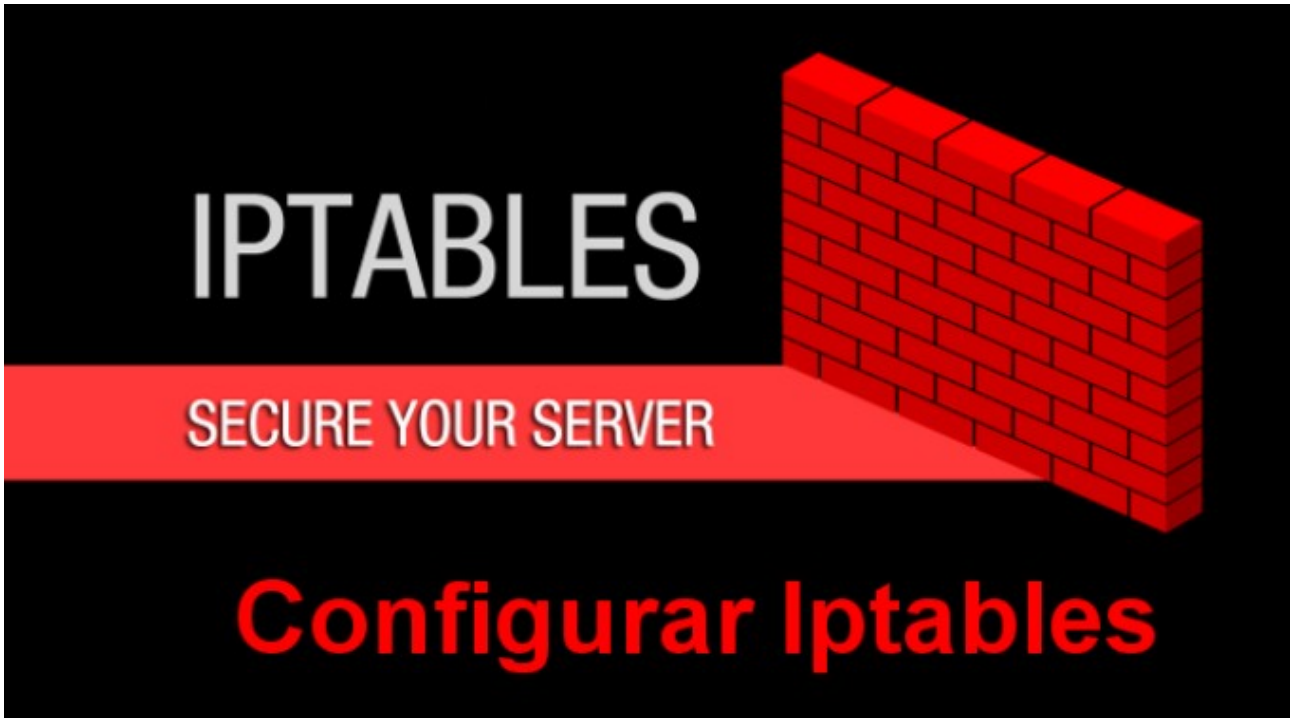
I.E.S Politécnico Jesús Marín
PROYECTO INTEGRADO

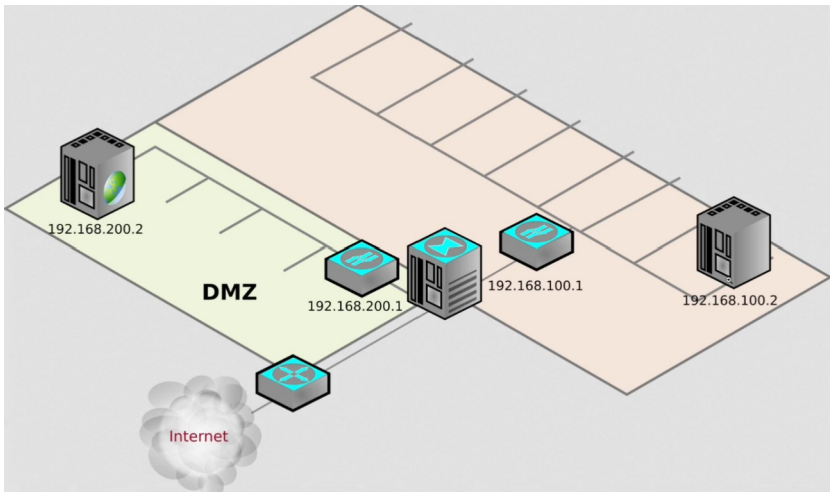


Alumno: Miguel Ángel de Miguel Bandera
GRADO SUPERIOR ASIR

ÍNDICE

1. Introducción

En este proyecto se va a realizar una configuración de un cortafuegos perimetral con IPTABLES y un diseño de la red informática de una empresa para proteger la red interna del exterior y también para ubicar los distintos servicios en una red distinta a la red común para protegerla en caso de que uno de nuestros servicios ubicados en la DMZ haya sido comprometido. Se configurará los distintos servicios desde el punto de vista de seguridad.



Topología de la red

Los servicios que existen en la red DMZ:

- Servidor VPN
- Servidor Samba
- Servidor Apache
- Servidor MySQL

En la red común hay un servidor PXE para instalar sistemas operativos vía red. Esta red estaría en una VLAN de manera que la red común no puede comunicarse con el servidor PXE. Es buena práctica ya que el propio servidor PXE trae consigo un servidor DHCP y ya tenemos un servidor DHCP instalado en el cortafuegos. Solo queremos un servidor DHCP en la red común, el servidor DHCP del cortafuegos.

De esta manera evitamos que el servidor DHCP de PXE configure equipos de la red común.

1.1 ¿Qué es un cortafuegos?

Cortafuegos es estrictamente un dispositivo de seguridad para controlar el tráfico de red, seleccionando el tráfico conforme a unas reglas establecidas.

Permite analizar el tráfico de red al atravesar un dispositivo y tomar decisiones acerca del mismo:

- Permitir el paso
- Denegar el paso (silenciosamente o no)
- Modificar el tráfico
- Seleccionar determinado tráfico para otra aplicación
- Se usa por seguridad, para controlar o modificar el tráfico y para mejorar el rendimiento

Ejemplos de filtrado:

- Analizamos el tráfico proveniente de un nodo de nuestra red que está provocando problemas.
- Bloqueamos el tráfico de un nodo de Internet que está atacando nuestra red.
- Bloqueamos el tráfico web desde una dirección IP que está haciendo descargas masivas ocasionando la ralentización del tráfico del resto de usuarios.
- Permitimos la respuesta de ping desde el exterior solo a un nodo de nuestra red y limitamos la tasa de solicitud a un ritmo máximo de 3 por segundo.

