC DNIIE 002.crt AC DNIIE 003.crt AC DNIIE 004.crt AC DNIIE 005.crt AC DNIIE 006.crt AC RAIZ DNIIE 2.crt
root@Proxy:/home/frodo/certificate#
```

Ahora toca concatenar los certificados en único archivo que llamaremos "todas.pem".

```
root@Proxy:/home/frodo/Certificate# cp AC\ RAIZ\ DNIIE\ 2.crt todas.pem
root@Proxy:/home/frodo/Certificate# cat ACDNI001.pem >> todas.pem
root@Proxy:/home/frodo/Certificate# cat ACDNI002.pem >> todas.pem
root@Proxy:/home/frodo/Certificate# cat ACDNI003.pem >> todas.pem
root@Proxy:/home/frodo/Certificate# cat ACDNI004.pem >> todas.pem
root@Proxy:/home/frodo/Certificate# cat ACDNI005.pem >> todas.pem
root@Proxy:/home/frodo/Certificate# cat ACDNI006.pem >> todas.pem
```

b. Instalación de FreeRADIUS

Descargaremos FreeRadius 3.0.20 y lo instalaremos.

```
root@Proxy:/home/frodo# wget ftp://ftp.freeradius.org/pub/freeradius/freeradius-server-3.0.20.tar.gz
```

```
root@Proxy:/home/frodo# tar xzf freeradius-server-3.0.20.tar.gz
```

Será necesario instalar los siguientes paquetes porque contienen librerías de desarrollo necesarias para la instalación de FreeRADIUS.

```
root@Proxy:/home/frodo# cd freeradius-server-3.0.20
root@Proxy:/home/frodo/freeradius-server-3.0.20# apt-get install libtalloc-dev
```

```
root@Proxy:/home/frodo/freeradius-server-3.0.20# apt install libssl-dev
```

```
root@Proxy:/home/frodo/freeradius-server-3.0.20# apt install build-essential
```

Ejecutamos él. ./configure.

```
root@Proxy:/home/frodo/freeradius-server-3.0.20# ./configure
```

```
root@Proxy:/home/frodo/freeradius-server-3.0.20# make
```

Instalamos radius.

```
root@Proxy:/home/frodo/freeradius-server-3.0.20# make install
```

c. Configuración de FreeRADIUS

Por defecto la instalación se realiza en /usr/local/etc/raddb/.

Empezaremos con el fichero `radius.conf`, en el apartado `PROXY CONFIGURATION` deshabilitaremos el servidor proxy y comentaremos el `include`, no es estrictamente necesario pero ahorrará recursos.

```
root@Proxy:/home/frodo/freeradius-server-3.0.20# nano /usr/local/etc/raddb/radiusd.conf
```

```
#
proxy_requests = no
$INCLUDE proxy.conf
```

En el apartado `security` modificamos el parámetro `allow_vulnerable_openssl` para permitir que FreeRADIUS se inicie con una versión vulnerable de OpenSSL.

```
#
allow_vulnerable_openssl = yes
```

Continuaremos con el fichero `clients.conf`, aquí añadimos el o los clientes RADIUS esto es el punto de acceso inalámbrico o en mi caso el router inalámbrico que configuraremos con los datos aquí introducidos, añadimos el cliente al final del fichero con la IP y prefijo que tendrá el punto de acceso inalámbrico, la `secret` es una clave cualquiera que permitirá comunicarse el punto de acceso con el servidor, y un nombre para el cliente que estamos introduciendo (`shortname`).

```
root@Proxy:/home/frodo# nano /usr/local/etc/raddb/clients.conf
```

```
1 client S&SFree {
2     ipaddr      = 192.168.3.1
3     secret      = bolson
4     shostname    = S&SFree
5 }
```

Continuamos con el archivo `default` dentro de `sites-enabled`, en este fichero se configuran los distintos host virtuales al igual que haríamos con Apache. En las secciones `listen` no es estrictamente necesario modificar nada, el servidor funcionará igualmente con la configuración que viene por defecto, las podemos configurar de la siguiente manera para hacer el servidor más seguro evitando que escuche peticiones provenientes de una IP o puertos distintos a los que tenemos planeado.

La primera sección `listen` (paquetes de autenticación) la configuramos de la siguiente manera, siendo `ipv4addr` la IP del adaptador que está conectado al mikrotik.

```
root@Proxy:/home/frodo# nano /usr/local/etc/raddb/sites-enabled/default
```

```
type = auth
```



```
# listen {  
    # ipv4addr = 192.168.3.2  
  
    # Port on which to listen.  
    # Allowed values are:  
    #   integer port number (1812)  
    #   0 means "use /etc/services for the proper port"  
    port = 1812
```

La segunda sección *listen* (paquetes de contabilización) la configuramos de la siguiente manera. La opción *ipaddr* se deberá cambiar por *ipv4addr*.

```
#  
listen {  
    # ipv4addr = 192.168.3.2  
    # ipv6addr = ::  
    port = 1813  
    type = acct  
    # interface = eth0  
    # clients = per_socket_clients
```

En la sección *authorize* tenemos que deshabilitar el filtro de nombre de usuario, éste comprueba si el nombre de usuario contiene espacios o caracteres inválidos rechazando el acceso en su caso. El *CommonName* del certificado de cliente contenido en el DN se compone de apellidos, nombre, espacios en blanco, una coma y el literal "(AUTENTICACIÓN)", el filtro rechazaría el acceso no permitiendo que continúe la petición del cliente.

```
#filter_username
```

Como nuestra intención es que los usuarios únicamente puedan tener acceso a la red WiFi mediante el DN, deshabilitaremos los módulos de autenticación que no vamos a utilizar, dejaremos *eap* habilitado ya que es la manera en que se autenticaran los clientes.

```
#chap
```

```
#mschap
```

```
#digest
```

```
#suffix
```

```
#pap
```

Una vez que el usuario está autorizado la sección *authenticate* se encargará de instanciar el módulo de autenticación, esta sección la podemos dejar tal cual ya que hemos descartado los módulos que no vamos a utilizar en la sección anterior.

Por ultimo debemos en el módulo *eap* tenemos que indicar dónde está el certificado de autoridad que controlará las credenciales de los clientes y firmará sus certificados contenidos en el DNle.

```
root@Proxy:/home/frodo# nano /usr/local/etc/raddb/mods-enabled/eap
```

```
ca_file = /usr/local/etc/raddb/certs/todas.pem
```

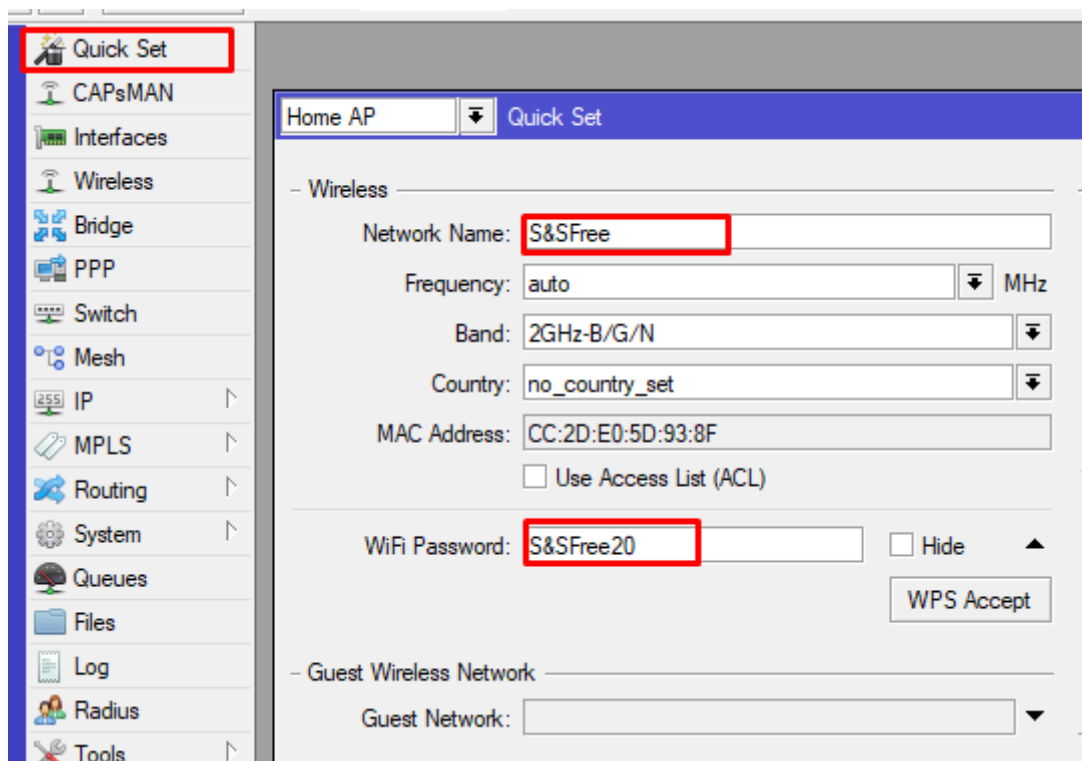
Por ultimo copiamos el fichero con los certificados al directorio de certificados de FreeRADIUS.