Hay dos tipos de cortafuegos:

- Cortafuegos por hardware
- Cortafuegos por software

El dispositivo que actúa como cortafuegos (hardware) puede tener otras funcionalidades:

- Proxy
- Router
- Dispositivo de NAT
- Modificación del tráfico
- VPN
- ...

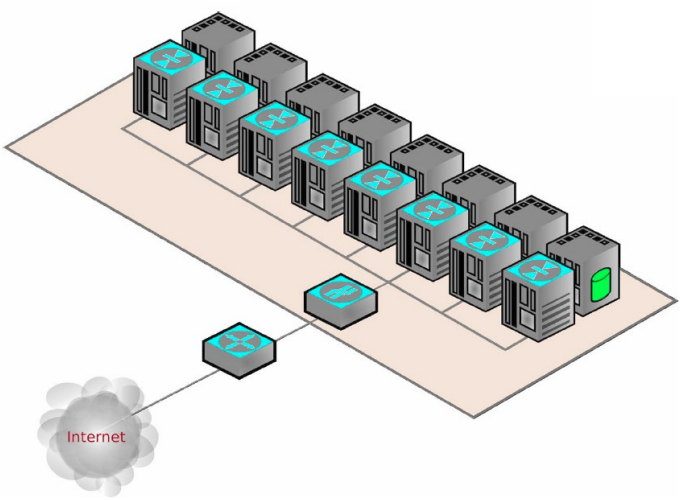
Cortafuegos (software) con **iptables**, las opciones más extendidas son:

- En un equipo con GNU/Linux o en un dispositivo específico, iptables proporciona más funciones que no son estrictamente de cortafuegos, principalmente:
  - NAT.
  - Modificación de los paquetes.

2. Esquemas de cortafuego

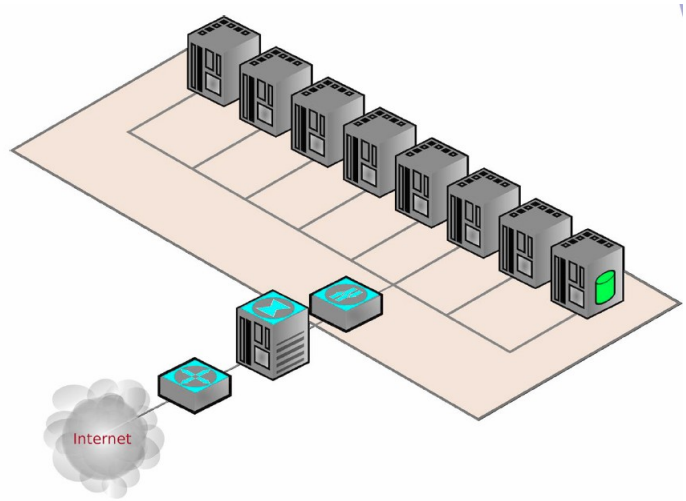
A continuación se presenta distintos esquemas que puede haber en la red local

a) Cortafuegos en cada uno de los nodos



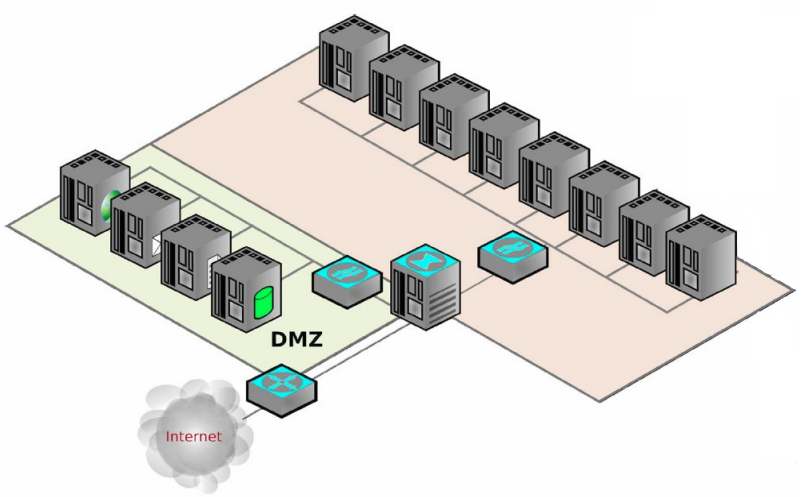
Cortafuegos en cada nodo

b) Cortafuegos antes del router



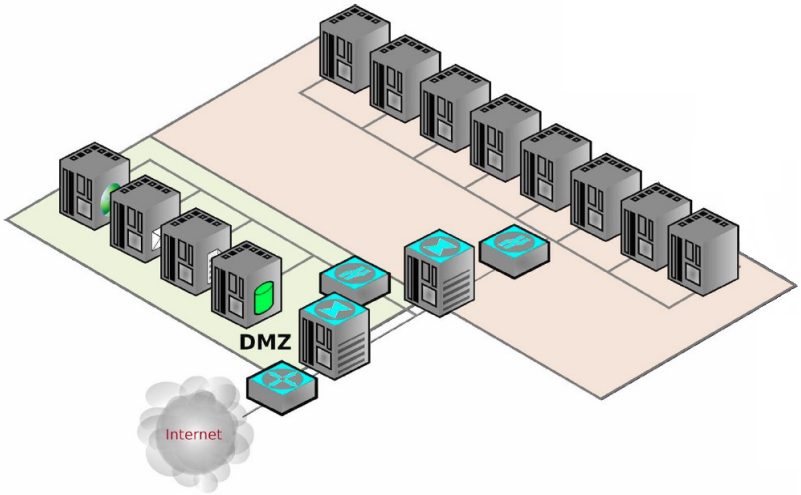
Cortafuegos entre la red local y el router

c) Cortafuegos perimetral



Cortafuegos perimetral

d) Doble cortafuegos perimetral



Doble cortafuegos perimetral

2.1 Categorías

- **Por nodo (personal)**: Ubicamos un cortafuegos delante de cada nodo o en el propio nodo (interno), protegiendo cada nodo individualmente.
  - Puede proteger del tráfico externo
  - Puede proteger del tráfico interno
  - No ha sido la opción preferida en ámbito corporativo
  - Vuelve a serlo gracias a la automatización de la configuración y tecnologías asociadas en cloud computing
- **Perimetral**: Ubicamos un cortafuegos en cada nodo del perímetro de la red, protegiendo la red del exterior.
  - Centramos la configuración en pocos nodos, en aquel o aquellos que nos conectan a otras redes