```
root@Proxy:/home/frodo# cp Certificate/todas.pem /usr/local/etc/raddb/certs/
```

d. Punto de acceso

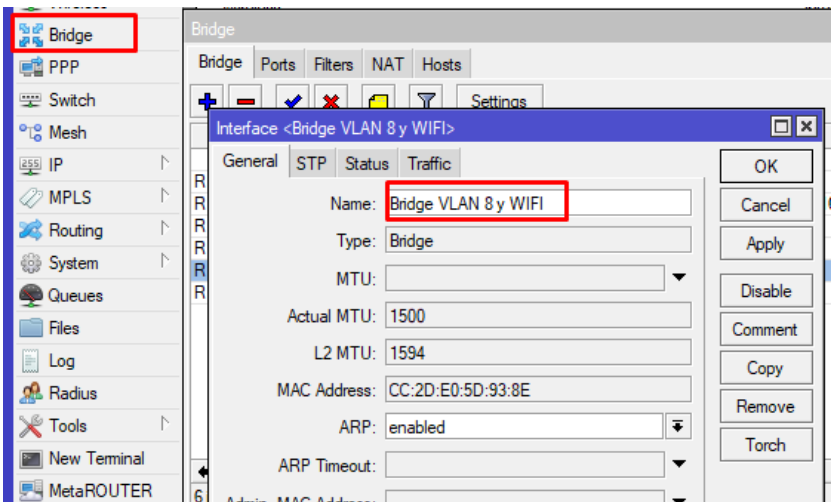
Como cliente RADIUS hace falta configurar un punto de acceso inalámbrico, en este caso se hará con un mikrotik. Lo configuramos con los siguientes pasos.

Pondremos un nombre a la WiFi, en la sección *Quick Set*.

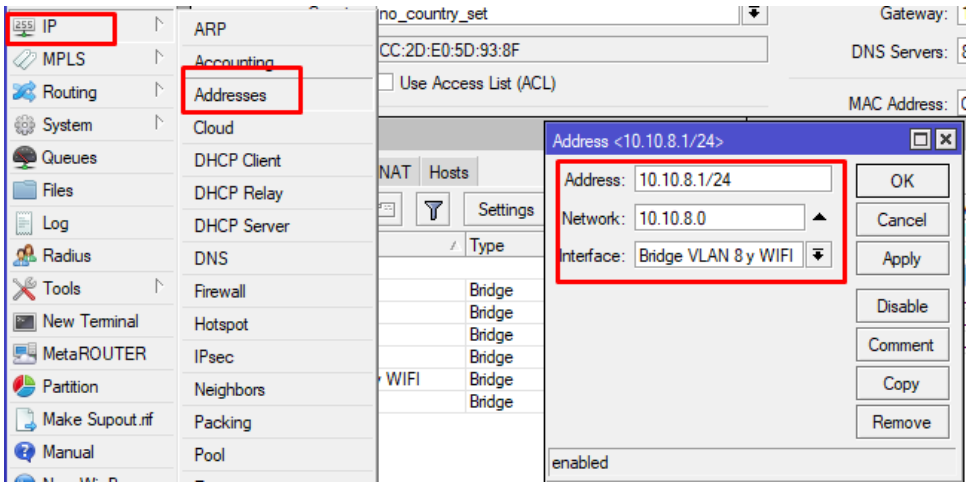


Crearemos un Bridge para la WLAN y definiremos una Ip para dicho Bridge.

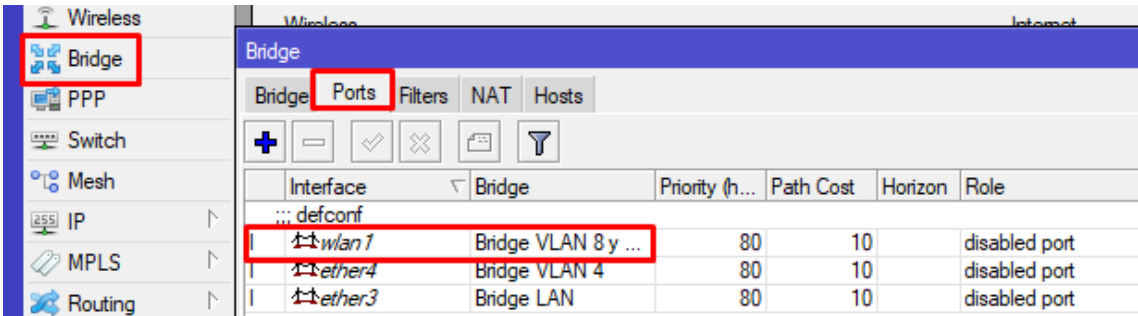
Bridge:

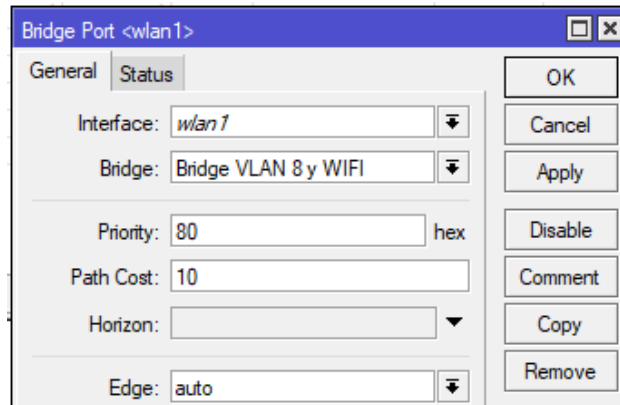


Definir una ip para ese bridge.



Lo siguiente que haremos será asociar el bridge que hemos creado a la interfaz de wlan.

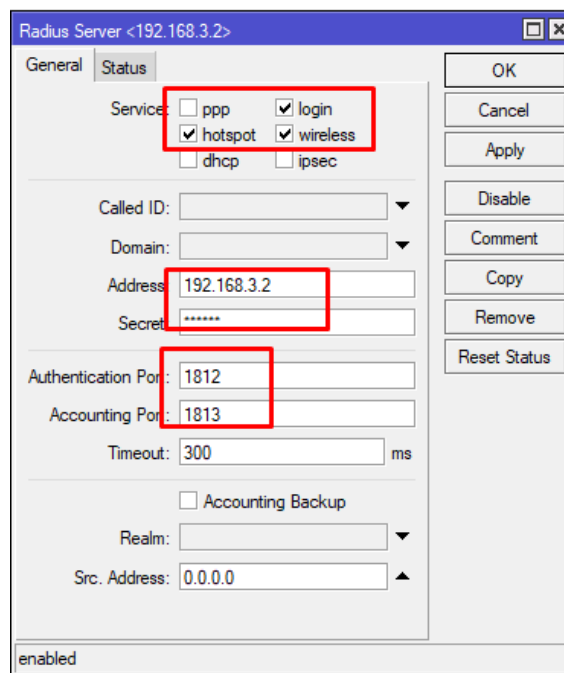




En este caso también se ha creado un servidor DHCP para que reparta IP por la Wifi, pero eso es irrelevante para hacer esta práctica.

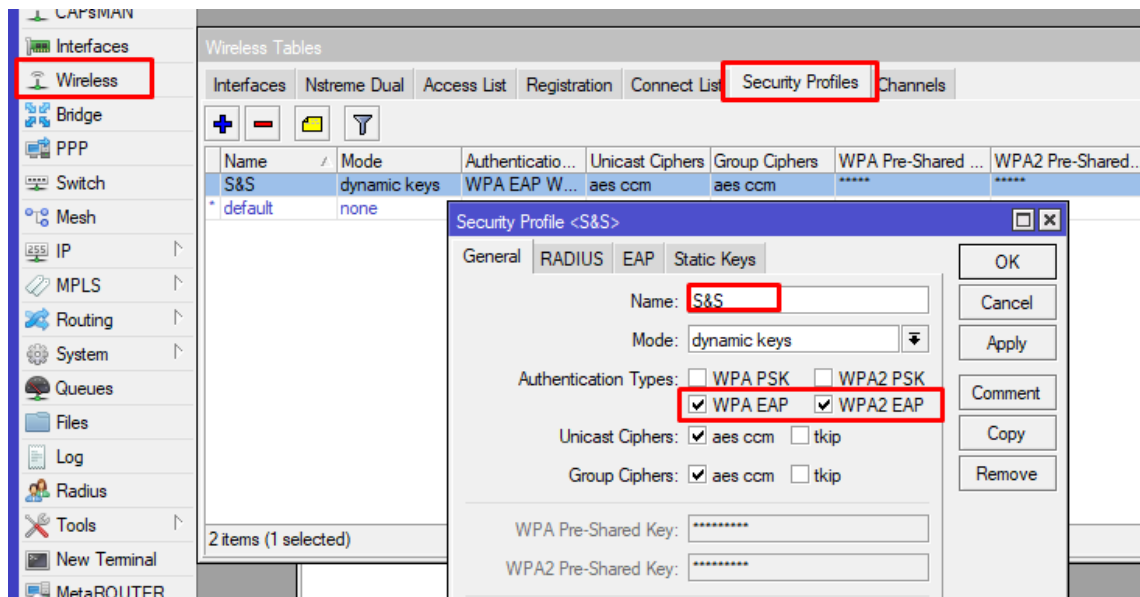
Una vez configurado esto nos iremos a la sección de Radius para hacer la comunicación con el servidor.

En esta sección lo que haremos será marcar las opciones de hotspot y Wireless, pondremos la IP del servidor Radius, la contraseña que hemos definido en el archivo client.conf y los puertos del servidor Radius.