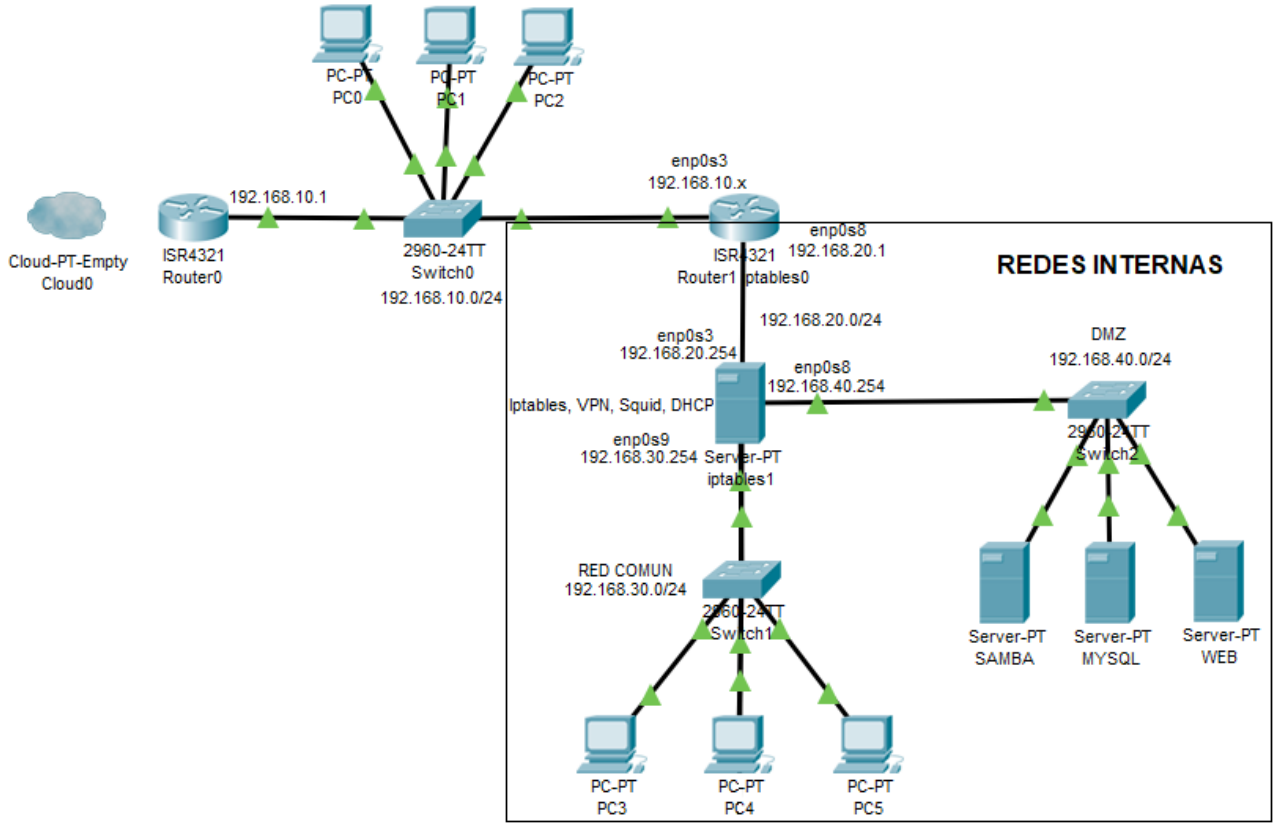
- Nodos fuertemente configurados y monitorizados
- Elección tradicional en ámbito corporativo

El problema del cortafuegos **perimetral**, es que en caso de que un equipo haya sufrido un ataque que desconoce el usuario y este dentro de la red local, está amenazando a todos los equipos de su red. En este escenario un cortafuegos perimetral no puede hacer nada. Por este motivo las empresas con cortafuegos perimetral suelen ser muy restrictivos en el uso de equipos precisamente por este motivo.

En este supuesto si todos los equipos de esa red estuvieran configurados por **nodo**, podría limitarse estos ataques. El problema es la dificultad de administrar todos los nodos.

3. Topología de la red

Una parte de la red pertenece a la red física de Escoeficiencia (la red 192.168.10.0/24). Las redes que están en el recuadro de REDES INTERNAS, son redes que he creado con máquinas virtuales en VirtualBox. El esquema es el siguiente:



Topología de la red

El router llamado iptables0, es simplemente una máquina virtual de Ubuntu server simulando un router, redireccionando el tráfico. Los paquetes que tengan como puerto destino, al servidor VPN (1194), para establecer el túnel y una vez establecido, podría

realizar una conexión al servidor Samba para compartir ficheros. De otra forma no se podría acceder a Samba desde fuera.

Desde Internet, podrán acceder al servidor web. El servidor web Apache se conectará a la base de datos MYSQL. Se ha separado estos servicios por si el servidor web sufre algún ataque, consiguiendo que el atacante no pueda tener acceso a toda la base de datos. Solo podrá acceder a la base de datos con un usuario que no tiene privilegios, negandole el acceso a otras bases de datos o alguna tabla con información sensible. En el servidor web no podrá descargarse nada de internet consiguiendo que no se descargue contenido malicioso desde el servidor web.

4. Reglas del “router” llamado iptables0

A continuación muestro las reglas establecidas en el iptables del “router”.

```
#!/bin/bash

# FLUSH de reglas

iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Establecemos politica por defecto

iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

# Todos los paquetes que entren por la interfaz enp0s3 que cumplan con esta regla, se redireccionará a la ip 192.168.20.254:1194 para poder establecer el tunel VPN. El puerto 1194 es el puerto del servidor openVPN

iptables -t nat -I PREROUTING -i enp0s3 -p udp --dport 1194 -j DNAT \
--to-destination 192.168.20.254:1194
 iptables -t nat -I PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT \
--to-destination 192.168.40.2:80


# Los paquetes que salgan por la interfaz enp0s3, se hará source nat, es decir se cambiará la ip a la 192.168.10.x

iptables -t nat -I POSTROUTING -s 192.168.20.0/24 -o enp0s3 -j MASQUERADE
```

4.1 Configuración de red del “router” iptables0

Muestro la configuración de las interfaces de red, ya que en esta topología he tenido que añadir rutas de encaminamiento.

```
# This file is generated from information provided by
# the datasource.  Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        enp0s3:
#           addresses: []
            dhcp4: true
        enp0s8:
            addresses: [192.168.20.1/24]
            routes:
             - to: 192.168.30.0/24
               via: 192.168.20.254
             - to: 192.168.40.0/24
               via: 192.168.20.254
    version: 2
```

Para que el paquete pueda llegar desde la red 192.168.10.x hasta la red 192.168.30.0 o a la red 192.168.40.0 hay que indicarle cual será el siguiente salto, en este caso es la ip **192.168.20.254.** Las rutas son para decirle al “router” como alcanzar las redes que no tiene directamente conectadas en sus “patas”.