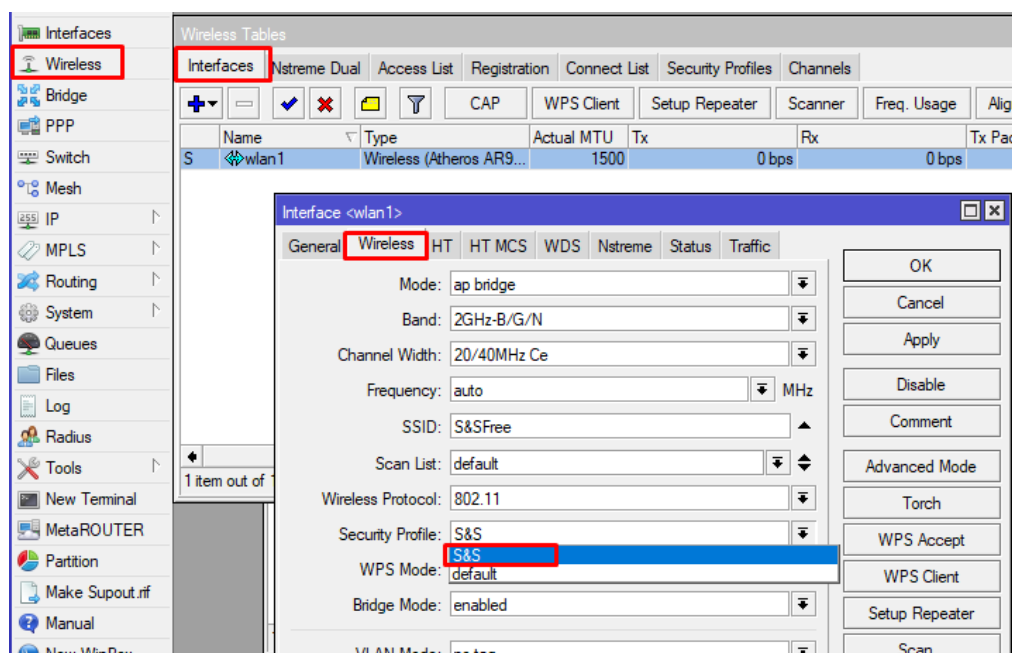


Después de establecer conexión con el servidor lo que haremos será ir a la sección de *Wireless* aquí nos iremos a la Pestaña de *Security Profiles*.

Aquí definiremos el Perfil de seguridad, le pondremos un nombre y marcaremos las opciones de WPA EAP y WPA2EAP, estas dos opciones nos dejarán autenticarnos con los DNIE.



Por ultimo deberemos de ir a la pestaña de Interfaces y seleccionar el perfil de seguridad que acabamos de crear.

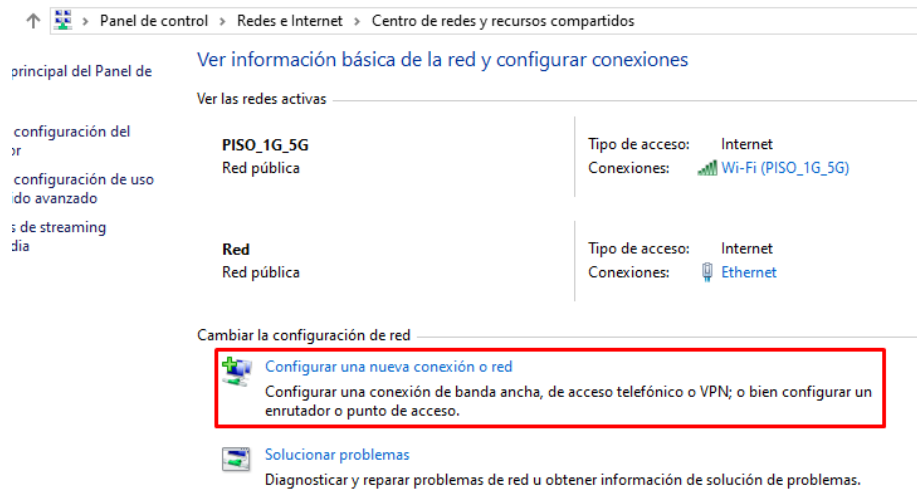


e. Clientes

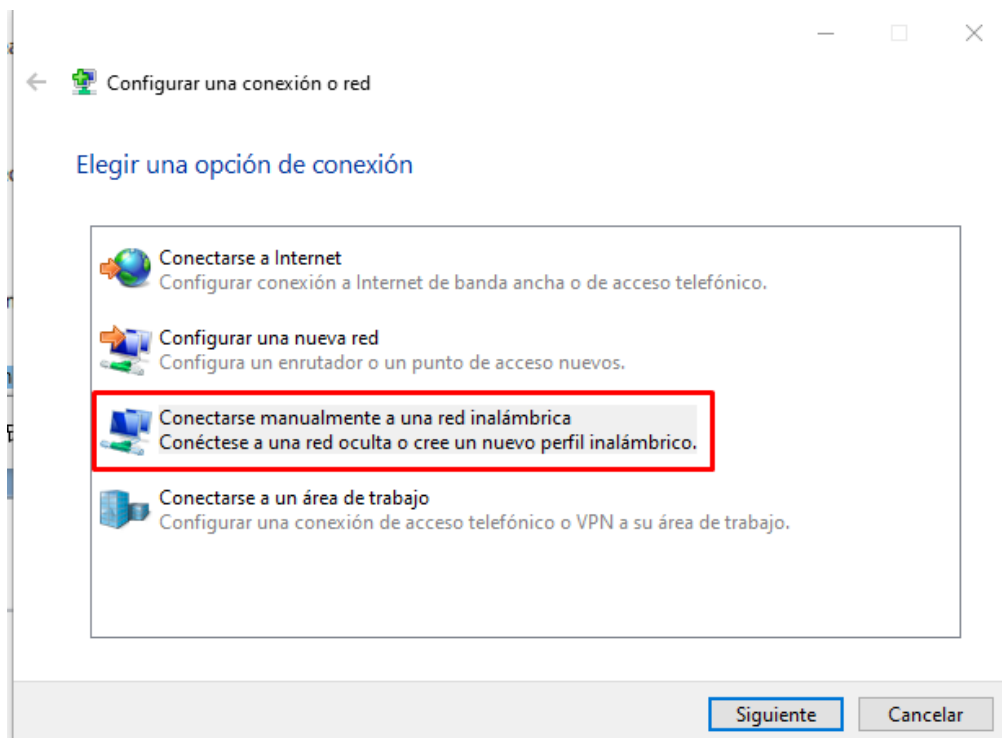
Para que un cliente pueda conectarse a la red WiFi será necesario un PC o portátil con tarjeta de red inalámbrica, sistema operativo Windows, un lector de tarjetas inteligentes, su DNI electrónico, y según qué casos hará falta el software que proporciona el Cuerpo Nacional de Policía llamado “Módulo criptográfico para el DNLe”.

En Windows 7,8 y 10 no es necesaria su instalación ya que al introducir el DNLe en el lector automáticamente se descargarán los drivers necesarios desde Windows Update, permitiendo trabajar al DNLe como dispositivo plug&play.

Vamos a crear una nueva conexión de red inalámbrica, nos vamos a “Panel de control\Redes e Internet\Centro de redes y recursos compartidos”, pinchamos sobre “Configurar una nueva conexión o red”.



Seleccionamos la opción de “Conectarse manualmente a una red inalámbrica” y pulsamos “Siguiente”.



Introducimos los datos que habíamos configurado en el punto de acceso WiFi:

- Nombre de la red: S&SFree
- Tipo de seguridad: WPA2-Enterprise
- Tipo de cifrado: AES

← Conectarse manualmente a una red inalámbrica

Escriba la información de la red inalámbrica que desea agregar.

Nombre de la red: S&SFree

Tipo de seguridad: WPA2-Enterprise

Tipo de cifrado: AES

Clave de seguridad: ☐ Ocultar caracteres

☐ Iniciar esta conexión automáticamente

☐ Conectarse aunque la red no difunda su nombre

Advertencia: esta opción podría poner en riesgo la privacidad del equipo.

Siguiete Cancelar

Pulsamos “Siguiete” lo que nos lleva a la siguiente ventana donde pincharemos sobre “Cambiar la configuración de conexión” para realizar algunos ajustes sobre la nueva conexión.

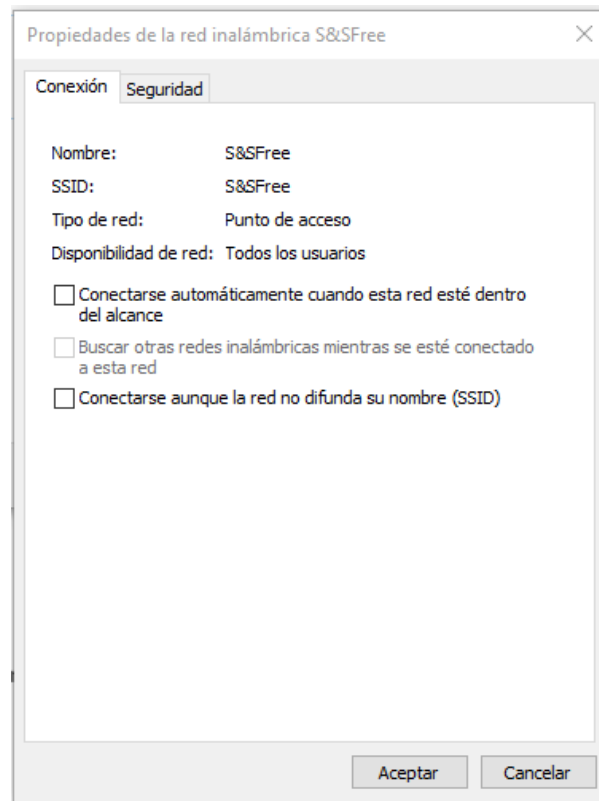
← Conectarse manualmente a una red inalámbrica

S&SFree se agregó correctamente.

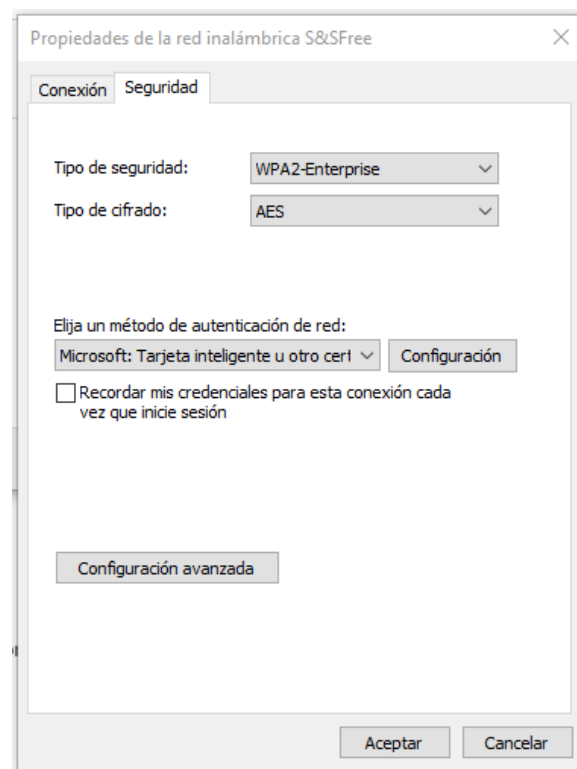
→ Cambiar la configuración de conexión
Abra las propiedades de la conexión para cambiar la configuración.

Cerrar

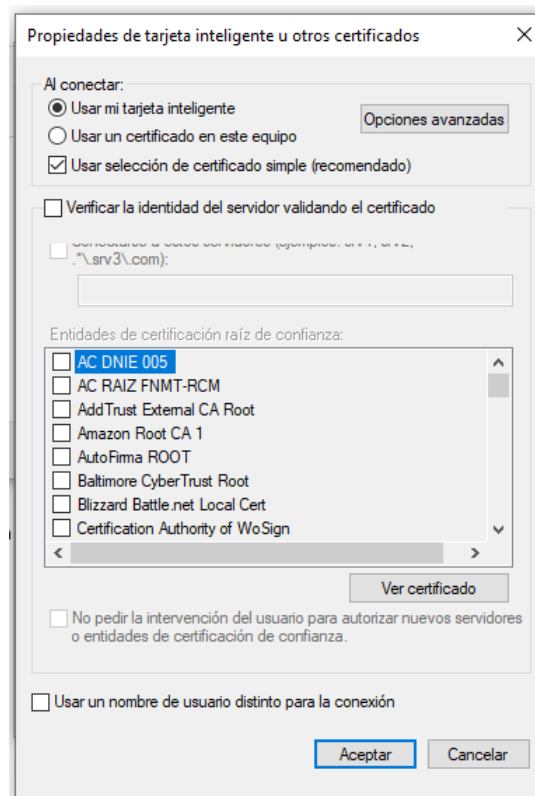
Nos aparece una nueva ventana con las propiedades de la conexión, de la pestaña “Conexión” no tocaremos nada.



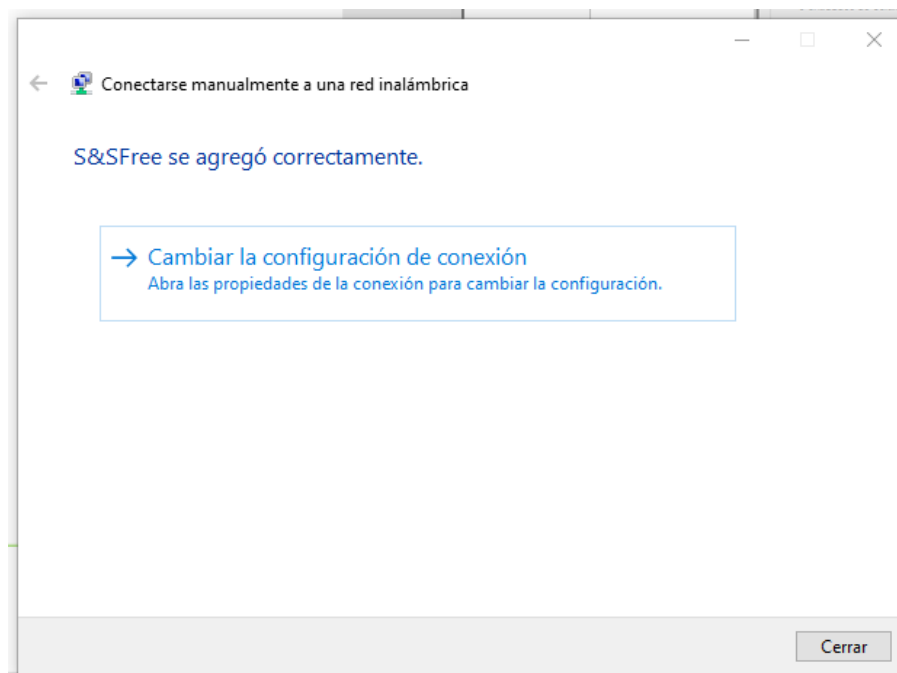
Pasamos a la pestaña “Seguridad”, en el menú “Elija un método de autenticación de red:” seleccionamos “Microsoft: Tarjeta inteligente u otro certificado”, luego pinchamos sobre el botón “Configuración”.



En “Al conectar:” seleccionamos “Usar mi tarjeta inteligente”, desmarcamos la casilla “Validar un certificado de servidor”.



Finalmente pinchamos sobre “Cerrar” en la ventana anterior con lo que tendremos configurada la conexión al servidor RADIUS mediante DNE.



f. Pruebas

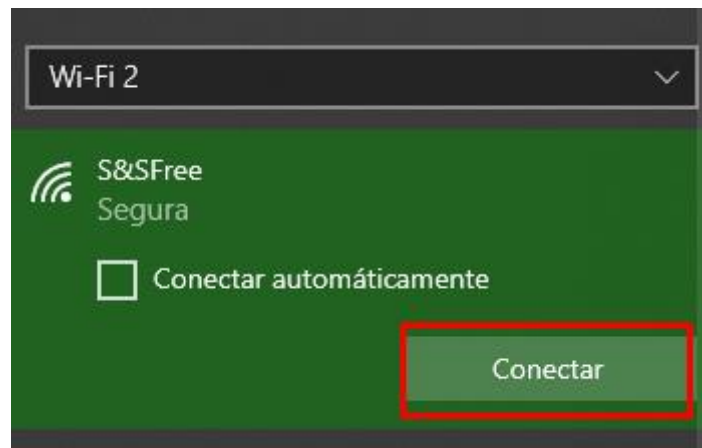
Iniciamos FreeRADIUS con alguno de los siguientes comandos, el “-X” es para iniciarlo en modo debug.