También se puede añadir manualmente en la terminal, pero al reiniciarse el equipo, las rutas introducidas se borrarían. Aún así, merece la pena saberlo por si en algún escenario tenemos que optar por esta vía.

```
route add -net 192.168.30.0/24 gw 192.168.20.254
route add -net 192.168.40.0/24 gw 192.168.20.254
```

5. Reglas iptables en el servidor principal.

En este firewall, la política por defecto es **DROP**, con lo que conlleva más dificultad a la hora de administrarlo, pero consigues más seguridad en tu red, ya que solo aceptas las reglas que necesitas, y si desconoces algún servicio, el firewall por defecto lo bloqueará. Cada vez que haya un servicio nuevo, habría que configurarlo en el firewall.

En el siguiente fichero mostraré las reglas del cortafuegos principal:

```
#clear
#echo 'Aplicando reglas de firewall ...'
#echo "echo '1' > /proc/sys/net/ipv4/ip_forward <-- recuerda!"
#echo 'Recuerda si estas usando la interfaz correcta en enp0s3!'

# FLUSH de reglas

iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Establecemos politica por defecto

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

# Configuracion VPN

iptables -I INPUT -p udp --dport 1194 -j ACCEPT
iptables -I OUTPUT -p udp --sport 1194 -j ACCEPT

iptables -A FORWARD -i tun+ -o enp0s8 -j ACCEPT
iptables -A FORWARD -o tun+ -i enp0s8 -j ACCEPT

# Reglas para que los usuarios de la conexión VPN puedan conectarse al servidor SAMBA

iptables -t nat -A PREROUTING -i tun+ -p tcp --match multiport --dport 135,139,445 \
-j DNAT --to-destination 192.168.40.1

# Configuracion VPN para que puedan acceder al servidor samba 192.168.40.1/32 . Esta es otra forma, son más reglas pero más especificas. Recordad que mientras más especifica sea, mejor. Si con esta especificación consigues hacer la funcionalidad que quieres, conseguirás más seguridad.

#iptables -A FORWARD -i tun+ -o enp0s8 -p tcp --dport 135 -j ACCEPT
#iptables -A FORWARD -o tun+ -i enp0s8 -p tcp --sport 135 -j ACCEPT
#iptables -A FORWARD -i tun+ -o enp0s8 -p tcp --dport 139 -j ACCEPT
#iptables -A FORWARD -o tun+ -i enp0s8 -p tcp --sport 139 -j ACCEPT
#iptables -A FORWARD -i tun+ -o enp0s8 -p tcp --dport 445 -j ACCEPT
#iptables -A FORWARD -o tun+ -i enp0s8 -p tcp --sport 445 -j ACCEPT
#iptables -A FORWARD -i tun+ -o enp0s8 -p udp --dport 137 -j ACCEPT
#iptables -A FORWARD -o tun+ -i enp0s8 -p udp --sport 137 -j ACCEPT
#iptables -A FORWARD -i tun+ -o enp0s8 -p udp --dport 138 -j ACCEPT
#iptables -A FORWARD -o tun+ -i enp0s8 -p udp --sport 138 -j ACCEPT
```

 # reglas para el servidor Squid

```
iptables -A INPUT -p udp --sport 53 -j ACCEPT
iptables -A INPUT -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -p udp --dport 53 -j ACCEPT

iptables -A INPUT -p tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT

iptables -A INPUT -p tcp --sport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
```

 # Reglas para que la RED COMÚN pueda navegar utilizando SQUID

La intención de estas reglas era redireccionar dichos puertos para no configurar los navegadores de los clientes.

```
#iptables -t nat -I PREROUTING -s 192.168.30.0/24 -i enp0s9 -p tcp --dport 443 -j
REDIRECT \
#--to-port 3128
#iptables -t nat -I PREROUTING -s 192.168.30.0/24 -i enp0s9 -p tcp --dport 80 -j
REDIRECT \
#--to-port 3128
```

 # Son reglas para poder ver con detalle los paquetes que van cumpliendo con estas características.

```
#iptables -A INPUT -p tcp -j LOG --log-prefix '**ENTRADA**'
#iptables -A OUTPUT -p tcp -j LOG --log-prefix '**SALIDA**'
```

 # Reglas, para que la red 192.168.30.0 (RED COMÚN) pueda acceder a internet (Si no utilizásemos el servidor SQUID).

```
#iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -o enp0s3 -p tcp --dport 80 -j
ACCEPT
#iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -i enp0s3 -p tcp --sport 80 -j
ACCEPT

#iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -o enp0s3 -p tcp --dport 443 -j
ACCEPT
#iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -i enp0s3 -p tcp --sport 443 -j
ACCEPT

#iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -o enp0s3 -p tcp --dport 53 -j
ACCEPT
#iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -i enp0s3 -p tcp --sport 53 -j
ACCEPT
```

```
#iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -o enp0s3 -p udp --dport 53 -j
ACCEPT
#iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -i enp0s3 -p udp --sport 53 -j
ACCEPT

# Reglas, para que la red 192.168.30.0 pueda acceder a 192.168.40.2 (servidor web)

iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.2/32 -i enp0s9 -o enp0s8 -p tcp
--dport 80 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.2/32 -o enp0s9 -i enp0s8 -p tcp
--sport 80 -j ACCEPT

# Reglas para aceptar peticiones de tipo broadcast destinado al servidor DHCP

iptables -A INPUT -p udp --sport 68 --dport 67 -m addrtype --dst-type BROADCAST -j
ACCEPT
iptables -A OUTPUT -p udp --dport 68 --sport 67 -m addrtype --dst-type BROADCAST -j
ACCEPT

# Reglas para que la red 192.168.30.0 puedan enviar pings a otras redes

iptables -A INPUT -p ICMP -i enp0s9 -j ACCEPT
iptables -A OUTPUT -p ICMP -o enp0s9 -j ACCEPT


iptables -A INPUT -s 192.168.30.0/24 -p tcp --dport 3128 -j ACCEPT
iptables -A OUTPUT -d 192.168.30.0/24 -p tcp --sport 3128 -j ACCEPT

# Reglas para redireccionar los puertos 80,443 al puerto 3128 (Squid). Con el
puerto 80 funciona, pero con el puerto 443 me daba fallos sobre SSL, para poder
solventar este problema habría que instalar en cada uno de los clientes un
certificado digital.  Con este certificado conseguiríamos mas seguridad.
# Reglas correo electronico IMAP y SMTP

iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -p tcp --dport 587 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -p tcp --sport 587 -j ACCEPT

iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -p tcp --dport 993 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -p tcp --sport 993 -j ACCEPT

# Reglas, para que la red 192.168.30.0 pueda acceder al servidor samba
# que esta en la red 192.168.40.0


iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.1/32 -i enp0s9 -o enp0s8 -p tcp
--dport 135 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.1/32 -o enp0s9 -i enp0s8 -p tcp
--sport 135 -j ACCEPT

iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.1/32 -i enp0s9 -o enp0s8 -p tcp
```

```
--dport 139 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.1/32 -o enp0s9 -i enp0s8 -p tcp
--sport 139 -j ACCEPT

iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.1/32 -i enp0s9 -o enp0s8 -p tcp
--dport 445 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.1/32 -o enp0s9 -i enp0s8 -p tcp
--sport 445 -j ACCEPT

iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.1/32 -i enp0s9 -o enp0s8 -p udp
--dport 137 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.1/32 -o enp0s9 -i enp0s8 -p udp
--sport 137 -j ACCEPT

iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.1/32 -i enp0s9 -o enp0s8 -p udp
--dport 138 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.1/32 -o enp0s9 -i enp0s8 -p udp
--sport 138 -j ACCEPT
```