```
root@Proxy:/home/frodo# radiusd -X
```

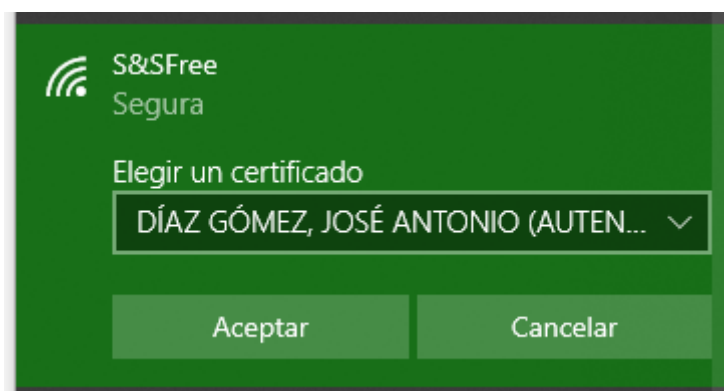
Se iniciará el servidor. Si todo ha ido bien al final del log nos aparecerá el mensaje “Ready to process requests”.

```
}  
}  
Listening on auth address 192.168.3.2 port 1812 bound to server default  
Listening on acct address 192.168.3.2 port 1813 bound to server default  
Ready to process requests
```

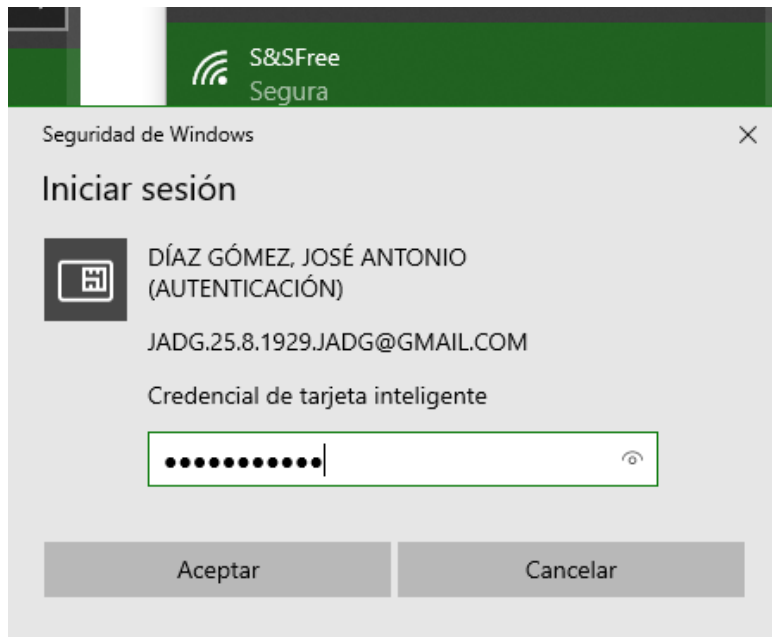
Con el lector de tarjetas inteligentes instalado, conectado, y el DNle insertado, buscamos la red WiFi que hemos configurado en el cliente FreeRADIUS y le damos a conectar.



Nos debería de seleccionar el método de autenticación de los certificados del DNle.



Seguidamente deberemos de poner el PIN del Dnie electrónico y si todo ha ido bien debería de conectar a la red.