 # reglas resumidas, para que la red 192.168.30.0 pueda acceder al servidor samba que esta en la red 192.168.40.0 (otra manera de hacerlo)

```
#iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -p tcp --match multiport --dport
135,139,445 -j ACCEPT
#iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -p tcp --match multiport --sport
135,139,445 -j ACCEPT

#iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -p udp --match multiport --dport
137,138 -j ACCEPT
#iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -p udp --match multiport --sport
137,138 -j ACCEPT
```

 # reglas ping en general

```
iptables -A FORWARD -p ICMP --icmp-type echo-request -j ACCEPT
iptables -A FORWARD -p ICMP --icmp-type echo-reply -j ACCEPT
```

 # reglas ping para el cortafuegos

```
iptables -A INPUT -p ICMP --icmp-type echo-reply -j ACCEPT
iptables -A OUTPUT -p ICMP --icmp-type echo-request -j ACCEPT
```

 # regla SSH. Esta regla bloqueara IP si intenta realizar 3 intentos de conexion por minuto. El estado se establece a NEW , esto significa que solo las nuevas conexiones no establecidas son afectadas. Las conexiones establecidas son el resultado de una autenticacion SSH exitosa, por lo que los usuarios que se autentiquen correctamente no serán bloqueados. los bloqueados estarán desbaneados a los 2 minutos.

```
iptables -A FORWARD -p tcp --dport 22 -i enp0s9 -m state --state NEW -m recent --set
```

iptables -A FORWARD -p tcp --dport 22 -i enp0s9 -m state --state NEW -m recent \
--update --seconds 60 --hitcount 3 -j DROP

 # Video ejemplo SSH 3 intentos.

 # Regla que no permite más de una conexión por IP en SSH. En el caso de haya una conexión activa, no se podrá realizar ninguna conexión más.

iptables -A FORWARD -i enp0s9 -o enp0s8 -p tcp --syn --dport 22 \
-m connlimit --connlimit-above 1 -j REJECT --reject-with tcp-reset

 # Video ejemplo una conexión por IP en SSH parte 1
 # Video ejemplo una conexión por IP en SSH parte 2

 # Regla SSH permitiendo conectarse desde la red 30.0 al equipo 40.1

iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.40.1/32 -i enp0s9 -o enp0s8 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 192.168.30.0/24 -s 192.168.40.1/32 -o enp0s9 -i enp0s8 -p tcp --sport 22 -j ACCEPT

 # Regla que realiza source nat desde la red 30.0 a la salida de la tarjeta de red enp0s3 (la interfaz próxima a la red del router). En este caso esta en MASQUERADE porque la interfaz enp0s3 recibe por DHCP del router la configuración de red.
En caso de tener una dirección ip estática, habría que especificarla para quitar esa carga innecesaria a IPTABLES. A continuación muestro las dos formas.

iptables -t nat -A POSTROUTING -s 192.168.30.0/24 -o enp0s3 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.30.0/24 -o enp0s3 -j SNAT –-to 192.168.20.254

5.1 Reglas Cortafuegos relacionados con Internet

Hay unos parámetros que son necesarios nombrar ya que son importantes cuando los paquetes van con destino a Internet o cuando ofrecemos un servicio a Internet. Estas reglas son:

-m state - -state NEW,ESTABLISHED (peticion)
-m state - -state ESTABLISHED (respuesta)

Un ejemplo para entender el significado de estas reglas es dejar el firewall con política **DROP** . y aplicar unas reglas para aceptar conexiones SSH en ambos sentidos:

iptables -A FORWARD -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -p tcp --sport 22 -j ACCEPT

En este escenario tanto el destino como el origen pueden realizar la conexión y si conocen el usuario y contraseña podrán autenticarse. **La conexión se puede realizar en ambos sentidos.**

Ahora añadiremos las reglas **-m state** para ver el efecto:

```
iptables -A FORWARD -p tcp --dport 22 -m state –state \ NEW,ESTABLISHED -J ACCEPT
iptables -A FORWARD -s 192.168.40.2/32 -p tcp --sport 22 -m state \
--state ESTABLISHED -j ACCEPT
```

En este ejemplo, todos podrán conectarse al servidor SSH 192.168.40.3 pero el servidor 192.168.40.3 no podrá conectarse con nadie. **La conexión se realiza en un solo sentido**.

A continuación dejo un par de vídeos de este apartado:

Ubuntu Desktop 18 04 iptables -m state parte 1
Ubuntu Desktop 18 04 iptables -m state parte 2

6. Configuración tarjetas de red del servidor principal.

Muestro a continuación la configuración de red del servidor principal:

```
# This file is generated from information provided by
# the datasource.  Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        enp0s3:
            addresses: [192.168.20.254/24]
            nameservers:
             addresses: [1.1.1.1, 8.8.8.8]
            dhcp4: true
            routes:
              - to: 0.0.0.0/0
                via: 192.168.20.1
                on-link: true
              - to: 192.168.10.0/24
                via: 192.168.20.1
              - to: 192.168.1.0/24
                via: 192.168.20.1
        enp0s8:
            addresses: [192.168.40.254/24]
            nameservers:
             addresses: [1.1.1.1, 8.8.8.8]
```

```
    enp0s9:
      addresses: [192.168.30.254/24]
      nameservers:
        addresses: [1.1.1.1, 8.8.8.8]
  version: 2
```

En la parte de las rutas, hay un parámetro nuevo llamado **on-link** , es la ruta por defecto en caso de que no haya una ruta mas restrictiva se escogerá la ruta por defecto.

7. Ejecutar script automáticamente al iniciar el equipo.

Ejecutar el script automáticamente al arrancar la máquina. Hoy en día los sistemas GNU/Linux utilizan mayoritariamente systemd en lugar de init System V, por lo que explicaremos brevemente como definir una unidad de systemd que ejecute el script de iptables al arrancar el equipo.

Creamos el fichero iptables.service en el directorio /etc/systemd/system/, con el siguiente contenido:

```
[Unit]
Description=Reglas de iptables
After=systemd-sysctl.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/iptables.sh

[Install]
WantedBy=multi-user.target
```

Automatizar script al inicio (parte 1)

Tendremos que habilitar la unidad anterior para que se ejecute la próxima vez y si queremos en este momento arrancarla:

```
systemctl enable iptables.service
systemctl start iptables.service
```

Automatizar script al inicio (parte 2)

Ya solo nos queda reiniciar y al ejecutar el comando **sudo iptables -L -nv** se cargará todas las reglas sin necesidad de cargar el script manualmente. Además también activa el bit de forward en **/proc/sys/net/ipv4/ip_forward**

8. Trabajar con cadenas para gestionar mejor las reglas

Es muy importante gestionar las reglas de la manera más efectiva posible ya que cuando hay muchas reglas y surge algún problema, podríamos demorarnos demasiado en solucionarlo.

Por ejemplo, veamos una vista general de mis reglas de iptables en el servidor principal con el comando **sudo iptables -L -nv**:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:1194
   20  2796 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp spt:53
    8   600 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:53
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp spt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp spt:443
    0     0 ACCEPT     tcp  --  *      *       192.168.30.0/24      0.0.0.0/0           tcp dpt:3128
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp spt:68 dpt:67 ADDRTYPE
match dst-type BROADCAST
    0     0 ACCEPT     icmp --  enp0s9 *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0           icmptype 0

Chain FORWARD (policy DROP 1769 packets, 106K bytes)
 pkts bytes target     prot opt in     out     source               destination
   23  1380 REJECT     tcp  --  enp0s9 enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:22 flags:0x17/0x02
#conn src/32 > 1 reject-with tcp-reset
    0     0 DROP       tcp  --  enp0s9 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22 state NEW recent:
UPDATE seconds: 60 hit_count: 3 name: DEFAULT side: source mask: 255.255.255.255
    4   240 DROP       tcp  --  enp0s9 *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22 state NEW recent: SET
name: DEFAULT side: source mask: 255.255.255.255
  190 22092 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22 state
NEW,ESTABLISHED
  189 27812 ACCEPT     tcp  --  *      *       192.168.40.2         0.0.0.0/0           tcp spt:22 state
ESTABLISHED
    0     0 ACCEPT     tcp  --  tun+   enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:135
    0     0 ACCEPT     tcp  --  enp0s8 tun+    0.0.0.0/0            0.0.0.0/0           tcp spt:135
    0     0 ACCEPT     tcp  --  tun+   enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:139
    0     0 ACCEPT     tcp  --  enp0s8 tun+    0.0.0.0/0            0.0.0.0/0           tcp spt:139
    0     0 ACCEPT     tcp  --  tun+   enp0s8  0.0.0.0/0            0.0.0.0/0           tcp dpt:445
    0     0 ACCEPT     tcp  --  enp0s8 tun+    0.0.0.0/0            0.0.0.0/0           tcp spt:445
    0     0 ACCEPT     udp  --  tun+   enp0s8  0.0.0.0/0            0.0.0.0/0           udp dpt:137
    0     0 ACCEPT     udp  --  enp0s8 tun+    0.0.0.0/0            0.0.0.0/0           udp spt:137
    0     0 ACCEPT     udp  --  tun+   enp0s8  0.0.0.0/0            0.0.0.0/0           udp dpt:138
    0     0 ACCEPT     udp  --  enp0s8 tun+    0.0.0.0/0            0.0.0.0/0           udp spt:138
    0     0 ACCEPT     tcp  --  enp0s9 enp0s8  192.168.30.0/24      192.168.40.2        tcp dpt:80
    0     0 ACCEPT     tcp  --  enp0s8 enp0s9  192.168.40.2        192.168.30.0/24     tcp spt:80
    0     0 ACCEPT     tcp  --  enp0s9 *       192.168.30.0/24      0.0.0.0/0           tcp dpt:587
    0     0 ACCEPT     tcp  --  *      enp0s9  0.0.0.0/0            192.168.30.0/24     tcp spt:587
    0     0 ACCEPT     tcp  --  enp0s9 *       192.168.30.0/24      0.0.0.0/0           tcp dpt:993
    0     0 ACCEPT     tcp  --  *      enp0s9  0.0.0.0/0            192.168.30.0/24     tcp spt:993
    0     0 ACCEPT     tcp  --  enp0s9 enp0s8  192.168.30.0/24      192.168.40.1        tcp dpt:135
```

```
    0     0 ACCEPT     tcp  --  enp0s8 enp0s9  192.168.40.1        192.168.30.0/24      tcp spt:135
    0     0 ACCEPT     tcp  --  enp0s9 enp0s8  192.168.30.0/24     192.168.40.1         tcp dpt:139
    0     0 ACCEPT     tcp  --  enp0s8 enp0s9  192.168.40.1        192.168.30.0/24      tcp spt:139
    0     0 ACCEPT     tcp  --  enp0s9 enp0s8  192.168.30.0/24     192.168.40.1         tcp dpt:445
    0     0 ACCEPT     tcp  --  enp0s8 enp0s9  192.168.40.1        192.168.30.0/24      tcp spt:445
    0     0 ACCEPT     udp  --  enp0s9 enp0s8  192.168.30.0/24     192.168.40.1         udp dpt:137
    0     0 ACCEPT     udp  --  enp0s8 enp0s9  192.168.40.1        192.168.30.0/24      udp spt:137
    0     0 ACCEPT     udp  --  enp0s9 enp0s8  192.168.30.0/24     192.168.40.1         udp dpt:138
    0     0 ACCEPT     udp  --  enp0s8 enp0s9  192.168.40.1        192.168.30.0/24      udp spt:138
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0          0.0.0.0/0            icmptype 8
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0          0.0.0.0/0            icmptype 0
    0     0 ACCEPT     tcp  --  enp0s9 enp0s8  192.168.30.0/24     192.168.40.1         tcp dpt:22
    0     0 ACCEPT     tcp  --  enp0s8 enp0s9  192.168.40.1        192.168.30.0/24      tcp spt:22