```
(8) Received Access-Request Id 19 from 192.168.3.1:37797 to 192.168.3.2:1812 length 268
(8) Service-Type = Framed-User
(8) Framed-MTU = 1400
(8) User-Name = "DÍAZ GÓMEZ, JOSÉ ANTONIO (AUTENTICACIÓN)"
(8) State = 0xfc077bddfb0e765669e81f09fe7738e5
(8) NAS-Port-Id = "wlan1"
(8) NAS-Port-Type = Wireless-802.11
(8) Acct-Session-Id = "82200000"
(8) Acct-Multi-Session-Id = "CC-2D-E0-5D-93-8F-50-3E-AA-77-5F-DD-82-20-00-00-00-00-00"
(8) Calling-Station-Id = "50-3E-AA-77-5F-DD"
(8) Called-Station-Id = "CC-2D-E0-5D-93-8F:S&SFree"
(8) EAP-Message = 0x020900060d00
(8) Message-Authenticator = 0xfb2291f8dd389ac2090ad844ce1f4fd2
(8) NAS-Identifier = "MikroTik"
```

Para poder definir que solo los usuarios que estén en el sistema se puedan loguear se tendrá que definir en el fichero **users** las siguientes opciones.

```
root@Proxy:/home/frodo# nano /usr/local/etc/raddb/users

# be given any additional resources.
#
DEFAULT Auth-Type := Reject
|       Reply-Message = "ACCESO DENEGADO."
#
```

Ahora comprobaremos que no nos deja loguearnos y nos saldrá el siguiente mensaje de error.

```
files: users: Matched entry DEFAULT at line 66
  [files] = ok
  [expiration] = noop
  [logintime] = noop
} # authorize = updated
Found Auth-Type = Reject
Auth-Type = Reject, rejecting user
Failed to authenticate the user
Using Post-Auth-Type Reject
```

Para poder loguearnos con un usuario específico solo tendremos que poner las siguientes líneas en el fichero **users**. En este fichero pondré el nombre del usuario y el tipo de protocolo que será **EAP**. La opción definida anteriormente se dejara sin comentar para que solo los usuarios que estén definidos en dicho archivo puedan acceder a la Wi-Fi.

```
"DÍAZ GÓMEZ, JOSÉ ANTONIO (AUTENTICACIÓN)" Auth-Type := EAP
  Reply-Message = "Acceso permitido para el usuario: %{User-Name}."

#
# Deny access for a group of users.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
DEFAULT Auth-Type := Reject
  Reply-Message = "ACCESO DENEGADO."
```

Ahora comprobaremos que el usuario se loguea en la red Wi-Fi y sale el mensaje que hemos definido en el fichero.

```
(8) eap: Peer sent EAP Response (code 2) ID 9 length 6
(8) eap: No EAP Start, assuming it's an on-going EAP conversation
(8) [eap] = updated
(8) files: users: Matched entry DÍAZ GÓMEZ, JOSÉ ANTONIO (AUTENTICACIÓN) at line 57
(8) files: EXPAND Acceso permitido para el usuario: %{User-Name}.
(8) files: --> Acceso permitido para el usuario: DÍAZ GÓMEZ, JOSÉ ANTONIO (AUTENTICACIÓN).
(8) [files] = ok
```

Vemos como se ha conectado a nuestra red.



Comprobamos que utilizando un DNIE que no está en el fichero **users** definido no es posible conectarnos a la red Wi-Fi.

```
} # authorize = updated
Found Auth-Type = Reject
Auth-Type = Reject, rejecting user
Failed to authenticate the user
Using Post-Auth-Type Reject
# Executing group from file /usr/local/etc/raddb/sites-enabled/default
Post-Auth-Type REJECT {
attr_filter.access_reject: EXPAND %{User-Name}
attr_filter.access_reject: --> DÍAZ GÓMEZ, ROCÍO (AUTENTICACIÓN)
attr_filter.access_reject: Matched entry DEFAULT at line 11
[attr_filter.access_reject] = updated
... Expired FRAP session with state 0x74448e0074448e00
```



WEBGRAFÍA

IPTABLES, NFTABLES

<https://linux-audit.com/differences-between-iptables-and-nftables-explained/>

<https://www.wifi-libre.com/topic-1319-por-que-y-como-pasar-de-iptables-a-nftables.html>

<https://www.ochobitshacenunbyte.com/2019/11/08/nftables-el-cortafuegos-sucesor-de-iptables/>

[https://wiki.archlinux.org/index.php/Nftables_\(Espa%C3%B1ol\)](https://wiki.archlinux.org/index.php/Nftables_(Espa%C3%B1ol))

<http://es.tldp.org/Manuales-LuCAS/doc-iptables-firewall/doc-iptables-firewall.pdf>

<https://www.linuxito.com/seguridad/793-tutorial-basico-de-iptables-en-linux>

https://openwebinars.net/cursos/iptables/?utm_source=blog&utm_medium=cta&utm_campaign=cursos-iptables

SQUID, SARG, DANSGUARDIAN

<https://wiki.squid-cache.org/SquidFaq/SquidAcl>

<http://javier-contreras.blogspot.com/2009/02/instalar-y-configurar-dansguardian.html>

<https://rooteando.com/pdf/servidor-proxy-squid-sarg-y-dansguardian>

https://www.server-world.info/en/note?os=Debian_9&p=squid&f=8

<https://openbinary20.com/2018/02/16/servidor-con-debian-instalacion-y-configuracion-de-un-servidor-proxy-cache/>

<https://www.bdat.net/documentos/squid/x30.html>

HTTPS

<https://docs.docker.com/install/linux/docker-ce/debian/>

<https://github.com/diladele/docker-websafety>

DNIE

<https://drive.google.com/drive/folders/1tArlmfmwXLazmNixgPJaqmIC1wIFCYJc>

http://oa.upm.es/1602/1/PFC_SERGIO_YEBENES_MORENO.pdf

<http://seguridadxredes.blogspot.com/2015/12/vpn-ipsec-ii-autenticando-con-dnie-o.html>

<https://www.eduardocollado.com/2017/07/24/conexion-a-mikrotik-via-radius/>