Chain OUTPUT (policy DROP 39 packets, 2800 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0            udp spt:1194
    8  1224 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0            udp spt:53
   20  1452 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0            udp dpt:53
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0          0.0.0.0/0            tcp dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0          0.0.0.0/0            tcp spt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0          0.0.0.0/0            tcp dpt:443
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0          0.0.0.0/0            tcp spt:443
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0          192.168.30.0/24      tcp spt:3128
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0          0.0.0.0/0            udp spt:67 dpt:68 ADDRTYPE
match dst-type BROADCAST
    0     0 ACCEPT     icmp --  *      enp0s9  0.0.0.0/0          0.0.0.0/0
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0          0.0.0.0/0            icmptype 8
```

Para la creación de nuevas cadenas se podría hacer por ejemplo así:

```
# Nuevas cadenas

iptables -N INTERNET_A_VPN
iptables -N VPN_A_INTERNET
iptables -N 30.0_A_40.0
iptables -N 40.0_A_30.0
```

Vamos a gestionar las reglas que veamos que se repiten mucho exceptuando algunos parametros. Por ejemplo:

```
# Configuracion VPN para que puedan acceder al servidor samba
192.168.40.1/32

iptables -A FORWARD -i tun+ -o enp0s8 -j INTERNET_A_VPN
iptables -A FORWARD -o tun+ -i enp0s8 -j VPN_A_INTERNET

iptables -A INTERNET_A_VPN -p tcp --dport 135 -j ACCEPT
iptables -A INTERNET_A_VPN -p tcp --dport 139 -j ACCEPT
iptables -A INTERNET_A_VPN -p tcp --dport 445 -j ACCEPT
iptables -A INTERNET_A_VPN -p udp --dport 137 -j ACCEPT
iptables -A INTERNET_A_VPN -p udp --dport 138 -j ACCEPT

iptables -A VPN_A_INTERNET -p tcp --sport 135 -j ACCEPT
iptables -A VPN_A_INTERNET -p tcp --sport 139 -j ACCEPT
```

```
iptables -A VPN_A_INTERNET -p tcp --sport 445 -j ACCEPT
iptables -A VPN_A_INTERNET -p udp --sport 137 -j ACCEPT
iptables -A VPN_A_INTERNET -p udp --sport 138 -j ACCEPT
# REGLAS CADENAS 30.0_A_40.0 y 40.0_A_30.0

iptables -A FORWARD -s 192.168.30.0/24 -i enp0s9 -o enp0s8 -j 30.0_A_40.0
iptables -A FORWARD -d 192.168.30.0/24 -o enp0s9 -i enp0s8 -j 40.0_A_30.0

# REGLAS CADENA 30.0_A_40.0 PETICIONES

iptables -A 30.0_A_40.0 -d 192.168.40.2/32 -p tcp --dport 80 -j ACCEPT
iptables -A 30.0_A_40.0 -d 192.168.40.1/32 -p tcp --dport 135 -j ACCEPT
iptables -A 30.0_A_40.0 -d 192.168.40.1/32 -p tcp --dport 139 -j ACCEPT
iptables -A 30.0_A_40.0 -d 192.168.40.1/32 -p tcp --dport 445 -j ACCEPT
iptables -A 30.0_A_40.0 -d 192.168.40.1/32 -p udp --dport 137 -j ACCEPT
iptables -A 30.0_A_40.0 -d 192.168.40.1/32 -p udp --dport 138 -j ACCEPT
iptables -A 30.0_A_40.0 -d 192.168.40.1/32 -p tcp --dport 22 -j ACCEPT

# REGLAS CADENA 40.0_A_30.0 RESPUESTAS

iptables -A 40.0_A_30.0 -s 192.168.40.2/32 -p tcp --sport 80 -j ACCEPT
iptables -A 40.0_A_30.0 -s 192.168.40.1/32 -p tcp --sport 135 -j ACCEPT
iptables -A 40.0_A_30.0 -s 192.168.40.1/32 -p tcp --sport 139 -j ACCEPT
iptables -A 40.0_A_30.0 -s 192.168.40.1/32 -p tcp --sport 445 -j ACCEPT
iptables -A 40.0_A_30.0 -s 192.168.40.1/32 -p udp --sport 137 -j ACCEPT
iptables -A 40.0_A_30.0 -s 192.168.40.1/32 -p udp --sport 138 -j ACCEPT
iptables -A 40.0_A_30.0 -s 192.168.40.1/32 -p tcp --sport 22 -j ACCEPT
```

A simple vista parece que no ha cambiado mucho, el resultado se ve a la hora de ejecutar el comando **iptables -L -nv.** Veamos un pequeño video para tener más claro la diferencia entre trabajar con cadenas y no trabajarlas.

Video ejemplo de la diferencia entre usar cadenas y no usarlas:
iptables_con_cadenas

9. Herramientas para analizar el tráfico que pasa por la interfaz de red.

9.1 TCPDUMP

Tcpdump es un herramienta en línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red.

Permite al usuario capturar y mostrar a tiempo real los paquetes transmitidos y recibidos en la red a la cual el ordenador está conectado. **Tcpdump** funciona en la mayoría de los sistemas operativos UNIX.

9.1.1 Filtrado básico

En la práctica no ha hecho falta añadir ningún filtro porque hay poco tráfico en la red. Pero en un escenario el cual haya un tráfico considerable, nos interesará más filtrar para conseguir una salida más limpia permitiendo un mejor análisis del problema que estamos teniendo en la red. El problema más típico sería el de el porque no está permitiendo una regla del cortafuegos. Con esta herramienta podemos saber si nos está llegando a la interfaz de red las peticiones o no. A continuación dejo algunos ejemplos:

Captura los paquetes que vienen de 192.168.1.11:

```
tcpdump -i enp0s3 src 192.168.1.11
```

Captura los paquetes que van hacia 192.168.1.11:

```
tcpdump -i enp0s3 dst 192.168.1.11
```

Captura los paquetes con el origen 192.168.1.11 y destinado a 192.168.1.35:

```
tcpdump -i enp0s3  src 192.168.1.11 and dst 192.168.1.35
```

Captura los paquetes de origen 192.168.1.11 con destino 192.168.1.35 con el puerto de destino 25.

```
tcpdump -i enp0s3 -n -S src 192.168.1.11 and dst 192.168.1.35 and dst port 25
```

Video ejemplo tcpdump escuchando la interfaz enp0s3 del firewall:

tcpdump -nvv -i enp0s3 tcp

También se podría realizar scripts para detectar uso mal intencionado en nuestra red local. Por ejemplo si quisiéramos detectar que equipos de nuestra red local están realizando pings a nuestro cortafuegos, podríamos detectarlo con el siguiente script:

```
while [ true ] do;

        ip=$(tcpdump -c 1 -ni enp0s8 'icmp and (icmp[0] = 0x08)' | cut -d " " -f 3)

        if ! [ -z "$ip" ]; then

                echo "La ip $ip realizó un ping a la interfaz enp0s8"
                #iptables -A INPUT -p tcp -s $ip -j DROP
                #iptables -A INPUT -p icmp-type 8 -s $ip -j DROP
                exit 100
        fi
done
```

Esto sería un ejemplo de como podríamos automatizar medidas de defensa según el escenario.

Dejo un video ejemplo para verlo:
sudo ./tcpdump.sh

9.2 Activando la cadena LOG:

Aplicamos la regla para activar la cadena LOG en nuestro fichero de iptables:

```
iptables -A INPUT -s 10.0.3.0/24 -p tcp --dport 80 -j LOG --log-prefix '** PUERTO 80 **'
```

Los paquetes que entren a la red 10.0.3.0/24 que utilicen el protocolo tcp con destino al puerto 80 active la cadena LOG con el mensaje ** PUERTO 80 **.

Con esta regla estamos introduciendo en el log **/var/log/kern.log** el mensaje ** PUERTO 80 ** para identificar mas fácilmente los paquetes que queremos analizar.

```
Feb 12 12:26:46 madmb-VirtualBox kernel: [  824.083501] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=44 TOS=0x00
PREC=0x00 TTL=64 ID=7624 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 SYN
URGP=0
Feb 12 12:26:46 madmb-VirtualBox kernel: [  824.083739] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=7625 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK
URGP=0
Feb 12 12:26:46 madmb-VirtualBox kernel: [  824.083895] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=518 TOS=0x00
PREC=0x00 TTL=64 ID=7626 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK PSH
URGP=0
Feb 12 12:26:46 madmb-VirtualBox kernel: [  824.086871] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=7627 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK
URGP=0
Feb 12 12:26:46 madmb-VirtualBox kernel: [  824.086887] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=7628 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK
URGP=0
Feb 12 12:26:46 madmb-VirtualBox kernel: [  824.086897] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=7629 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK
URGP=0
Feb 12 12:26:47 madmb-VirtualBox kernel: [  824.321925] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=398 TOS=0x00
PREC=0x00 TTL=64 ID=7630 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK PSH
URGP=0
Feb 12 12:26:47 madmb-VirtualBox kernel: [  824.323067] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=7631 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK
URGP=0
Feb 12 12:26:52 madmb-VirtualBox kernel: [  829.321075] ** PUERTO 80 **IN=enp0s8 OUT=
MAC=08:00:27:06:59:4d:52:54:00:12:35:02:08:00 SRC=10.0.3.2 DST=10.0.3.15 LEN=40 TOS=0x00
PREC=0x00 TTL=64 ID=7632 PROTO=TCP SPT=57982 DPT=80 WINDOW=65535 RES=0x00 ACK
URGP=0
```

Procedo a explicar casi todos los campos de cada paquete:

IN: Interfaz de entrada (enp0s8)
OUT: Interfaz de salida (en este caso esta vacío porque solo vienen paquetes entrantes)
MAC, hay que dividirlo por partes para entenderlo:

- 08:00:27:06:59:4d (10.0.3.15) enp0s8
- 52:54:00:12:35:02 (10.0.3.2) enp0s8 (la pasarela)
- 08:00 (protocolo que esta utilizando en la capa 3, en este caso IPv4)

SRC: IP origen
DST: IP destino
LEN: tamaño del paquete
TTL: Tiempo de vida del paquete
ID: Identificador del paquete (si nos fijamos, es incremental)
PROTO: protocolo (en este caso TCP)
SPT: Puerto origen
DPT: Puerto destino
FLAG´s TCP (hay más):

- **SYN**: inicia la conexión entre hosts.
- **ACK**: Reconoce la recepción de un paquete.
- **PSH**: Envia todos los datos almacenados al buffer.

Como podemos comprobar, el primer paquete se envia con el flag **SYN**, para establecer la conexión, en el siguiente paquete recibe un **ACK** de confirmación. Se ha establecido conexión.

10. IPTABLES con IP´s dinámicas (NoIP)

Si en casa utilizas un servicio DNS gratuito como NoIP, puedes filtrar el acceso con iptables. Así, cada vez que la dirección IP de NoIP cambie, el cortafuegos lo detectará y permitirá el acceso a la nueva IP actualizada de NoIP.

```
#!/bin/bash

DNS_HOSTNAME=madmb.ddns.net
LOGFILE=/tmp/ddns.log
CADENA="noip_hosts"

IP_actual=$(host $DNS_HOSTNAME | cut -f4 -d ' ' |  head -n 1)

if [ $LOGFILE = "" ] ; then

        iptables -I $CADENA -i enp0s3 -s $IP_actual -p tcp --dport ssh -j ACCEPT
        echo $IP_actual > $LOGFILE
else

IP_anterior=$(cat $LOGFILE)

        if [ "$IP_actual" = "$IP_anterior" ] ; then

                echo "La direccion IP no ha cambiado"
```

```
        else
                iptables -D $CADENA -i enp0s3 -s $IP_anterior -p tcp --dport ssh -j ACCEPT
                iptables -I $CADENA -i enp0s3 -s $IP_actual -p tcp --dport ssh -j ACCEPT
                echo $IP_actual > $LOGFILE
                echo "iptables actualizadas"
        fi
fi
```

Ya solo queda añadir en crontab y que se ejecute cada 5 minutos por ejemplo.

```
*/5 * * * * /root/iptables_ip_dinamica.sh > /dev/null 2>&1
```

11. Port knocking

Es un mecanismo para abrir puertos externamente si se recibe conexiones a puertos cerrados. Si por ejemplo se recibe una secuencia de conexiones al puerto 1000, 2000, 3000, se abrirá el puerto SSH. Si no se recibiese esa secuencia, el puerto SSH estaría cerrado. También podríamos cerrar el puerto SSH con otra secuencia de conexiones.

Para instalarlo:

```
Sudo apt-get install knockd
```

Para realizar estas conexiones podemos usar la herramienta NMAP, con el siguiente comando:

```
nmap -sT -p7000,8000,9000 host
```

El fichero de configuración:

```
cat /etc/knockd.conf

[options]
        logfile = /var/log/knockd.log

[openSSH]
        sequence    = 7000,8000,9000
        seq_timeout = 10
        command     = /usr/sbin/iptables -I TRAFFIC -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn

[closeSSH]
        sequence    = 9000,8000,7000
        seq_timeout = 10
        command     = /usr/sbin/iptables -D TRAFFIC -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn
```